An extended ant colony optimization algorithm for integrated process and control system design

Martin Schlüter[a], Jose A. Egea[b], Luis T. Antelo[b], Antonio A. Alonso[b], Julio R. Banga[b]
schluetm@maths.bham.ac.uk, ltaboada@iim.csic.es, jegea@iim.csic.es, antonio@iim.csic.es, julio@iim.csic.es

[a] *Theoretical & Computational Optimization Group, University of Birmingham*
Birmingham B15 2TT, United Kingdom
[b] *Process Engineering Group, Instituto de Investigaciones Marinas (IIM-CSIC)*
36208 Vigo, Spain

6th May 2009

Abstract

The problem of integrated process and control system design is discussed in this paper. We formulate it as a mixed integer nonlinear programming problem subject to differential-algebraic constraints. This class of problems is frequently non-convex and, therefore, local optimization techniques usually fail to locate the global solution. Here we propose a global optimization algorithm, based on an extension of the Ant Colony Optimization metaheuristic, in order to solve this challenging class of problems in an efficient and robust way. The ideas of the methodology are explained and, on the basis of different full-plant case studies, the performance of the approach is evaluated. The first set of benchmark problems deal with the integrated design and control of two different wastewater treatment plants, consisting on both NLP and MINLP formulations. The last case study is the well-known Tennessee Eastman Process. Numerical experiments with our new method indicate that we can achieve an improved performance in all cases. Additionally, our method outperforms several other recent competitive solvers for the challenging case studies considered.

Keywords: Integrated process and control system design, mixed integer nonlinear programming (MINLP), metaheuristics, Ant Colony Optimization, Tennessee Eastman Process, oracle penalty method.

1 Introduction

The general statement of the simultaneous (integrated) design and control problem takes into account the process and control superstructures, indicating the different design alternatives.[1–3] This general approach results in mixed integer nonlinear programming problems. The aim is to simultaneously find the static variables of the process design as well as the operating conditions and the controllers' parameters which optimize a combined measure of the plant economics and its controllability, subject to a set of constraints which ensure appropriate dynamic behavior and process specifications. Here we will consider that the physical design of the plants are given and the parameters to be optimized are related to operational and controllability issues. We state our problem as follows:

$$\min_{\mathbf{v},\mathbf{b}} C(\mathbf{z},\mathbf{v},\mathbf{b}) = \sum_i w_i \cdot \phi_i \tag{1}$$

subject to

$$\begin{aligned} \mathbf{f}(\dot{\mathbf{z}}, \mathbf{z}, \mathbf{p}, \mathbf{v}, \mathbf{b}, t) &= 0 \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \\ \mathbf{h}(\mathbf{z}, \mathbf{p}, \mathbf{v}, \mathbf{b}) &= 0 \\ \mathbf{g}(\mathbf{z}, \mathbf{p}, \mathbf{v}, \mathbf{b}) &\geq 0 \\ \mathbf{v}^L \leq \mathbf{v} &\leq \mathbf{v}^U \\ \mathbf{b}^L \leq \mathbf{b} &\leq \mathbf{b}^U \end{aligned} \tag{2}$$

where $\mathbf{v} \in \mathbb{R}^{ncont}$ and $\mathbf{b} \in \mathbb{R}^{nint}$ are, respectively, the vector of continuous and integer (including binary) decision variables (e.g., design variables, operating conditions, parameters of controllers, set points, etc.); C is the cost (objective function) to minimize (normally a weighted combination of capital, operation and controllability costs, ϕ_i); $\mathbf{f}$ is a functional for the system dynamics (i.e., the nonlinear process model); $\mathbf{z}$ is the vector of the states (and $\dot{\mathbf{z}}$ is its derivative); $\mathbf{p}$ is a set of time-invariant parameters; t_0 is the initial time for the integration of the system of differential algebraic equations (and, consequently, $\mathbf{z}_0$ is the vector of the states at that initial time); $\mathbf{h}$ and $\mathbf{g}$ are possible equality and inequality path and/or point constraint functions which express additional requirements for the process performance; and, finally, $\mathbf{v}^L$ and $\mathbf{v}^U$, $\mathbf{b}^L$ and $\mathbf{b}^U$ are the lower and upper bounds for the decision variables. The dependence of ϕ_i upon the decision variables, $\mathbf{v}, \mathbf{b}$, is defined by the problem formulation. In some cases this dependence is simple and straightforward (i.e., when minimizing the cost of a chemical process, one of the ϕ_i can be equal to a reactor size, which may also be a decision variable), whereas in others there might not be an explicit expression for this dependence (i.e., ϕ_i can be the integral square error, *ISE*, of a controller which, in general, does not depend explicitly on the decision variables).

Typically, most of the problems in engineering applications are highly constrained and exhibit non-linear dynamics. These properties often result in non-convexity and multimodality. Furthermore, in many complex process models some kind of noise and/or discontinuities (either due to numerical methods, or to intrinsic properties of the model) are present. Therefore, there is a need of robust global optimization solvers which can locate the vicinity of the global solution in a reasonable number of iterations and handle noise and/or discontinuities.

Since some of the decision variables in these problems can have an integer or binary nature, the formulation above results in a mixed integer nonlinear program (MINLP). In this paper we will discuss a hybrid metaheuristic approach that was developed to solve complex non-convex mixed integer optimization problems. Combining the robustness of an extended Ant Colony Optimization framework with the precision of a mixed integer sequential quadratic programming algorithm, we obtain a powerful global solver for general MINLP problems. The job of the ACO framework is to locate the attraction basin of the global minimum and to start then the MISQP routine as a local solver. In case the local solver is still not able to obtain the global minimum, the ACO metaheuristic will be used again. This is done to improve the current solution by a new ACO search process concentrated within the former located attraction basin.

This paper is structured as follows: the ACO framework used in this study, based on extensions to a previous work[4] is described in Section 2. A novel strategy to handle constraints (the robust oracle penalty method) is presented in Section 3. Details on a set of novel heuristics included in the algorithm's design and on its implementation are provided in Section 4. Section 5 presents the case studies considered in this work and provides numerical results for each of them. The paper finishes with the associated conclusions.

2 The ant colony optimization framework

The original ACO metaheuristic was developed by Dorigo[7] and was intended for combinatorial optimization problems only. The ACO framework considered in this paper is based on an extended

ACO methodology for continuous domains proposed by Socha and Dorigo.[8] This methodology works by the incremental construction of solutions regarding a probabilistic choice according to a probability density function (PDF). In principle any function $P(x) \geq 0$ for all x with the property:

$$\int_{-\infty}^{\infty} P(x)\ dx \ = \ 1 \tag{3}$$

can act as a PDF. One of the most popular functions to be used as a PDF is the Gaussian function. This function has some clear advantages like an easy implementation[10] and a corresponding fast sampling time of random numbers. On the other hand, a single Gaussian function is only able to focus on one mean and therefore not able to describe situations where two or more disjoint areas of the search domain are promising. To overcome this drawback and still taking advantage of the benefits of the Gaussian function, a PDF $G^i(x)$ consisting of a weighted sum of several one-dimensional Gaussian functions $g_l^i(x)$ is considered for every dimension i of the original search domain:

$$G^i(x) \ = \ \sum_{l=1}^{k} w_l^i \cdot g_l^i(x) \ = \ \sum_{l=1}^{k} w_l^i \frac{1}{\sigma^i \sqrt{2\pi}}\ e^{-\frac{(x-\mu_l^i)^2}{2\ {\sigma^i}^2}} \tag{4}$$

This function is characterized by the triplets $(w_l^i, \sigma^i, \mu_l^i)$ that are given for every dimension i of the search domain and the number of kernels k of Gauss functions used within $G^i(x)$. Within this triplet, w denotes the weights for the individual Gaussian functions for the PDF, σ represents the standard deviations, and μ are the means for the corresponding Gaussian functions. The indices i and l refer, respectively, to the i-th dimension of the decision vector of the MINLP problem and the l-th kernel number of the individual Gaussian function within the PDF.

Since the triplets above fully characterize the PDF and, therefore, guide the sampled solution candidates throughout the search domain, they are called pheromones in the ACO sense and constitute the biological background of the ACO metaheuristic presented here. Besides the incremental construction of the solution candidates according to the PDF, the update of the pheromones plays a major role in the ACO metaheuristic.

A good and obvious choice to update the pheromones is the use of information, which has been gained throughout the search process. This can be done by using a solution archive SA in which the so far most promising solutions are saved. In the case of k kernels, this can be done choosing an archive size of k. Thus the SA contains k n-dimensional solution vectors s_l and the corresponding k objective function values.[9]

As the focus is here on constrained MINLPs, the solution archive SA also contains the corresponding violation of the constraints and the penalty function value for every solution s_l. In particular, the attraction of a solution s_l saved in the archive is measured regarding the penalty function value instead of the objective function value. Details on the measurement of the violation and the penalty function will be described in Section 3.

We now explain the update process for the pheromones which is directly connected to the update process of the solution archive SA. The weights w (which indicate the importance of an ant and therefore rank them) are calculated with a linear proportion according to the parameter k:

$$w_l^i \ = \ \frac{(k-l+1)}{\sum_{j=1}^{k}\ j} \tag{5}$$

With this fixed distribution of the weights, a linear order of priority within the solution archive SA is established. Solutions s_l with a low index l are preferred. Hence, s_1 is the current best solution found, while s_k is the solution of the lowest interest, saved in SA. Updating the solution archive will then directly imply a pheromone update based on best found solutions. Every time that a new solution (ant) is created and evaluated within a generation, its attraction (penalty function value) is compared to the attraction of the solutions saved in SA, starting with the very best solution

s_1 and ending up with the last one s_k in the archive. In case that the new solution has a better attraction than the j-th one saved in the archive, the new solution will be saved on the j-th position in SA, and all solutions formerly taking the j-th till $k-1$-th position will drop down one index in the archive, discarding the solution formerly taking the last k-th position. As it is explained in detail in the following lines, the solutions saved in SA define the deviations and means used for the PDF. This way of updating the solution archive with better solutions leads automatically a positive pheromone update. Note that a negative pheromone update (evaporation) is indirectly performed as well by dropping the last solution s_k of SA every time that a new solution is introduced in it. Explicit pheromone evaporation rules are known[8] but were not considered here provided to the implicit negative update and for the sake of simplicity of the framework.

Standard deviations σ are calculated by exploiting the variety of solutions saved in SA. For every dimension i, the maximal and minimal distance between the single solution components s^i of the k solutions saved in SA is calculated. Then the distance between these two values, divided by the number of generations, defines the standard variation for every dimension i:

$$\begin{aligned} \sigma^i &= \frac{dis_{\max}(i) - dis_{\min}(i)}{\#\text{generation}} \\ dis_{\max}(i) &= \max\{|s_p^i - s_q^i| : \ p,q \ \in \ \{1,...,k\}, \ p \neq q\} \\ dis_{\min}(i) &= \min\{|s_p^i - s_q^i| : \ p,q \ \in \ \{1,...,k\}, \ p \neq q\} \end{aligned} \tag{6}$$

For all k single Gaussian functions within the PDF this deviation is then used regarding the corresponding dimension i. The means μ are given directly by the single components of the solutions saved in SA:

$$\mu_l^i = s_l^i \tag{7}$$

The incremental construction of a new ant works as follows: a mean μ_l^i is chosen first for every dimension i . This choice is done respectively to the weights w_l^i. According to the weights defined in (5) and the identity of μ_l^i and s_l^i defined in (7), the mean μ_1^i has the highest probability to be chosen, while μ_k^i has the lowest probability to be chosen. Second, a random number is generated by sampling around the selected mean μ_l^i using the deviation σ^i defined by (6). Proceeding in this way through all dimensions $i = 1, ..., n$, we create a new ant, which can be evaluated regarding its objective function value and constraint violations in a next step.

So far, this algorithm framework does not differ from the one proposed by Socha,[9] except that here explicit rules for the pheromone parameter w_l^i are given. The novel extension which enables the algorithm to handle mixed integer search domains modifies the deviations σ^i used for the integer variables and it is described in detail next.

To handle integer variables besides the continuous ones, as described above, a discretization of the continuous random numbers (sampled by the PDF) is necessary. The main advantage of this approach is the straightforward integration in the same framework described above. A disadvantage is the missing flexibility in cases where for an integer dimension i all k solution components $s^i_{1,...,k}$ in SA share the same value. In this case, the corresponding deviations $\sigma^i_{1,...,k}$ are zero regarding the formulation in (6). As a consequence, no further progress in these components is possible, as the PDF would only sample the exact mean without any deviation.

Introducing a lower bound for the deviation of integer variables helps to overcome this disadvantage and enables the ACO metaheuristic to handle integer and continuous variables simultaneously without any major extension in the same framework. For a dimension i that corresponds to an integer variable, the deviations σ^i are calculated by:

$$\sigma^i = \max\left\{\frac{dis_{\max}(i) - dis_{\min}(i)}{\#\text{generation}}, \ \frac{1}{\#\text{ generation}}, \ (1 - \frac{1}{\sqrt{n_{int}}})/2\right\} \tag{8}$$

With this definition, the deviations (according to the corresponding Gaussian functions for integer variables) will never fall under a fixed lower bound of $(1 - \frac{1}{\sqrt{n_{int}}})/2$, determined by the number

of integer variables n_{int} considered in the MINLP problem formulation. For MINLPs with a large amount of integers, this lower bound converges toward 0.5 and therefore ensures a deviation that hopefully keeps the algorithm flexible enough to find its way through the (large) mixed integer search domain. A small number of integer variables leads to a smaller lower bound, while for only one integer variable this bound is actually zero. In case of a small amount of integer variables in the MINLP, it is reasonable to think that the optimal combination of integers is found at some point of the search progress and therefore no further searching with a wide deviation is necessary for the integer variables. But even in the case of only one integer variable, with a corresponding absolute lower bound of zero, the middle term ($\frac{1}{\#\text{generation}}$) in (7) ensures a not too fast convergence of the deviation. Therefore, the calculation of the standard deviation σ for integer variables by (7) seems to be a reasonable choice and is confirmed by the numerical results.

3 Robust oracle penalty method

Penalty methods are well known strategies to handle constraints in optimization problems. Here we present a general penalty method based on only one parameter, named Ω. This parameter is selected best equivalent or just slightly greater than the optimal (feasible) objective function value for a given problem. As for most real-world problems this value is unknown a priori, the user has to guess this parameter Ω at first. After an optimization test run, the quality of the chosen parameter can be directly compared to the truly reached solution value. Due to this predictive nature of the parameter it is called an *oracle*.

In order to apply this methodology to real-world problem, in which the actual value of the global optimum is usually unknown, it is important that the method performs satisfactory even for bad or unreasonable choices of Ω. Indeed, the method is constructed this way and numerical results fortified its robustness. A detailed description of the development, robustness and performance of the method can be found online[6] or in Schlüter and Gerdts.[5] Therefore, only the essential mathematical formulation is given here.

The oracle method works with a norm-function over all violations of the constraints of an optimization problem; this function is called a residual. Commonly used norm-functions are the $L-1, L-2$ or $L-\infty$ norms. Here we assume the $L-1$ norm as residual (see (9)). To simplify the notation of the robust oracle method, we denote here with $z := (x, y)$ the vector of all decision variables without explicit respect to x and y, which are the vectors of continuous and integer variables in the MINLP formulation, respectively. Such a vector z is called an iterate from now due to the generality of the method. In case of ACO z represents an ant.

$$res(z) \;=\; \sum \; |h_i(z)| \quad - \sum \min\{0, g_j(z)\} \tag{9}$$

where h_i denote the equality constraints and g_j the inequality constraints. The penalty function $p(z)$ is then calculated by:

$$p(z) \;=\; \begin{cases} \alpha \cdot |f(z) - \Omega| + (1-\alpha) \cdot res(z) - \beta & , \text{ if } \quad f(z) > \Omega \;\text{ or } \quad res(z) > 0 \\ -|f(z) - \Omega| & , \text{ if } \quad f(z) \leq \Omega \;\text{ and } \; res(z) = 0 \end{cases} \tag{10}$$

where α is given by:

$$\alpha \;=\; \begin{cases} \frac{|f(z)-\Omega|\cdot\frac{6\sqrt{3}-2}{6\sqrt{3}}-res(z)}{|f(z)-\Omega|-res(z)} & \text{, if} \quad f(z)>\Omega \quad \text{and} \quad res(z)<\frac{|f(z)-\Omega|}{3} \\ 1-\frac{1}{2\sqrt{\frac{|f(z)-\Omega|}{res(z)}}} & \text{, if} \quad f(z)>\Omega \quad \text{and} \quad \frac{|f(z)-\Omega|}{3}\le\ res(z)\le|f(z)-\Omega| \\ \frac{1}{2}\sqrt{\frac{|f(z)-\Omega|}{res(z)}} & \text{, if} \quad f(z)>\Omega \quad \text{and} \quad res(z)>|f(z)-\Omega| \\ 0 & \text{, if} \quad f(z)\le\Omega \end{cases} \tag{11}$$

and β by:

$$\beta \;=\; \begin{cases} \frac{|f(z)-\Omega|\cdot\frac{6\sqrt{3}-2}{6\sqrt{3}}}{1+\frac{1}{\sqrt{\#\text{generation}}}}\cdot\left(1-\frac{3\ res(z)}{|f(z)-\Omega|}\right) & \text{, if} \quad f(z)>\Omega \quad \text{and} \quad res(z)<\frac{|f(z)-\Omega|}{3} \\ 0 & \text{, else} \end{cases} \tag{12}$$

Implementations of the oracle penalty method in Matlab, C/C++ and Fortran can freely be downloaded[6] together with some graphical illustrations and further readings.

4 The hybrid strategy - `ACOmi`

After explaining the background and extension of the ACO metaheuristic for general MINLP problems, this section provides detailed information on our implementation, named `ACOmi`, which combines this extended ACO metaheuristic with a deterministic local solver (MISQP[11]). In addition, several other heuristics are incorporated in `ACOmi`, that will appear in the presented pseudo-code algorithm scheme and illuminated within this section. Our algorithm was implemented in Matlab®.

It is to note that the fitness (or attraction) of ants in `ACOmi` is measured according to their objective function value in case of unconstrained problems. In case of constrained optimization problems the penalty function introduced in Section 3 is applied as fitness criterion.

4.1 Novel heuristics

4.1.1 Dynamic population heuristic

In our algorithm, the number of ants used within every iteration (or generation) does not remain constant during the optimization process, and it is calculated by a heuristic.

The calculation of the actual population size Pop_{size} used for a given generation ($\#iteration$) is shown in Algorithm 1, where n_{ants}, dyn_{max} and dyn_{mean} are parameters to be selected.

Here, n_{ants} is the minimum allowed population size of ants used within every generation. Thus, the population size will never be smaller than n_{ants}. The actual population size will increase and decrease in a linear matter over time and will reach its maximal size of dyn_{max} ants in the dyn_{mean}-th iteration. This is done to employ the ants in a more efficient way than it is done by a constant population size. It is assumed that during the first algorithm stage (namely till some iteration dyn_{mean}) the search process is most critical and the use of many ants is useful. Therefore, the population size will be increased up to this stage. After reaching the dyn_{mean}-th generation it will be immediately decreased about 50 percent and from there on it will be decreased in a linear

Algorithm 1 *Dynamic population heuristic*

if #iteration $\leq dyn_{mean}$ **then**
  $Pop_{size} = [\ n_{ants} + (dyn_{max} - n_{ants})\ \frac{\#iteration-1}{dyn_{mean}-1}\ ]$
else
  if #iteration $> dyn_{mean}$ **and** $\#iteration \leq 2 \cdot dyn_{mean}$ **then**
    $Pop_{size} = [\ dyn_{max} + (n_{ants} - dyn_{max})\ \frac{\#iteration}{2\ dyn_{mean}}\ ]$
  else
    $Pop_{size} = n_{ants}$
  end if
end if

matter till it reaches it minimum size of n_{ants}. The immediate reduction after the dyn_{mean}-th generation is done to save a significant amount of function evaluations, as the first stage is seen much more critical than the following ones. The population size is then decreased in a linear matter and not dropped to n_{ants} immediately to assure a smoother change.

Figure 1 illustrates the population size Pop_{size} during 150 iterations for parameters $n_{ants} = 100$, $dyn_{max} = 500$ and $dyn_{mean} = 50$. Regarding Algorithm 1, the population size will increase and decrease in an asymmetric way with respect to the dyn_{mean}-th iteration. The algorithm stage between the first iteration and the dyn_{mean}-th iteration is seen as the most important and therefore more ants are employed during this stage. It must be noted that both, the calculation of the population size and the parameter selection are of heuristic nature.

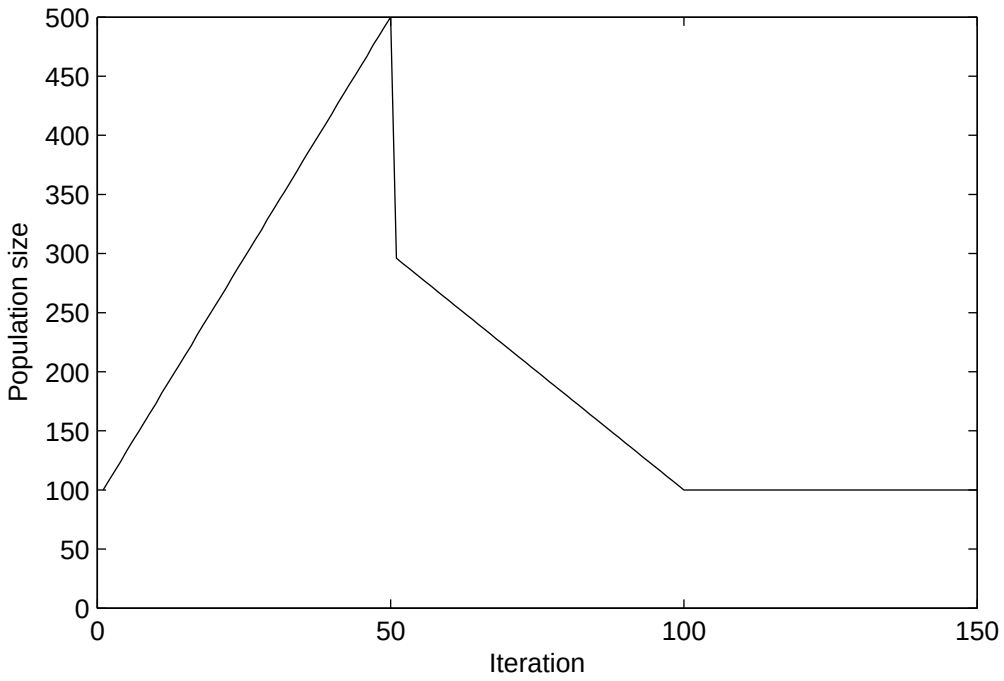

Figure 1: Population size during 150 iterations.

4.1.2 Single dimension tuning heuristic (SDT)

This heuristic aims to improve the current best solution s_1 saved in the solution archive SA by only optimizing single variables out of its n components. Obviously this approach can easily lead to small improvements for high-dimensional problems. The optimization of a single dimension of the solution is done by randomly sampling around the current best known component of s_1

with a proper deviation. In case of mixed integer optimization problems this heuristic is particular interesting, because the algorithm can gain an additional flexibility in handling the integer variables.

First, the treatment of dimensions that corresponds to continuous variables is explained. A new ant x^i_{sdt} will be created for every dimension $i \leq n_{cont}$ by:

$$x^i_{sdt} \;=\; s^i_1 \;+\; \frac{(x^i_u - x^i_l) \cdot x^i_{rand}}{\#\text{iteration}} \tag{13}$$

where x_{rand} is a vector of dimension n whose component are uniform random number within the interval $[0, 1]$. The variables range divided by the number of iterations performed so far, acts here as a proper deviation. The integer variables are handled in a different way. According to a another sampled uniform random number between zero and one, the current integer value is decreased or increased by only one unit:

$$x^i_{sdt} \;=\; s^i_1 \;+\; \begin{cases} 1, & \text{if } x^i_{rand} \geq \frac{1}{2} \\ -1, & \text{if } x^i_{rand} < \frac{1}{2} \end{cases} \tag{14}$$

In case that the current component s^i_1 has reached its lower or upper bound and the x^i_{sdt} component violates it, no change should be done for this component.

This procedure is performed in every iteration for every single dimension i, which leads to n additional function evaluations per iteration. For MINLPs with a large number of variables, this leads to a large amount of extra function evaluations. The experienced improvement of the algorithm's performance in a wide range of problems based on this heuristic, justifies this additional computational cost.

4.1.3 Weighted average best ant heuristic (WABA)

It must be pointed out that this heuristic aims to create an improved solution by using all the information captured so far in the solution archive SA. Out of the k solutions s_l saved in SA, a weighted average ant x_{waba} is created according to the previous described weights w:

$$x^i_{waba} \;=\; \sum_{l=1}^{k} w^i_l \cdot s^i_l, \tag{15}$$

where i refers to the dimension of the optimization variable. In case i refers to an integer variable, the value of x^i_{waba} is rounded to the next integer. This heuristic is very cheap in terms of computational cost. No sampling of random numbers is required and only one additional function evaluation has to be performed in every iteration.

4.1.4 Final stage heuristic

Our implementation `ACOmi` is able to detect a final stage of the algorithm progress by monitoring the decline of the fitness of ants. This is measured by the objective function value in case of unconstrained problems and by the penalty function (Section 3) in case of constrained ones. During the whole progress, the difference between the fitness values of the so far best found ant in two consecutive iterations is computed. The maximum and average value of these differences are called, respectively, dc_{max} and $dc_{average}$. The comparison of both values provides a final stage criterion. In every iteration the progress of the algorithm is checked according to the relation between these two values. This is done by calculating the following binary flag $stage_{final} \in \{0, 1\}$, where a flag $stage_{final} = 1$ indicates the transition to final stage.

$$stage_{final} \;=\; \begin{cases} 1, & \text{if} \quad dc_{average} < dc_{max}/W_{final} \qquad (W_{final} \in \mathbb{N}^+) \\ 0, & \text{else} \end{cases} \tag{16}$$

In (16) the parameter W_{final} acts as a weight to compare $dc_{average}$ with dc_{max}. In case that $dc_{average}$ has become smaller than the weighted dc_{max}, no further extensive progress is expected. Consequently the final stage flag $stage_{final}$ indicates this by becoming active.

If a final stage is detected by `ACOmi`, it will perform some local search actions. As `ACOmi` is a hybrid implementation, it has the option to call a deterministic local solver. The local solver is then called with the current best solution s_1 saved in SA as initial point. In case the local solver is not able to provide a feasible solution with an objective function value lower or equal to fex (which is a desired (feasbile) objective function value, see Section 4.2), a restart with a population concentrated around the current best solution s_1 is performed. Therefore the pheromones are initialized again with the means identical to the components of s_1 and a very small deviation around them. The complete solution archive SA will be cleared to avoid getting stuck again in the same solution s_1, even so the information of the best solution so far is represented in the pheromones. The hope is to improve the solution by sampling very closely around the current best without keeping it as a reference in the SA anymore and therefore biasing the solution hierarchy in SA. During this second ACO based search process, the local solver will be called frequently. This means that, for a given frequency $freq \in \mathbb{N}^+$, the local solver is called after every $freq$-th iteration.

4.2 Implementation scheme of `ACOmi`

The implementation is divided into two significant stages: `ACOmain` and `ACOfinal`. The algorithm starts by initializing pheromones randomly. Then, the `ACOmain` stage performs a pure ACO search process as described in Section (2). It must be noted that the number of ants per generation n_{ants} is not fixed throughout the search process. The number of ants is calculated in every iteration according to a heuristic (see Section 4.1.1). Within the `ACOmain` stage two additional heuristics (SDT and WABA) are executed in every generation to improve the current best solution found by the ants. At the end of every `ACOmain` generation, a stopping criterion is calculated by a binary flag variable, $stage_{final}$. This flag indicates the algorithm to switch to the `ACOfinal` stage. The steps followed in this stage are explained in Section 4.1.4. A pseudocode of our implementation is shown in Algorithm 2.

Three conditions act here as a stop criteria: (i) a maximal limit for function evaluations, (ii) a maximal budget for computation time, and (iii) the achievement of a feasible solution with a corresponding objective function value smaller or equal to the user-given value of fex. Those criteria are checked after every iteration.

According to the parameter selection rules of thumb are given now. Note that those rules are based solely on experience with `ACOmi`. A reasonable kernel size k has been found between 5 and 50. The minimum number of ants per generation n_{ants}, should be around two to ten times of the kernel size k. The maximal amount of ants per generation dyn_{max} should be around two to four times of n_{ants}. The generation of maximal ants dyn_{mean} should be about a half to four times of the kernel size k. As a reasonable weight W_{final} for the final stage $stage_{final}$ heuristic the value 100 was observed.

5 Integrated design and control case studies

In this section we consider a set of different integrated design and control problems of full industrial plant of medium complexity. This set comprises both NLP and MINLP problems. To check their practical multimodality, a multistart procedure using sequential quadratic programming (SQP) optimization methods was applied to each of the problems considered. For a further comparison of our `ACOmi` results, three additional solvers have been tested: MITS,[12] SSm[13] and CMAES.[14] For mixed integer problems, CMAES was not applied, because it is designed for continuous problems only. All numerical results were calculated on a personal computer with a 3.2 GHz clock rate and 1 GB RAM working memory.

Algorithm 2 `ACOmi`

Specify stopping criterion: Maximal evaluation budget $eval_{max}$
Maximal time budget $time_{max}$
Objective function value fex to be reached
Optional preliminary stage: Multistart of local solver with random initial points
Initialization: Choose pheromones (randomly)
Set $stage_{final} = 0$
Set $ACO_{final} = 0$
while stop criteria not met **do**
  if $stage_{final} = 0$ **then**
    Construct dynamic population of N_{ants} ants
    Evaluate fitness of ants
    Save k best ants in SA
    Improve best ant with SDT heuristic and update SA
    Calculate WABA and update SA
    Update pheromones according SA
    if $ACO_{final} = 0$ **then**
      Calculate final stage flag $stage_{final}$
      if $stage_{final} = 1$ **then**
        $ACO_{final} = 1$
      end if
    else
      Run local solver (frequently)
    end if
  else
    Run local solver with s_1 as initial point
    $stage_{final} = 0$
    if stop criteria not met **then**
      Choose pheromones (according to s_1)
      Clear SA
    end if
  end if
end while

5.1 WWTP COST model

5.1.1 Introduction

In order to enhance the development and acceptance of new control strategies, the International Water Association (IWA) Task Group on Respirometry, together with the European COST work group, proposed a standard simulation benchmarking methodology for evaluating the performance of activated sludge plants. The COST 624 work group defines the benchmark as *a protocol to obtain a measure of performance of control strategies for activated sludge plants based on numerical, realistic simulations of the controlled plant.* According to this definition, the benchmark consists of a description of the plant layout, a simulation model and definitions of (controller) performance criteria. The layout of this benchmark plant combines nitrification with predenitrification by a five compartment reactor with an anoxic zone (see Figure 2). A secondary settler composed by 10 layers separates the microbial culture from the liquid being treated. A basic control strategy consisting of 2 PI controllers is proposed to test the benchmark. Its aim is to control the dissolved oxygen level in the final compartment of the reactor (AS Unit 5) by manipulation of the oxygen transfer, and to control the nitrate level in the last anoxic compartment (AS Unit 2) by manipulating the internal recycle flow rate.[16]

In this work, a Simulink® implementation of the benchmark implemented by Dr. Ulf Jeppsson was used for the simulations (for more information about the implementation, see Alex et al.,[17] Copp[18]

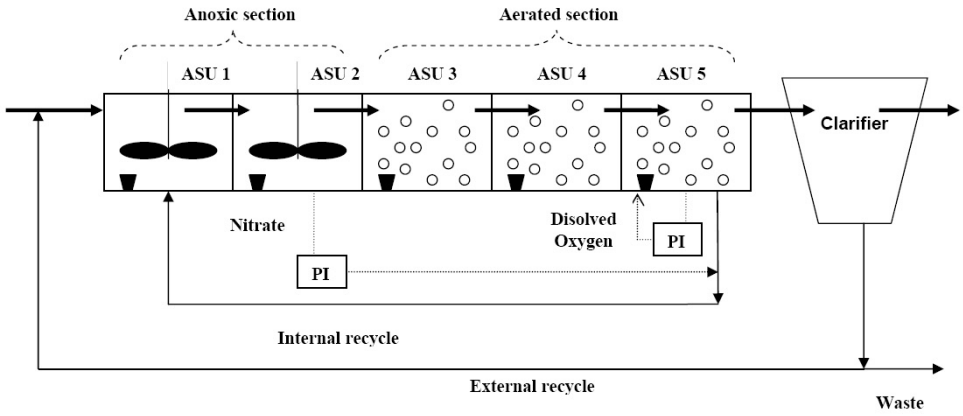

Figure 2: WWT.COST plant scheme

or Jeppsson and Pons[19]). Each function evaluation consists of an initialization period of 100 days to achieve steady state, followed by a period of 14 days of dry weather and a third period of 14 days of rainy weather. Calculations of the controller performance criterion are based on data from the last 7 rain days. A study comparing different global optimization methods, including surrogate model-based methods for computationally expensive models, over this model, was recently carried out by Egea et al.[20]

The system dynamics are described by algebraic mass balance equations, ordinary differential equations for the biological processes in the bioreactors as defined by the ASM1-model,[21] and the double-exponential settling velocity function[22] as a fair representation of the settling process. The overall process is formed by 8 sub-processes and it is described by a set of more than 100 DAE's with 13 state variables. For the sake of brevity, the detailed model of the full plant as well as the parameters and design variables values are not shown in this work, but they can be found in the *IWA Task Group on Benchmarking of Control Strategies for WWTPs* web page*.

Provided the physical design and the control strategy of the plant, there is a number of operating variables over which we can apply optimization techniques to minimize a performance index of the plant. In this work we have considered the ones listed in Table 1. Default values are proposed by the benchmark authors.

Table 1: Operational variables for the WWT COST benchmark

Variable	Description	Symbol	Default value	Units
v_1	Proportional gain O_2 controller	$K_{(O)}$	500	$d^{-1}(g(-COD)m^{-3})$
v_2	Integral time O_2 controller	$\tau_{i(O)}$	0.001	d
v_3	Antiwindup constant O_2 controller	$\tau_{t(O)}$	0.0002	d
v_4	Proportional gain N controller	$K_{(N)}$	15000	$m^3d^{-1}(gNm^{-3})^{-1}$
v_5	Integral time N controller	$\tau_{i(N)}$	0.05	d
v_6	Antiwindup constant N controller	$\tau_{t(N)}$	0.03	d
v_7	Aeration factor ASU1	KLa_1	0	d^{-1}
v_8	Aeration factor ASU2	KLa_2	0	d^{-1}
v_9	Aeration factor ASU3	KLa_3	240	d^{-1}
v_{10}	Aeration factor ASU4	KLa_4	240	d^{-1}
v_{11}	External recycle flow rate	Q_r	18446	m^3d^{-1}
v_{12}	Purge flow rate	Q_w	385	m^3d^{-1}
v_{13}	Settler input layer (integer variable)	L_{feed}	5	-

*http://www.ensic.inpl-nancy.fr/benchmarkWWTP/Bsm1/Benchmark1.htm

For the optimization of this model, convergence curves will be plotted with respect to the number of simulations instead of the computation time. The reason is that, for some simulations, the numerical integration fails producing an algebraic loop which can spend severals hours of computation time. Besides, the overhead introduced by every optimization method can be considered negligible compared to the time needed for each simulation.

5.1.2 WWT.COST.1

In a first approach, we will try to optimize the control performance of the plant, tested by using the *ISE* (Integral Square Error). Both the nitrate level and oxygen level controllers will be optimized with respect to their controller parameters, that is, the gain K (i.e., v_1 and v_4) and integral time constant τ_i (i.e., v_2 and v_5). The problem is formulated as follows:

$$\min \quad J(v) = c \cdot W^T \cdot ISE \tag{17}$$

subject to the system dynamics. $W^T \in \mathbb{R}^{1\times 2}$ contains the weighting coefficients and $ISE \in \mathbb{R}^{2\times 1}$ contains the integral squared errors of the two PI controllers. The weighting vector W^T, the integral square error, *ISE*, and the decision variables vector are as follows:

$$W^T = [w_1 \quad w_2] = \left[\frac{1000}{1001} \quad \frac{1}{1001}\right] \tag{18}$$

$$ISE = \begin{bmatrix} ISE_O \\ ISE_N \end{bmatrix} \tag{19}$$

$$ISE_{(.)} = \int_{t_0}^{t_f} \varepsilon\,(\tau)^2_{(.)}\, d\tau \tag{20}$$

$$v = \begin{bmatrix} v_O \\ v_N \end{bmatrix} = \begin{bmatrix} K_{(O)} \\ \tau_{i(O)} \\ K_{(N)} \\ \tau_{i(N)} \end{bmatrix} \tag{21}$$

The weighting vector is chosen such that the ISE_O equals to the ISE_N part when using the benchmark default settings provided by the COST project.[18] Boundaries on the decision variables (v^L and v^U) are chosen such that the process dynamics would not show (exceptional) unstable behavior:

$$v^L = [100 \quad 7.0 \cdot 10^{-4} \quad 100 \quad 1.0 \cdot 10^{-2}]^T \tag{22}$$

$$v^U = [1000 \quad 7.0 \cdot 10^{-1} \quad 50000 \quad 1.0]^T \tag{23}$$

The objective function values are normalized with respect to the performance obtained with the tuned controller settings provided by the COST project (i.e., default value of Table 1) using the constant parameter $c = 1.1845 \cdot 10^3$ to obtain a function value equal to one when using default values for the decision variables.

The histogram (in log-scale) depicting the multistart procedure to check the non-convexity of the problem is shown in Figure 3. Due to the high computational cost of every simulation, the number of initial points used in the multistart procedure was only 40.

The histogram shows the practical non-convexity of the problem and that the best value reported ($f(x) = 0.7463$) outperforms the value obtained with default parameters but not the solutions obtained applying global optimization methods, as shown below.

Table 2 lists the parameters of `ACOmi` used for all test runs on this application together with a brief explanation and reference information. In Table 3 the best (f_{best}), worst (f_{worst}) and mean

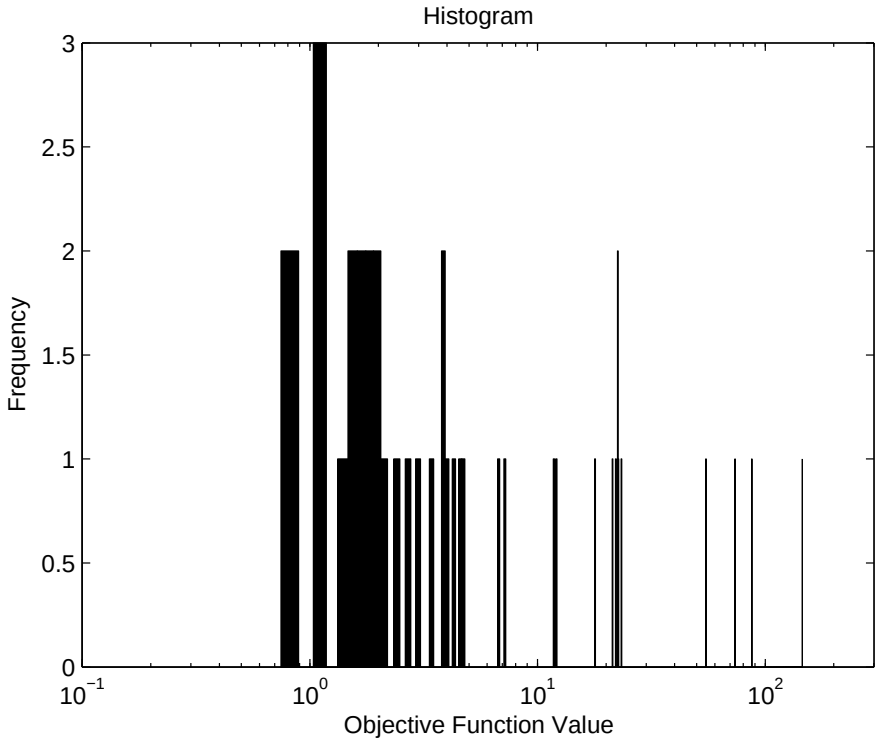

Figure 3: Histogram of solutions obtained from the multistart procedure for WWT.COST.1

(f_{mean}) objective function values, obtained by each solver throughout 10 test runs, are reported. In addition the mean number of function evaluations ($\text{eval}_{\text{mean}}$) and the corresponding cpu-time ($\text{time}_{\text{mean}}$) in seconds are also given. We assigned a maximal budget of 400 simulations (function evaluations) as a limit for every test run, but we did not declare a maximal time budget. All test runs started from the same initial point, which can be found together with the lower and upper bounds in Table 10 (see Appendix). Table 10 also shows the optimal vector (x^*) obtained by all tested solvers.

Table 2: `ACOmi` parameter setup for WWT.COST.1

parameter	value	explanation	reference
k	5	number of kernels = size of SA	Section 2
n_{ants}	10	number of ants	Section 4.1.1
dyn_{max}	20	maximal number of ants	Section 4.1.1
dyn_{mean}	5	number of iteration for dyn_{max}	Section 4.1.1
W_{final}	100	weight for $stage_{final}$	Section 4.1.4
$freq$	3	frequency for local solver	Section 4.1.4

Table 3: Results for the WWT.COST.1

solver	f_{best}	f_{worst}	f_{mean}	$\text{eval}_{\text{mean}}$	$\text{time}_{\text{mean}}$
MITS	0.7433	2.6579	1.3798	421	36979
`ACOmi`	0.5273	0.7568	0.5528	426	32352
CMAES	0.5313	1.8751	0.7690	402	27926
SSm	0.5354	0.8013	0.6309	415	33240

Analyzing the results on this application `ACOmi` was able to obtain the best overall objective function value and also the best mean objective function value. While SSm performed slightly better than CMAES, MITS was clearly the worst solver on this application.

To illustrate the convergence rate of the tested solvers, Figure 4 displays (in log-log-scale) the convergence curves corresponding to the best run for each solver.

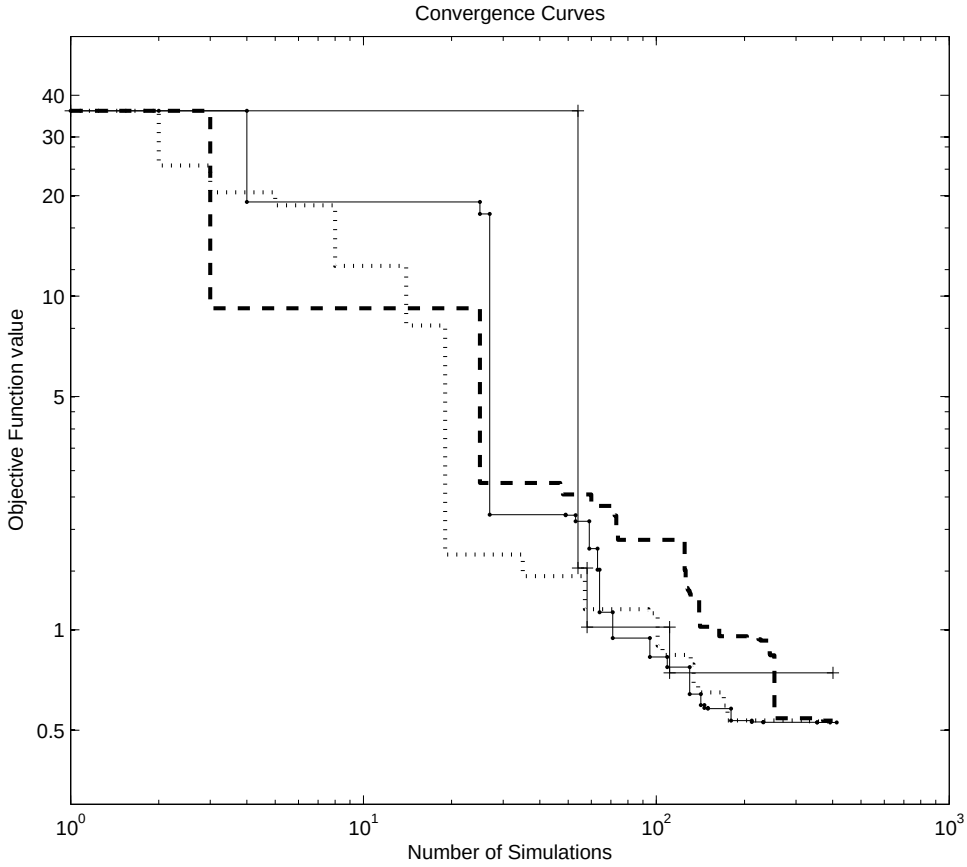

Figure 4: Convergence curves of best runs of MITS, SSM, CMAES and `ACOmi` for WWT.COST.1

5.1.3 WWT.COST.2

After having tested the different optimization algorithms over the COST benchmark model, the next step is to pose a more complicated problem in terms of design and therefore in terms of number of decision variables. The new formulated objective function will be more complex and will take into account not only controllability aspects but also the process economy. The selected decision variables for this extended problem will be, appart from the controllers parameters chosen in the previous section, the aeration factors of the aerated tanks (i.e., KLa_3 and KLa_4), the external recycling flow rate, Q_r, and the sludge purge flow rate, Q_w. The new optimization problem is formulated as follows:

$$\min \quad C(v) = w_1 \cdot \phi_{cont} + w_2 \cdot \phi_{econ} \tag{24}$$

where ϕ_{cont} is the same term defined in Equation 17. ϕ_{econ} takes into account the different terms which define the operating costs of the process, such as effluent quality, EQ, aeration and pumping energies, AE and PE, and the amount of sludge for disposal, P_{sludge}. Vanrolleghem and Gillot[23] defined particular economic costs derived from each of these indexes. Based on the relations among these costs, we have defined ϕ_{econ} as:

$$\phi_{econ} = 2 \cdot EQ + AE + PE + 3 \cdot P_{sludge} \tag{25}$$

w_1 and w_2 are chosen for both terms, ϕ_{cont} and ϕ_{econ}, to be in the same order of magnitude when using the default values for the decision variables. As in the previous section, the optimization problem is subject to the system dynamics and the bounds for the decision variables. For the controllers parameters we use the same bounds as in the previous case (i.e., equations 22 and 23). Upper bounds for these new considered decision variables were chosen taking into account the recommendations of the benchmark authors, whereas the lower bounds were chosen to avoid systematic numerical integration errors along the optimizations. These bounds, together with the initial point used for the optimizations are shown in Table 11 (see Appendix).

The histogram (in log-scale) depicting the multistart procedure to check the non-convexity of the problem is shown in Figure 5. The number of initial points used was also 40 for the same reasons given in the last section.

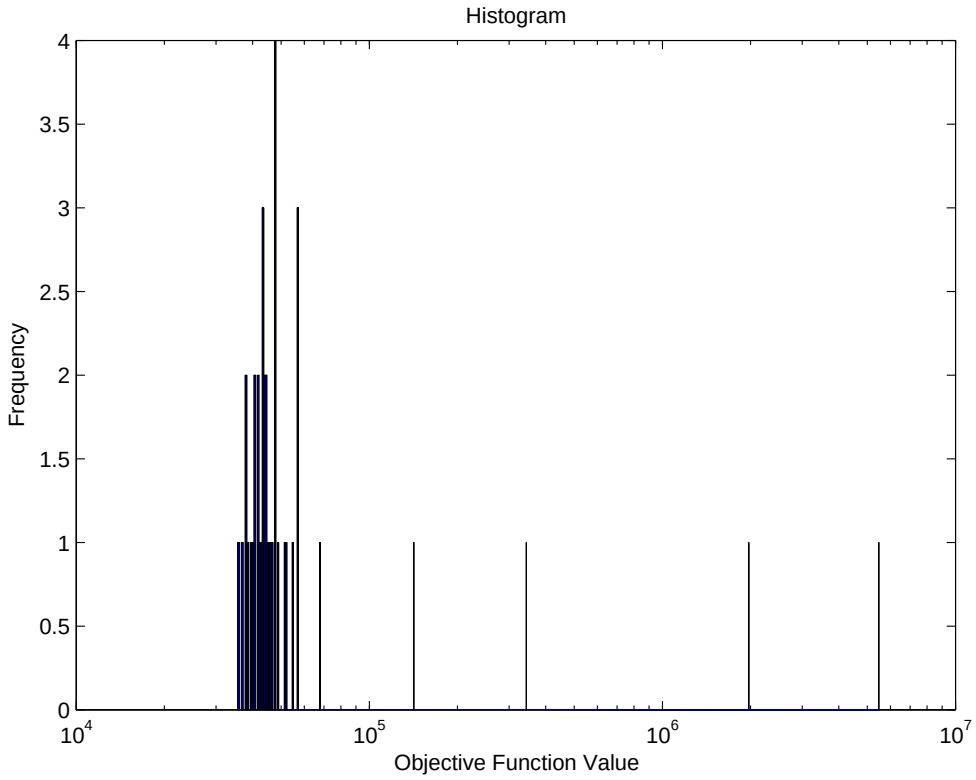

Figure 5: Histogram of solutions obtained from the multistart procedure for WWT.COST.2

The histogram shows the practical non-convexity of the problem and the best value reported by the multistart ($f(x) = 35521$) does not improve the value obtained using default values for the decision variables ($f(x) = 35225$).

Table 4 lists the parameters of `ACOmi` used for all test runs on this application together with a brief explanation and reference information. In Table 5 the best (f_{best}), worst (f_{worst}) and mean (f_{mean}) objective function values, obtained by each solver throughout 10 test runs, are reported. In addition the mean number of function evaluations ($\text{eval}_{\text{mean}}$) and the corresponding cpu-time ($\text{time}_{\text{mean}}$) in seconds are also given. We assigned a maximal budget of 800 simulations (function evaluations) as a limit for every test run, but we did not declare a maximal time budget. All test runs started from the same initial point, which can be found together with the lower and upper bounds in Table 11. Table 11 also shows the optimal vector (x^*) obtained by all tested solvers.

Table 4: `ACOmi` parameter setup for WWT.COST.2

parameter	value	explanation	reference
k	30	number of kernels = size of SA	Section 2
n_{ants}	100	number of ants	Section 4.1.1
dyn_{max}	300	maximal number of ants	Section 4.1.1
dyn_{mean}	50	number of iteration for dyn_{max}	Section 4.1.1
W_{final}	100	weight for $stage_{final}$	Section 4.1.4
$freq$	3	frequency for local solver	Section 4.1.4

Table 5: Results for WWT.COST.2

solver	f_{best}	f_{worst}	f_{mean}	$\text{eval}_{\text{mean}}$	$\text{time}_{\text{mean}}$
MITS	34167	39411	36596	893	95877
`ACOmi`	33678	35228	34994	970	257610
CMAES	34852	37561	35531	802	88369
SSm	33993	36652	34794	853	103750

Analyzing the results on this application SSm and `ACOmi` obtained the best solutions amongst the four tested solvers. While the best `ACOmi` of 33678 was slightly better than the one achieved by SSm of 33993, the mean objective function value of SSm was slightly better than those of `ACOmi`.

To illustrate the convergence rate of the tested solvers, Figure 6 displays (in log-log-scale) the convergence curves corresponding to the best run for each solver.

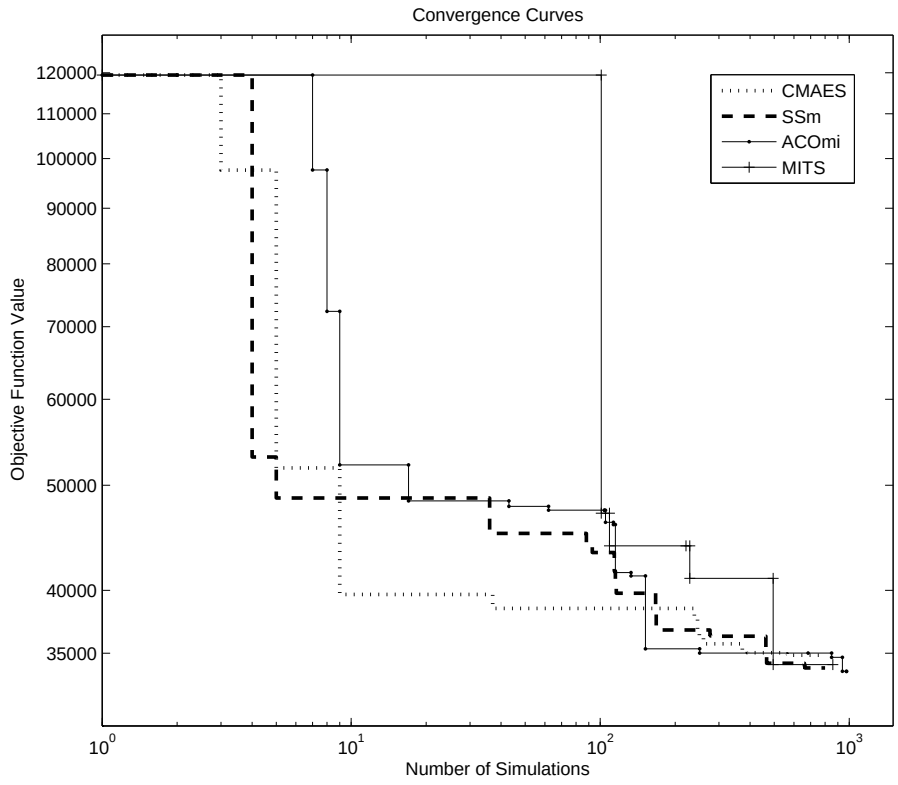

Figure 6: Convergence curves of best runs of MITS, SSM, CMAES and `ACOmi` for WWT.COST.2

5.1.4 WWT.COST.3

In addition to the above NLP formulation of the WWT COST benchmark operational design problem, an extension of the previous problem is proposed here. In this case, all the variables shown in Table 1 will be used as decision variables. Some of the bounds are extended to allow a possible change of the plant configuration (e.g., the anoxic tanks could became aerated and vice-versa, changing from a pre-denitrification plant to a post-denitrification plant). The objective function used will be the same as in the previous problem. For the sake of comparison with a previous work by Exler et al.,[12] the initial point used for this case will be the default one provided by the benchmark authors, and shown in Table 1.

The histogram (in log-scale) depicting the multistart procedure to check the non-convexity of the problem is shown in Figure 7. The number of initial points used was also 40 and the local solver used for this case was MISQP.

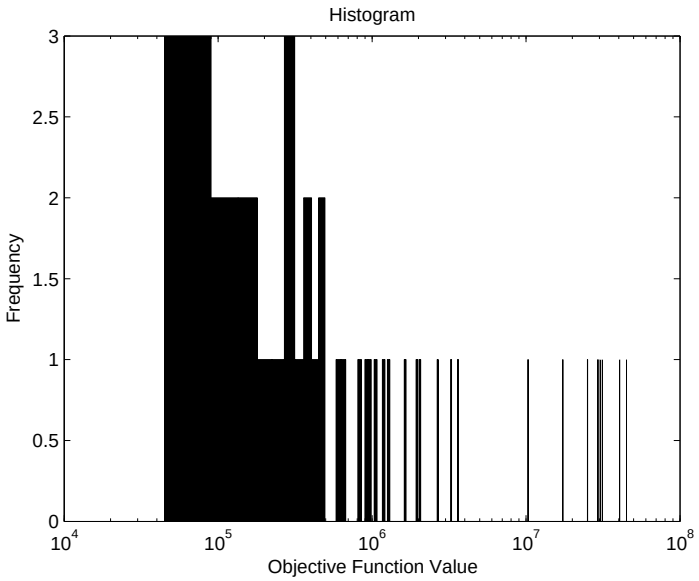

Figure 7: Histogram of solutions obtained from the multistart procedure for WWT.COST.3

The histogram shows the practical non-convexity of the problem and the best value reported by the multistart ($f(x) = 44937$) is very far from the value obtained using default values for the decision variables ($f(x) = 35225$).

Table 6 lists the parameters of `ACOmi` used for all test runs on this application together with a brief explanation and reference information. In Table 7 the best (f_{best}), worst (f_{worst}) and mean (f_{mean}) objective function values, obtained by each solver throughout 10 test runs, are reported. In addition the mean number of function evaluations ($\text{eval}_{\text{mean}}$) and the corresponding cpu-time ($\text{time}_{\text{mean}}$) in seconds are also given. We assigned a maximal budget of 1500 simulations (function evaluations) as a limit for every test run, but we did not declare a maximal time budget. All test runs started from the same initial point, which can be found together with the lower and upper bounds in Table 12 (see Appendix). Table 12 also shows the optimal vector (x^*) obtained by all tested solvers.

Analyzing the results on this application `ACOmi` outperformed the other solvers in terms of the best, worst and mean objective function value. In particular the worst `ACOmi` solution is still better than the best ones of MITS and SSm. MITS and SSm achieved a similar mean objective function value while the best SSm result is significantly better than the one of MITS.

To illustrate the convergence rate of the tested solvers, Figure 8 displays (in log-log-scale) the convergence curves corresponding to the best run for each solver.

Table 6: `ACOmi` parameter setup for WWT.COST.3

parameter	value	explanation	reference
k	5	number of kernels = size of SA	Section 2
n_{ants}	15	number of ants	Section 4.1.1
dyn_{max}	50	maximal number of ants	Section 4.1.1
dyn_{mean}	15	number of iteration for dyn_{max}	Section 4.1.1
W_{final}	100	weight for $stage_{final}$	Section 4.1.4
$freq$	0	frequency for local solver	Section 4.1.4

Table 7: Results for WWT.COST.3

solver	f_{best}	f_{worst}	f_{mean}	$\text{eval}_{\text{mean}}$	$\text{time}_{\text{mean}}$
MITS	34296	34296	34296	1828	71580
`ACOmi`	32544	32874	32694	1512	161460
SSm	33104	35226	34471	1500	160643

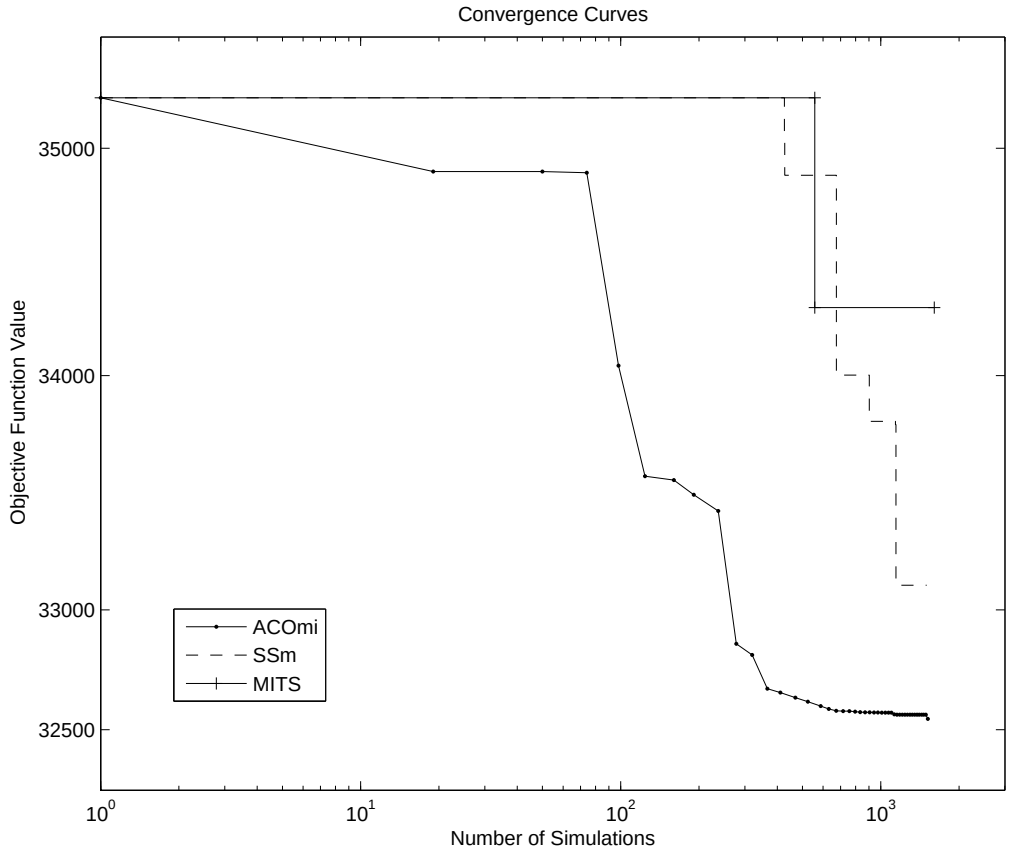

Figure 8: Convergence curves of best runs of MITS, SSm and `ACOmi` for WWT.COST.3

5.2 Tennessee Eastman Process

Since the publication of the Tennessee Eastman process (TEP) example by Downs and Vogel,[24] it has been widely used in the literature as a benchmark due to its challenging properties from a control engineering point of view: it is highly nonlinear, open-loop unstable and it presents a large number of measured and manipulated variables which offer a wide set of candidates for possible

control strategies. The flow sheet for the TEP is depicted in Figure 9. Two products (G and H) are produced from four reactants (A, C, D and E). A further inert trace component (B) and one byproduct (F) are present. The process units consist of a continuous stirred tank reactor, a condenser, a flash drum, a compressor and a stripper. The gaseous reactants are fed to the reactor where they are transformed into liquid products. The following reactions take place in gas phase:

$$\begin{aligned} A_{(g)} + C_{(g)} + D_{(g)} &\rightarrow G_{(l)} \\ A_{(g)} + C_{(g)} + E_{(g)} &\rightarrow H_{(l)} \\ A_{(g)} + E_{(g)} &\rightarrow F_{(l)} \\ 3D_{(g)} &\rightarrow 2F_{(l)} \end{aligned} \tag{26}$$

These reactions are irreversible and exothermic with rates that depend on temperature through Arrhenius expressions and on the reactor gas phase concentration of the reactants. The reaction heat is removed from the reactor by a cooling bundle. The products and the unreacted feeds pass through a cooler and, once condensed, they enter a vapor-liquid separator. The non condensed components recycle back to the reactor feed and the condensed ones go to a product stripper in order to remove the remaining reactants by stripping with feed stream. Products G and H are obtained in bottoms. The inert (B) and the byproduct (F) are mainly purged from the system as a vapor from the vapor-liquid separator.

Recently, Antelo et al.[25] applied their systematic approach to a plant-wide control design developed in a previous work[26] to derive robust decentralized controllers for the Tennessee Eastman Process. In this framework, the TEP is represented as a process network. Then, conceptual mass and energy inventory control loops for each node are designed first to guarantee that the states of the plant will remain on a convex invariant region, where the system will be passive and therefore input output stability can be stated.[26] The next step is to realize the proposed conceptual inventory control loops using the physical inputs-outputs of the process. Some extra control loops are needed to achieve the convergence of the intensive variables since the inventory control by itself does not ensure the convergence of these variables to a desired operation point. In some cases, the available degrees of freedom are not enough to implement the complete control structure that ensures both extensive and intensive variables convergence to the reference values. As a consequence, the setpoints of the inventory controllers can be used as new manipulated variables to complete the decentralized control design.

We explain the alternatives we introduced to extend the original hierarchical control design proposed by Antelo et al.[25] Concerning the reactor level control loop in the original design, its set point modifies the reference for the flow controller acting over E feed. As an alternative for closing this loop, both D as well as $A+C$ feeds are proposed as alternative manipulated variables. The other feed present in the TEP (A Feed) is not considered to close this loop since one of the disturbances defined by Downs and Vogel[24] is a total loose of this stream (IDV6).

For the reactor pressure case, the original proposal by Antelo et al.[25] considers the condenser cooling water flow as the manipulated variable. By using this, the reactor pressure can be varied since the separator pressure and, as a consequence, the recycle rate can be modified. It is at this point where the control over the vapor mass inventory in the separator by using the purge rate is established in order to ensure that all inventories in the TEP will remain bounded and, therefore, input-output stability is guaranteed. As alternatives,we are considering here:

1. The **purge flow**, which is a manipulated variable widely used in the literature for the reactor pressure control loop. By modifying this, it is possible to regulate the separator pressure as well as the recycle flow, and therefore the reactor pressure. When this alternative is considered, an extra loop controlling the separator temperature (energy inventory) by acting over the condenser coolant flow is defined.
2. The **recycled flow**, which can be modified by manipulating the compressor recycle valve. As a result, the gaseous reactor feed will vary and, therefore, also the reactor pressure.
3. The **A+C Feed**, which is the largest gaseous feed in the TEP and, as a consequence, it can be a candidate to regulate the pressure in the reactor, as presented in the work by Luyben et al.[27]

In order to determine the best control alternative among the proposed ones, a new binary vector b is added to our system dynamics. These 0-1 variables express which of the control strategies is being used, and they are defined as follows:

$$\begin{array}{ll} x(37) \equiv b_1 \in \{0,1\} & (Condenser\ Cooling\ Flow) \\ x(38) \equiv b_2 \in \{0,1\} & (Purge\ Rate) \\ x(39) \equiv b_3 \in \{0,1\} & (E\ Feed) \\ x(40) \equiv b_4 \in \{0,1\} & (D\ Feed) \\ x(41) \equiv b_5 \in \{0,1\} & (A+C\ Feed\ for\ pressure\ loop) \\ x(42) \equiv b_6 \in \{0,1\} & (A+C\ Feed\ for\ level\ loop) \\ x(43) \equiv b_7 \in \{0,1\} & (Recycle\ Rate) \end{array} \tag{27}$$

Therefore, the original control design proposal by Antelo et al.[25] will be characterized by the vector $b = (1,0,1,0,0,0,0)^T$ since it uses E Feed to control the reactor level and the condenser coolant flow to control the reactor pressure. From all the exposed, the optimization problem consists now on solving the following MINLP of the form:

$$\begin{array}{l} \min\limits_{\mathbf{v},\mathbf{b}} J(\mathbf{z},\mathbf{v},\mathbf{b}) \\ \\ s.a. \\ \\ \mathbf{f}(\dot{\mathbf{z}},\mathbf{z},\mathbf{p},\mathbf{v},\mathbf{b},t) = \mathbf{0} \\ \mathbf{h}(\mathbf{z},\mathbf{p},\mathbf{v},\mathbf{b}) = \mathbf{0} \\ \mathbf{g}(\mathbf{z},\mathbf{p},\mathbf{v},\mathbf{b}) \geq \mathbf{0} \\ b_1 + b_2 + b_5 + b_7 = 1 \\ b_3 + b_4 + b_6 = 1 \\ 1 - b_5 - b_6 \geq 0 \\ \mathbf{v^l} \leq \mathbf{v} \leq \mathbf{v^u} \\ \mathbf{b^l} \leq \mathbf{b} \leq \mathbf{b^u} \end{array} \tag{28}$$

where $\mathbf{b} \in \{0,1\}^7$ is the vector of binary variables (0-1 variables) and $\mathbf{v} \in \mathbb{R}^{36}$ are the continuous variables (the controller parameters). The lower and upper bounds for the binary variables will be of the form $\mathbf{b^l} = (0,...,0)^T$ and $\mathbf{b^u} = (1,...,1)^T$. It must be pointed out that we are considering that only one of the alternatives for each loop can be active at one time, being necessary to introduce the additional linear constraints $b_1 + b_2 + b_5 + b_7 = 1$ and $b_3 + b_4 + b_6 = 1$. The linear constraint $1 - b_5 + b_6 \geq 0$ ensures that only one (or no one) of the alternatives b_5 or b_6 is active at the same time. The rest of the decision variables are connected to the tuning of the PI controllers.

Note that the MINLP is also subject to the dynamics (DAEs) of the system which are expressed by f in Eq. (28). The TEP has 171 DAEs (30 ODEs and 141 algebraic equations). The MINLP is also made up of the following constraints which are related with the reactor pressure, temperature and volume, and with the separator and the stripper volumes:

$$\begin{array}{c} P_r \leq 3000\ KPa \\ \\ 2\ m^3 \leq V_{L_r} \leq 24\ m^3 \\ \\ T_r \leq 175\ ^\circ C \\ \\ 1\ m^3 \leq \mu_{L_s} \leq 12\ m^3 \\ \\ 1\ m^3 \leq \mu_{L_v} \leq 6\ m^3 \end{array} \tag{29}$$

The objective function proposed by Downs and Vogel[24] in the TEP definitions is based on the operating costs and can be defined as follows:

$$\begin{array}{c} TC = PC \cdot PR + PrC \cdot PrR + CC \cdot CW + SC \cdot SR \\ \\ TC = \ 7.5973 \frac{\$}{kmol} \cdot PC + 0.1434 \frac{\$}{kmol} \cdot PrR + 0.0536 \frac{\$}{kWh} \cdot CW + 0.0318 \frac{\$}{kg} \cdot SC \end{array} \tag{30}$$

where TC are the total operating costs at the base case, PC and PR are the purge costs and purge flowrate, respectively. Similarly, PrC, CC and SC are the costs associated to the product stream, compressor and steam, and PrR, CW and SR are the product rate, the compressor work and the steam rate, respectively.

Operating costs for this process are primarily determined by the loss of raw materials (in the purge, in the product stream and by means of the two side reactions). Economic costs for the process are determined by summing the costs of the raw materials and the products leaving in the purge stream and the product stream, and using an assigned cost to the amount of F formed. The costs concerning the compressor work and the steam to the stripper are also included. Note that the objective function used in the MINLP formulation will be the mean of these operating costs along the whole simulation time horizon. For this work, this simulation time horizon was set to t = 10 h. This is enough time for stabilization of the TEP. After all these considerations concerning the objective function, the problem can be represented as an MINLP of the form (28):

$$\begin{aligned} & v \in \mathbb{R}^{36}, b \in \{0,1\}^7 \\ & min J(x,v,b) = \overline{Total\ Operating\ Costs\ at\ Base\ Case} \\ & v_0 - 0.5v_0 \leq v \leq v_0 + 0.5v_0 \end{aligned} \tag{31}$$

The lower and upper bound for the decision variables have been set to be the $\pm 50\%$ of the initial value for the decision vector. The reason for this selection is to avoid as much as possible the saturation problems that can be exhibited by the valves. These situations have been detected in preliminary dynamic simulations when considering a value of $\pm 100\%$ of v_0 as bounds for the decision vector.

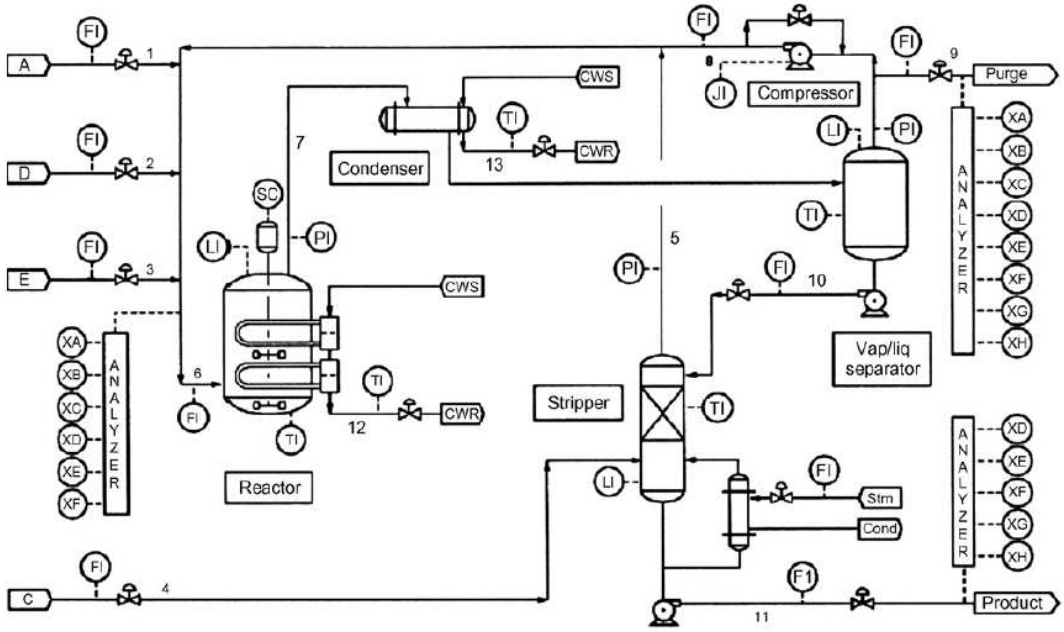

Figure 9: The Tennesee Eastman Process Flowsheet

The histogram depicting the multistart procedure to check the non-convexity of the problem is shown in Figure 10. The number of initial points used was 100 and the local solver used for this case was MISQP.

The histogram shows the practical non-convexity of the problem and the best value reported by the multistart ($f(x) = 135.97$) is far from the best value of 84.19 obtained by `ACOmi` (see Table 9). Out of the 100 MISQP executions with random initial points only 41 converged to a feasible solution. In total the multistart of MISQP required 66518 function evaluations.

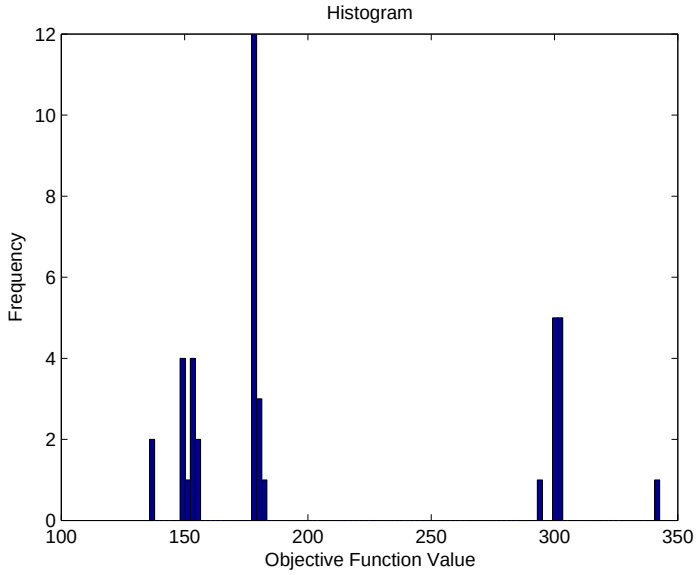

Figure 10: Histogram of solutions obtained from the multistart procedure for TEP

Table 8 lists the parameters of `ACOmi` used for all test runs on this application together with a brief explanation and reference information. In case of this application we assumed two independent `ACOmi` setups regarding the oracle parameter Ω, which is used to evaluate the fitness of ants according to the penalty function (see Section 3). Those two setups were named $\texttt{ACOmi}_{\Omega_1}$ and $\texttt{ACOmi}_{\Omega_2}$ regarding their oracle parameter. It is to note that, except the oracle parameter, all other parameters were set identical between $\texttt{ACOmi}_{\Omega_1}$ and $\texttt{ACOmi}_{\Omega_2}$ as listed in Table 8.

In Table 9 the best (f_{best}), worst (f_{worst}) and mean (f_{mean}) objective function values, obtained by each solver throughout 10 test runs, are reported. In addition the mean number of function evaluations ($\text{eval}_{\text{mean}}$) and the corresponding cpu-time ($\text{time}_{\text{mean}}$) in seconds are given. We assigned a maximal budget of 10000 simulations (function evaluations) as a limit for every test run, but we did not declare a maximal time budget. All test runs started from the same initial point, which can be found together with the lower and upper bounds in Table 13 (see Appendix). Table 13 also shows the optimal vector (x^*) obtained by all tested solvers.

Table 8: $\texttt{ACOmi}_{\Omega_1}$ and $\texttt{ACOmi}_{\Omega_2}$ parameter setup for the TEP

parameter	value	explanation	reference
k	15	number of kernels = size of SA	Section 2
n_{ants}	150	number of ants	Section 4.1.1
dyn_{max}	500	maximal number of ants	Section 4.1.1
dyn_{mean}	10	number of iteration for dyn_{max}	Section 4.1.1
W_{final}	100	weight for $stage_{final}$	Section 4.1.4
$freq$	10	frequency for local solver	Section 4.1.4
Ω_1	100	Oracle for penalty method	Section 3
Ω_2	10^{12}	Oracle for penalty method	Section 3

Table 9: Results for the TEP

solver	f_{best}	f_{worst}	f_{mean}	eval_{mean}	time_{mean}
$\texttt{ACOmi}_{\Omega_1}$	84.19	152.51	112.65	10636	6593.59
$\texttt{ACOmi}_{\Omega_2}$	147.57	148.72	148.06	10113	9843.04
MITS	147.951	149.015	148.484	19309	10435.8
SSM	147.547	148.82	147.991	21822	10857.8

Comparing the performance of MITS, SSm and $\texttt{ACOmi}_{\Omega_2}$, no significant difference can be noticed, all three solver achieved a mean solution objective function value about 148. Analyzing the results of $\texttt{ACOmi}_{\Omega_1}$ on this constrained problem, the importance of the oracle parameter Ω for this constrained problem is striking. The `ACOmi` performance was dramatically increased be applying a reasonable guess of the oracle parameter of 100, leading to solution objective function values about 84. Interestingly the $\texttt{ACOmi}_{\Omega_1}$ setup also provided the worst objective function value of all tested solvers. Nevertheless the mean objective function value obtained by $\texttt{ACOmi}_{\Omega_1}$ is around 112 and much better than the ones of MITS, SSm and $\texttt{ACOmi}_{\Omega_2}$.

To illustrate the convergence rate of the tested solvers, Figure 11 displays (in log-log-scale) the convergence curves corresponding to the best run for each solver and setup.

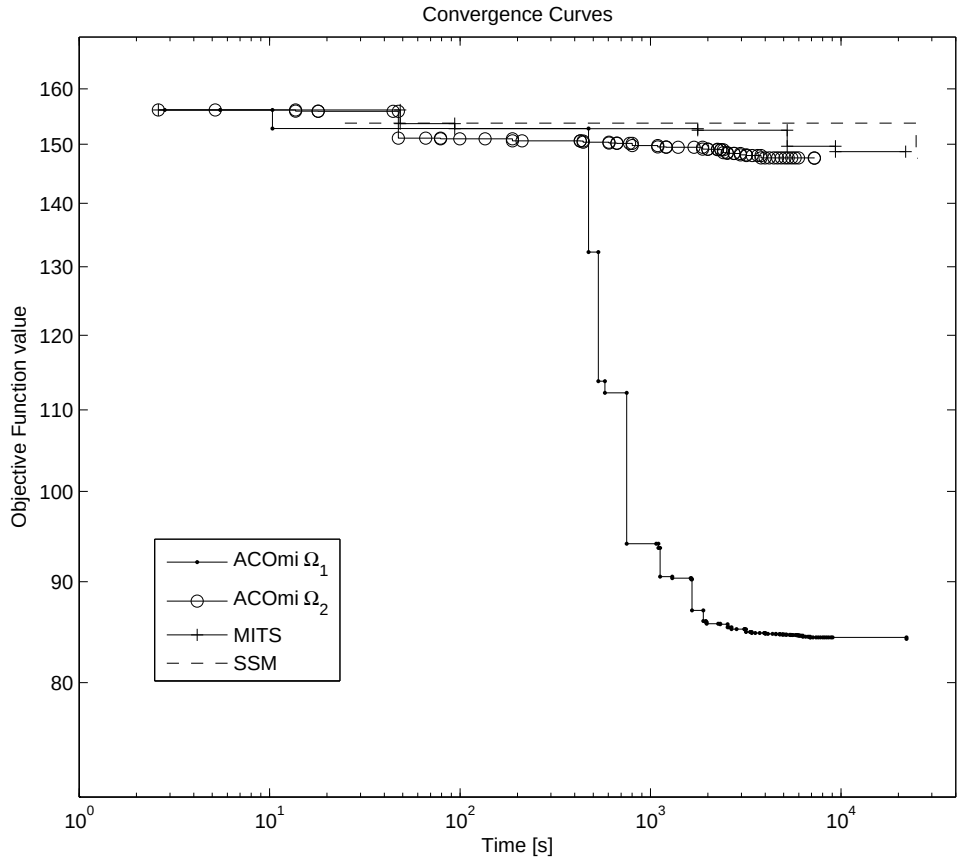

Figure 11: Convergence curves of the best runs of MITS, SSm, $\texttt{ACOmi}_{\Omega_1}$ and $\texttt{ACOmi}_{\Omega_2}$, respectively to the obtained solution objective function value for TEP

Conclusions

We presented an extension of the Ant Colony Optimization metaheuristic enabling the methodology to handle mixed integer variable search domains. Furthermore we introduced a new penalization strategy which can be applied in the extended ACO framework to face constraint optimization problems. A detailed explanation of the hybrid implementation `ACOmi`, incorporating the extended ACO framework and robust oracle penalty method, was given before results on four case studies were presented.

The results on the case studies fortify the success of our new approach for the integrated design and control of nonlinear dynamic models of medium complexity. On every case study `ACOmi` outperformed the rest of compared solvers. In particular the difference in the solution quality obtained by `ACOmi` and the concurrent solvers was significantly on the two mixed integer applications (WWT.COST.3 and TEP). Furthermore the results on the last case study (TEP) illustrate the high potential of the introduced oracle penalty method.

Acknowledgments

We acknowledge the support of the EU and the Marie-Curie Actions program. We thank the project PRISM - MRTN-CT-2004-512233 - *Towards Knowledge-based Processing Systems* for the support.

References

(1) Schweiger, C. A.; Floudas, C. A. Interaction of Design and Control: Optimization with Dynamic Models. In *Optimal Control: Theory, Algorithms, and Applications*; Hager, W. W., Pardalos, P. M., Eds.; Kluwer Academic Publishers, 1997; pp 388–435.

(2) Bansal, V.; Perkins, J. D.; Pistikopoulos, E. N.; Ross, R.; Van Schijndel, J. M. G. Simultaneous design and control optimisation under uncertainty. *Comput. Chem. Eng.* **2000**, *24*, 261–266.

(3) Sakizlis, V.; Perkins, J. D.; Pistikopoulos, E. N. Recent advances in optimization-based simultaneous process and control design. *Comput. Chem. Eng.* **2004**, *28*, 2069–2086.

(4) Schlüter, M.; Egea, J. A.; Banga, J. R. Extended ant colony optimization for non-convex mixed integer nonlinear programming. *Computers & Operations Research* **2009**, *7(36)*, 2217–2229.

(5) Schlüter, M.; Gerdts, M. The oracle penalty method. *J. Global Optim.* **2009**, *submitted*.

(6) Schlüter, M. Global Optimization Software for Mixed Integer Nonlinear Programming (Http://www.midaco-solver/oracle.html), **2009**.

(7) Dorigo, M. *Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale*, Ph.D. thesis, Politecnico di Milano, Italy, 1992.

(8) Socha, K.; Dorigo, M. Ant colony optimization for continuous domains. *Eur. J. Oper. Res.* **2008**, *185*, 1155–1173.

(9) Socha, K. ACO for Continuous and Mixed-Variable Optimization. In *Ant Colony Optimization and Swarm Intelligence*; Dorigo, M., Birattari, M., Blum, C., Gambardella, L. M., Mondada, F., Stützle, T., Eds.; Springer: Berlin-Heidelberg, 2004; pp 25–36.

(10) Box, G. E. P.; Müller, M. E. A note on the generation of random normal deviates. *Ann. Math. Stat.* **1958**, *29*, 610–611.

(11) Exler, O.; Schittkowski, K. A trust region SQP algorithm for mixed-integer nonlinear programming. *Optim. Letters* **2007**, *1*, 269–280.

(12) Exler, O.; Antelo, L. T.; Egea, J. A.; Alonso, A. A.; Banga, J. R. A Tabu search-based algorithm for mixed-integer nonlinear problems and its application to integrated process and control system design. *Comput. Chem. Eng.* **2008**, *32*, 1877–1891.

(13) Egea, J. A.; Rodríguez-Fernández, M.; Banga, J. R.; Martí, R. Scatter search for chemical and bio-process optimization. *J. Global Optim.* **2007**, *37*, 481–503.

(14) Auger, A.; Hansen, N. A Restart CMA Evolution Strategy With Increasing Population Size. *IEEE Congress on Evolutionary Computation, IEEE (CEC)*, 2005; pp 1769–1776.

(15) Gutiérrez, G.; Vega, P. Integrated design of activated sludge process taking into account the closed loop controllability. *Proceedings of ESCAPE-10*, Firenze, 2000; pp 63–69.

(16) Jeppsson, U. *Modelling Aspects of Wastewater Treatment Processes*, Ph.D. thesis, Industrial Electrical Engineering and Automatics (IEA), Lund Institute of Technology (LTH), Sweden, 1996.

(17) Alex, J.; Beteau, J. F.; Copp, J.; Hellinga, C.; Jeppsson, U.; Marsili-Libelli, S.; Pons, M. N.; Spanjers, H.; Vanhooren, H. Benchmark for Evaluating Control Strategies in Wastewater Treatment Plants. *ECC'99 (European Control Conference)*, Karlsruhe (Germany), 1999.

(18) Copp, J. *The COST Simulation Benchmark - Description and Simulator Manual*; COST (European Cooperation in the field of Scientific and Technical Research): Brussels, 2001.

(19) Jeppsson, U.; Pons, M. N. The COST benchmark simulation model-current state and future perspective. *Control Eng. Pract.* **2004**, *12*, 299–304.

(20) Egea, J. A.; Vries, D.; Alonso, A. A.; Banga, J. R. Global Optimization for Integrated Design and Control of Computationally Expensive Process Models. *Ind. Eng. Chem. Res.* **2007**, *46*, 9148–9157.

(21) Henze, M.; GradyJr, C. P. L.; Gujer, W.; Marais, G. V. R.; T., M. *Activated Sludge Model n° 1*; IAWQ Scientific and Technical Report 1, 1987.

(22) Takács, I.; Patry, G. G.; Nolasco, D. A dynamic model of the clarification-thickening process. *Water Res.* **1991**, *25*, 1263–1271.

(23) Vanrolleghem, P. A.; Gillot, S. Robustness and economic measures as control benchmark performance criteria. *Water Sci. Technol.* **2002**, *45*, 117–126.

(24) Downs, J. J.; Vogel, E. F. Plant-wide industrial process control problem. *Comput. Chem. Eng.* **1993**, *17*, 245–255.

(25) Antelo, L. T.; Banga, J. R.; Alonso, A. A. Hierarchical design of decentralized control structures for the Tennessee Eastman Process. *Comput. Chem. Eng.* **2008**, *32*, 1995–2015.

(26) Antelo, L. T.; Otero-Muras, I.; Banga, J. R.; Alonso, A. A. A systematic approach to plant-wide control based on thermodynamics. *Comput. Chem. Eng.* **2007**, *31*, 677–691.

(27) Luyben, M. L.; Tyreus, B. D.; Luyben, W. L. Plantwide Control Design Procedure. *AIChE J.* **1997**, *43*, 3161–3174.

6 Appendix

Detailed information on lower and upper bounds, initial and best vectors are given here for the considered case studies of Section 5.

Table 10: Information on bounds, initial and best vectors for WWT.COST.1

variable	lower bound	upper bound	initial value	x^*_{MITS}	$x^*_{\texttt{ACOmi}}$	x^*_{CMAES}	x^*_{SSm}
$K_{(O)}$	100	1000	750.62	896.77	514.75	517.88	378.49
$\tau_{i(O)}$	0.0007	0.7	0.5069	0.0021	0.0007	0.0007	0.0007
$K_{(N)}$	100	50000	27831	24657	20763	20370	18865
$\tau_{i(N)}$	0.01	1	0.0932	0.0292	0.0270	0.0274	0.0247
objective function value			35.91	0.7433	0.5273	0.5313	0.5354

Table 11: Information on bounds, initial and best vectors for WWT.COST.2

variable	lower bound	upper bound	initial value	x^*_{MITS}	$x^*_{\texttt{ACOmi}}$	x^*_{CMAES}	x^*_{SSm}
$K_{(O)}$	100	1000	955.12	625.72	619.42	1000.0	671.32
$\tau_{i(O)}$	0.0007	0.7	0.1623	0.0026	0.0013	0.0007	0.0007
$K_{(N)}$	100	50000	303810	13195	19014	28487	20167
$\tau_{i(N)}$	0.01	1	0.4911	0.0161	0.0283	1.0000	0.0279
KLa_3	160	360	338.26	220.11	173.78	172.46	202.41
KLa_4	160	360	312.42	197.12	213.16	229.26	211.24
Q_r	0.0001	36892	222750	11913	18701	14647	18605
Q_w	100	1844.6	132.28	243.68	390.71	286.51	404.04
objective function value			119430	34167	33678	34852	33993

Table 12: Information on bounds, initial and best vectors for WWT.COST.3

variable	lower bound	upper bound	initial value	x^*_{MITS}	$x^*_{\texttt{ACOmi}}$	x^*_{SSm}
$K_{(O)}$	100	1000	500	499.950	571.902	522.714
$\tau_{i(O)}$	0.0007	0.7	0.001	0.00096	0.00149	0.00253
$\tau_{t(O)}$	0.0001	0.7	0.0002	0.00020	0.00025	0.18933
$K_{(N)}$	100	50000	15000	14998.0	24897.5	14366.0
$\tau_{i(N)}$	0.01	1	0.05	0.04994	0.03388	0.04563
$\tau_{t(N)}$	0.0001	0.07	0.03	0.02999	0.00011	0.03336
KLa_1	0	360	0	0.00178	0.19231	0
KLa_2	0	360	0	0.01972	35.3000	71.0659
KLa_3	0	360	240	240.020	132.393	126.026
KLa_4	0	360	240	239.920	216.948	183.455
Q_r	0	36892	18446	18444.0	16382.9	10315.7
Q_w	0	1844.6	385	385.100	351.598	199.944
$Lfeed$	1	10	5	8	7	7
objective function value			35225	34296	32544	33104

Table 13: Information on bounds, initial and best vectors for TEP

variable	lower bound	upper bound	initial value	ACOmi$_{\Omega_1}$	ACOmi$_{\Omega_2}$	MITS	SSM
1	0.0005	0.0015	0.00141	0.0014	0.00141	0.00116	0.00142
2	0.0015	0.0045	0.00191	0.00151	0.00336	0.00217	0.00359
3	9e-011	2.7e-010	2.01e-010	9e-011	9e-011	2.01e-010	1.98e-010
4	10	30	15.4	30	22.4	16.2	16.6
5	3.5e-007	1.05e-006	5.04e-007	8.97e-007	7.15e-007	6.44e-007	6.52e-007
6	0.0002	0.0006	0.000485	0.000446	0.000262	0.000592	0.000525
7	0.002	0.006	0.0042	0.00317	0.00386	0.00511	0.00378
8	1.6	4.8	4.61	4.8	1.6	1.6	1.97
9	-0.03	-0.01	-0.0234	-0.0193	-0.0298	-0.0172	-0.0152
10	-0.075	-0.025	-0.0398	-0.0339	-0.025	-0.0348	-0.0434
11	5	15	14.4	5.18	9.8	12.6	8.12
12	-0.00015	-5e-005	-9.18e-005	-5e-005	-0.000112	-5.65e-005	-7.54e-005
13	-0.048	-0.016	-0.0198	-0.027	-0.0186	-0.0185	-0.0355
14	0.00045	0.00135	0.00112	0.000681	0.00135	0.00105	0.000925
15	0.000625	0.00187	0.0011	0.00128	0.00187	0.00187	0.00187
16	-12	-4	-6.2	-10.7	-12	-7.63	-8.06
17	50	150	66.3	120	94.6	93.5	101
18	16	48	46.6	19.9	48	44.6	39.5
19	23	69	32	32.9	68	37.2	25.2
20	8.33e-006	2.5e-005	2.13e-005	1.53e-005	9.24e-006	1.32e-005	8.33e-006
21	8.33e-006	2.5e-005	1.86e-005	2.24e-005	1.08e-005	1.27e-005	8.88e-006
22	2.08	6.25	2.76	4.05	2.51	6.22	5.41
23	0.833	2.5	0.885	0.833	1.57	1.64	1.67
24	2.08	6.25	3.29	2.08	2.08	3	4.1
25	8.33e-006	2.5e-005	2.45e-005	8.33e-006	2.42e-005	1.86e-005	1.5e-005
26	8.33e-006	2.5e-005	2.42e-005	8.66e-006	1.13e-005	2.06e-005	1.42e-005
27	1	3	1.46	1.13	2.88	2.09	2.11
28	0.167	0.5	0.486	0.387	0.263	0.403	0.223
29	1.67	5	3.93	2.4	1.67	2.1	3.75
30	0.00833	0.025	0.00925	0.025	0.0229	0.0201	0.0184
31	0.167	0.5	0.367	0.167	0.192	0.452	0.198
32	0.833	2.5	1.49	2.04	1.11	1.49	1.81
33	4.68	14.1	6.7	14	4.68	5.19	9.27
34	12.5	37.5	17.1	33.8	14.2	12.5	12.5
35	0.0625	0.188	0.0721	0.137	0.188	0.117	0.179
36	4.17	12.5	4.23	11.5	11.3	8.67	5.93
37	0	1	1	0	1	1	1
38	0	1	0	1	0	0	0
39	0	1	1	0	1	1	1
40	0	1	0	1	0	0	0
41	0	1	0	0	0	0	0
42	0	1	0	0	0	0	0
43	0	1	0	0	0	0	0
objective	function	value:	159.330	84.190	147.570	147.950	147.540