ABCM Symposium Series in Mechatronics - Vol. 4 - pp.72-79

CONTROLLER TUNING USING MULTIOBJECTIVE PARTICLE SWARM OPTIMIZATION APPLIED TO A QUADRUPLE-TANK PROCESS

Helon Vicente Hultmann Ayala, helon.ayala@pucpr.br
Pontifical Catholic University of Paraná, PUCPR
Undergraduate Program at Mechatronics Engineering
Imaculada Conceição, 1155, Zip code 80215-901, Curitiba, Paraná, Brazil

Leandro dos Santos Coelho, leandro.coelho@pucpr.br
Pontifical Catholic University of Paraná, PUCPR
Industrial and Systems Engineering Graduate Program, LAS/PPGEPS
Imaculada Conceição, 1155, Zip code 80215-901, Curitiba, Paraná, Brazil

Abstract. *Conventional PI (Proportional-Integral) controllers are characterized with simple structure and simple design procedures. They enable good control performance and are therefore widely applied in industry. However, in a number of nonlinear processes cases, such as those when parameter variations take place and/or when disturbances are present, design of PI control system based on a multiobjective optimization approach may be a better choice. In this paper, we introduce an improved multiobjective particle swarm optimization (IMPSO) approach. This paper presents the design and the tuning of a PI control through IMPSO. Simulation numerical results of PI control and convergence of the IMPSO is presented and discussed with application in a multivariable quadruple-tank process. The proposed design method is intuitive and practical that offers an effective way to implement simple but robust solutions covering a wide range of process perturbation and, in addition, provides excellent tracking performance without resorting to excessive control.*

Keywords*: multiobjective optimization, PI control, particle swarm optimization, nonlinear process, multivariable process.*

1. INTRODUCTION

In many fields of science, the procedure of optimization sometimes has more than one objective, thus the need for multiobjective optimization is obvious. One of the factors that differentiate single objective optimization when compared to multiobjective optimization is that the optimum solution for multiobjective optimization is not necessarily unique. In a typical multiobjective problem optimization (also known as multicriterion optimization), there exists a family of equivalent solutions that are superior to the rest of the solutions and are considered equal from the perspective of simultaneous optimization of multiple (and possibly competing) objective functions. In other words, in multiobjective optimization there is no single optimal solution. Instead, the interaction of multiple objectives yields a set of efficient (noninferior) or non-dominated solutions, known as Pareto-optimal solutions, which give a decision maker more flexibility in the selection of a suitable alternative.

The Pareto optimal solutions of a multiobjective optimization problem often distribute very regularly in both decision and objective space. A problem that arises from the existence of multiple optimal solutions however is how to normalize, prioritize and weight the contributions of the various objectives in arriving at a suitable measure. Also these objectives can interact or conflict with each other, for example, increasing one can reduce others in turn and this can happen in nonlinear ways. Most of the classical Operational Research methods of obtaining solutions or approaching the Pareto front (including the multicriterion decision-making methods) focus on the first stage of ranking the objectives, i.e. trying to reduce the design space to a more easily managed mathematical form (since most such problems are far too complex to enumerate and evaluate all the possible combinations in any reasonable time) (Khare, 2002).

For a multiobjective optimization problem, any two solutions can have one of two possibilities: one dominates the other or none dominates the other. In general, the goal of a multiobjective optimization algorithm is not only to guide the search towards the Pareto-optimal front but also to maintain population diversity in the set of the Pareto optimal solutions. The recent studies on evolutionary algorithms (Van Veldhuizen *et al*., 2000; Deb, 2001; Coello, 1999; Coello *et al*., 2002) have shown that the population-based algorithms, such as genetic algorithms, tabu search, differential evolution, and evolution strategies, are potential candidate to solve multiobjective optimization problems and can be efficiently used to eliminate most of the difficulties of classical single objective methods such as the sensitivity to the shape of the Pareto-optimal front and the necessity of multiple runs to find multiple Pareto-optimal solutions (Abido, 2009).

On the other hand, most optimization problems in control systems (Carvalho and Ferreira, 1995; Liu and Wang, 2000; Liao and Li, 2002; Zambrano and Camacho, 2002; Ayala and Coelho, 2008; Panda, 2009) involve the optimization of more than one objective function, which in turn can require a significant computational time to be evaluated. In some studies this problem of evaluating more than one objective function was treated as one only objective function that summarizes all objective functions to be optimized, as a sum of all of them for example.

However, the limitation of this method has already been commented in literature (Van Veldhuizen *et al.*, 2000; Deb, 2001).

In this context, a modern meta-heuristic algorithm that can be useful and effective tool for optimization applications in control systems is the particle swarm optimization (PSO). PSO is a population-based approach of swarm intelligence field that was first developed by James Kennedy and Russell Eberhart (Kennedy and Eberhart, 1995; Eberhart and Kennedy, 1995). Their original idea was to simulate the social behavior of a flock of birds trying to reach an unknown destination (fitness function), e.g., the location of food resources when flying through the field (search space).

In recent literature, several PSO approaches to handle multiple objectives have been proposed (Sierra and Coello, 2006). This paper presents the design and the tuning of two decoupled proportional-integral (PI) controllers through of an improved multiobjective PSO (IMOPSO) approach inspired on Raquel and Naval (2005). Simulation numerical results of PI control and convergence of the IMOPSO is presented and discussed with application in a multivariable quadruple-tank process.

The remaining sections of this paper are organized as follows: in section 2, a description of quadruple-tank process is detailed. Section 3 presented the fundamentals of multiobjective optimization, MOPSO and IMOPSO approaches. In sections 4 and 5, the simulation results and conclusion are presented, respectively.

2. DESCRIPTION OF QUADRUPLE-TANK PROCESS

The quadruple-tank process and consists of four interconnected water tanks and two pumps. Its inputs are v_1 and v_2 (input voltages to the pumps) and the outputs are y_1 and y_2 (voltages from level measurement devices) (Johansson, 2000). The quadruple-tank process can easily be build by using two double-tank processes, which are standard processes in many control laboratories. The schematic diagram of quadruple-tank process is presented in Figure 1.

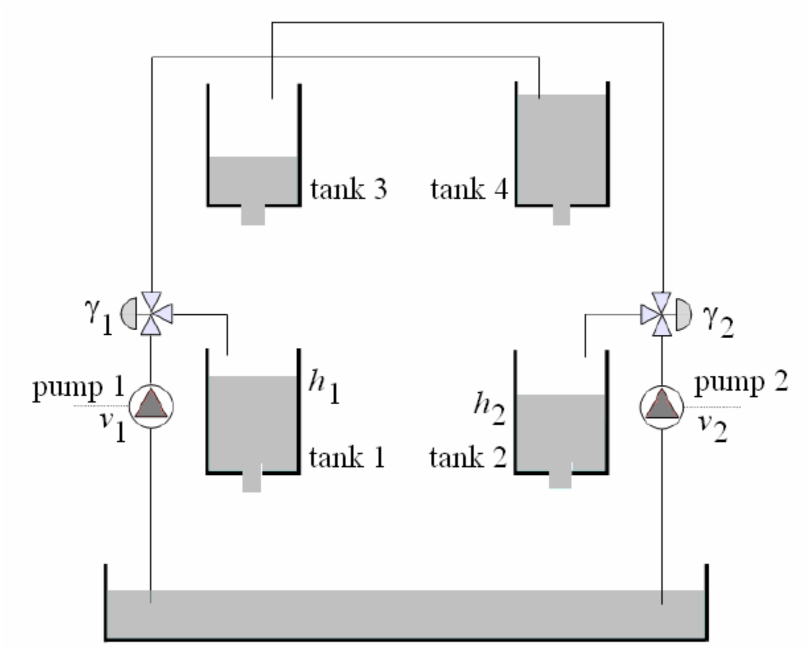


Figure 1. Schematic diagram of quadruple-tank process.

For this process, mass balances and Bernoulli's law yield (Johansson, 2000):

$$\frac{dh_1}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1} v_1 \tag{1}$$

$$\frac{dh_2}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2} v_2 \tag{2}$$

$$\frac{dh_3}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{A_3} v_2 \tag{3}$$

$$\frac{dh_4}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{A_4}v_1 \tag{4}$$

where A_i is the cross-section of tank i, a_i is the cross-section of outlet hole and h_i is the water level. The voltage applied to pump i is v_i, and the corresponding flow is k_iv_i. The parameters γ_1, $\gamma_2 \in (0, 1)$ are determined from how the valves are set. The flow to Tank 1 is $\gamma_1k_1v_1$ and the flow to tank 4 is $(1 - \gamma_1)\ k_1v_1$ and similarly to tank 2 and tank 3. The acceleration of gravity is denoted as g. The measured level signals are k_ch_1 and k_ch_2. These signal represented the outputs signals y, i.e., $y_1(t)=k_ch_1(t)$ and $y_2(t)=k_ch_2(t)$, where t is the time. The adopted time sampling in this work was 1 s. The parameter values used in this paper, as in Johansson (2000), are given in Table 1.

Table 1. Parameters values adopted for the quadruple-tank process.

Parameter	Unit	Value
A_1, A_3	cm²	28
A_2, A_4	cm²	32
a_1, a_3	cm²	0.071
a_2, a_4	cm²	0.057
k_c	V/cm	0.50
g	cm/s²	981

In terms of two PI controllers, in this work is considered K(s) with the following classical structure:

$$K(s) = \begin{bmatrix} k_{11}(s) & \cdots & k_{1n}(s) \\ \vdots & \ddots & \vdots \\ k_{n1}(s) & \cdots & k_{nn}(s) \end{bmatrix}. \tag{5}$$

where n = 2. The form of $k_{ij}(s)$, $i,j \in n = \{1,2,...,n\}$ is given by

$$k_{ij}(s) = Kp_{ij}\left(1 + \frac{1}{Ti_{ij}\cdot s} + Td_{ij}\cdot s\right), \tag{6}$$

where Kp_{ij} is the proportional gain, and Ti_{ij} is the integral gain. In this work, the decoupled PI design was adopted, where the proportional and integral gains are null for $i \neq j$. The MOPSO and IMOPSO approaches must search the parameters of a 2x2 decoupled PI, i. e., search the parameters Kp_{11}, Kp_{22}, Ti_{11} and Ti_{22}.

3. FUNDAMENTALS OF MULTIOBJECTIVE OPTIMIZATION AND THE PSO APPROACH

This section presents the fundamentals of multiobjective optimization and PSO. First, a brief overview of the multiobjective optimization is provided, and finally the design of the MOPSO proposed by Raquel and Naval (2005) and the IMOPSO algorithm are discussed.

3.1. Multiobjective optimization

In contrast to single-objective optimization, it is essential to obtain a well-distributed and diverse solution set for finding the final tradeoff in multi-objective optimization. Multiobjective optimization can be defined as the problem of finding a vector of decision variables that satisfies constraints and optimizes a vector function whose elements represent the objective functions. A general multiobjective optimization problem containing a number of objectives to be minimized and (optional) constraints to be satisfied can be written as:

$$\text{Minimize } f_m(\mathbf{X}),\ m = 1, 2, \ldots, M$$

$$\text{subject to constraint } g_k(\mathbf{X}) \leq c_k\ ,\ k = 1, 2, \ldots, K$$

where $\mathbf{X} = \{x_n,\ n = 1, 2, \ldots, N\}$ is a vector of decision variables and $\mathbf{F} = \{f_m,\ m = 1, 2, \ldots, M\}$ are M objectives to be minimized (Lu and Yen, 2003).

In a typical multiobjective optimization problem, there exists a family of equivalent solutions that are superior to the rest of the solutions and are considered equal from the perspective of simultaneous optimization of multiple (and possibly competing) objective functions. Such solutions are called noninferior, nondominated, or Pareto-optimal solutions, and are such that no objective can be improved without degrading at least one of the others, and, given the constraints of the model, no solution exist beyond the true Pareto front. The goal of multiobjective algorithms is to locate the (whole) Pareto front.

Each objective component of any nondominated solution in the Pareto optimal set can only be improved by degrading at least one of its other objective components. A vector f_a is said to dominate another vector f_b, denoted as

$$f_a \prec f_b \text{ iff } f_{a,i} \leq f_{b,i} \quad \forall i = \{1,2,...,M\} \text{ and } \exists j \in \{1,2,...,M\}, \text{ where } f_{a,i} \prec f_{b,i}.$$

Summarizing, there are two goals in multiobjective optimization: i) to discover solutions as close to the Pareto-front as possible, and ii) to find solutions as diverse as possible in the obtained nondominated front.

Methods of multiobjective optimization can be classified in many ways according to different criteria. Hwang and Masud (1979) classify the methods according to the participation of the decision maker in the solution process. The classes are: i) methods where no articulation of preference information is used (no-preference methods); ii) methods where a posteriori articulation of preference information used (a posteriori methods); iii) methods where a priori articulation of preference information used (a priori methods); and iv) methods where progressive articulation of preference information is used (interactive methods).

3.2. PSO algorithm for multiobjective optimization

The PSO algorithm employs a number of particles that constitute a swarm. In PSO, it starts with a random initialization of a population (swarm) of individuals (particles) in the search space and works on the social behavior of the particles in the swarm.

These particles fly with a certain velocity and find the global best position after some iteration. At each iteration, each particle can adjust its velocity vector, based on its momentum and the influence of its best position (*pbest* - *p*ersonal *best*) as well as the best position of its neighbors (*gbest* - *g*lobal *best*), and then compute a new position that the ‘‘particle’’ is to fly to. On other words, it finds the global optimum by simply adjusting the trajectory of each individual towards its own best location and towards the best particle of the swarm at each generation of evolution. The swarm direction of a particle is defined by the set of particles neighboring the particle and its history experience.

Moore and Chapman (1999) proposed the first extension of the PSO strategy for solving multi-objective problems in an unpublished manuscript. There have been several recent fundamental proposals using PSO to handle multiple objectives, surveyed in Sierra and Coello (2006).

However, the high speed of convergence in MOPSO approaches often implies a rapid loss of diversity during the optimization process. In this context, several MOPSO have difficulties in controlling the balance between explorations and exploitations.

Raquel and Naval (2005) propose a multiobjective PSO (MOPSO) incorporating the concept of nearest neighbor density estimator for selecting the global best particle and also for deleting particles from the external archive of nondominated solutions. When selecting a leader, the archive of nondominated solutions is sorted in descending order with respect to the density estimator, and a particle is randomly chosen from the top part of the list. On the other hand, when the external archive is full, it is again sorted in descending order with respect to the density estimator value and a particle is randomly chosen to be deleted, from the bottom part of the list. This approach uses the mutation operator proposed in Coello, Pullido and Lechuga (2004) in such a way that it is applied only during a certain number of generations at the beginning of the process. Finally, the authors adopt the constraint-handling technique from the NSGA-II (Deb *et al.*, 2002).

The procedure for implementing the MOPSO given in (Raquel and Naval, 2005) is given by the following steps:

i) Initialize a population or swarm of particles with random positions and velocities in the *n* dimensional problem space using uniform probability distribution function. Set the generation counter, $t = 0$;

ii) Evaluate the particles and store the nondominated particles in swarm in an external archive *A*;

iii) Compute the crowding distance values of each nonodominated solution in archive *A*;

iv) Sort the nondominated solutions in *A* in descending crowding distance values;

v) Randomly select the global best guide for the swarm form a specified top portion (e.g. top 10%) for the sorted archive *A* and store its position to *gbest*.

vi) Change the velocity, v_i, and position of the particle, x_i, according to equations:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot ud \cdot [p_i(t) - x_i(t)] + c_2 \cdot Ud \cdot [p_g(t) - x_i(t)] \tag{7}$$

$$x_i(t+1) = x_i(t) + \Delta t \cdot v_i(t+1) \tag{8}$$

where w is the inertia weight; i=1,2,…,N indicates the number of particles of population (swarm); t=1,2,…$t_{\max}$, indicates the generations (iterations), w is a parameter called the inertial weight; $v_i = [v_{i1}, v_{i2}, ..., v_{in}]^T$ stands for the velocity of the i-th particle, $x_i = [x_{i1}, x_{i2}, ..., x_{in}]^T$ stands for the position of the i-th particle of population, and $p_i = [p_{i1}, p_{i2}, ..., p_{in}]^T$ represents the best previous position of the i-th particle.

Positive constants c_1 and c_2 are the cognitive and social factors, respectively, which are the acceleration constants responsible for varying the particle velocity towards *pbest* and *gbest*, respectively. Index g represents the index of the best particle among all the particles in the swarm. Variables ud and Ud are two random functions with uniform distribution in the range [0,1].

Equation (8) represents the position update, according to its previous position and its velocity, considering $\Delta t = 1$.

vii) Perform the mutation operation proposed in Coello *et al.* (2004) with probability of 0.5;

viii) Evaluate the particles in swarm;

ix) Insert all new nondominated solution in swarm into A if they are not dominated by any of the stored solutions. All dominated solutions in the archive by the new solution are removed from the archive. If the archive is full, the solution to be replaced is determined by the following steps: a) compute the crowding distance values of each nondominated solution in the archive A; b) sort the nondominated solutions in A in descending crowding distance values, and iii) randomly select a particle from a specified bottom portion (e.g. lower 10%) which comprise the most crowded particles in the archive then replace it with the new solution;

x) Increment the generation counter, $t = t + 1$;

xi) Return to Step (iii) until a stop criterion is met, usually a sufficiently good fitness or a maximum number of iterations, $t_{\max}$. In this work, the $t_{\max}$ value is adopted.

3.3. The proposed IMOPSO approach

The proposed IMOPSO approach uses social and cognitive time-variant factors (Ratnaweera and Halgamuge, 2004) and an operator of velocity updating based on truncated Gaussian distribution (Coelho and Krohling, 2005). In this case the equation (7) presented in section 3.2 is modified for:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot ud \cdot [p_i(t) - x_i(t)] + c_2 \cdot Gd \cdot [p_g(t) - x_i(t)] \tag{9}$$

where Gd are numbers generated with Gaussian distribution in range [0,1]. The updating of c_2 is given by Ratnaweera and Halgamuge (2004):

$$c_2 = \left(c_{2f} - c_{2i}\right) \cdot \frac{t}{t_{\max}} + c_{2i} \tag{10}$$

where c_{2i} and c_{2f} are positive constants.

4. SIMULATION RESULTS

The experiments were conducted for 30 independent runs to evaluate the performance of MOPSO and IMOPSO on the tuning of two PI controllers applied to the quadruple-tank process. The adopted setup for the MOPSO was c_1 = c_2 = 1.0 and c_1 = 1.0, c_{2i} =0.4, c_{2f} = 1.0 for the IMOPSO, and the range of the inertia weight w is from 0.5 to 0.3 during the generations for the MOPSO and IMOPSO approaches. The population size was 20 particles, stopping criterion, $t_{\max}$, of

200 generations, and external archive size equal to 500. The search space was Kp_1, $Kp_2 \in$ [-50, 50] and Ti_1, $Ti_2 \in$ [0, 400]. The total of samples to evaluate the fitness function and time sampling Ts = 1 s. Unstable solutions are penalized. The fitness function (minimization problem) is given by

$$f = f_1 + f_2 \tag{10}$$

$$f_1 = \sum_{t=1}^{N} \left[y_{r,1}(t) - y_1(t) \right]^2 \tag{11}$$

$$f_2 = \sum_{t=1}^{N} \left[y_{r,2}(t) - y_2(t) \right]^2 \tag{12}$$

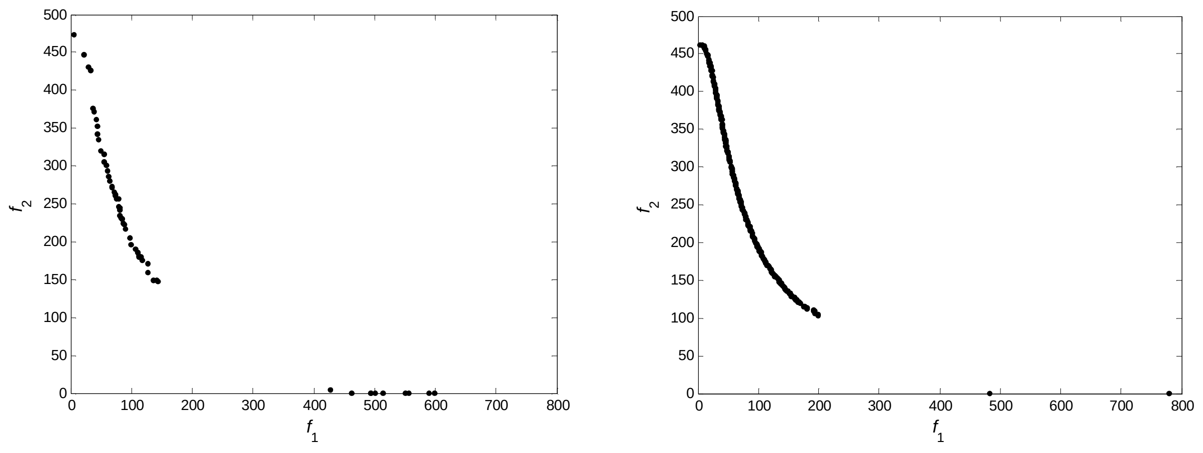
where $y_{r,1}$ and $y_{r,2}$ are the setpoints for the outputs 1 and 2, y_1 and y_2 are the outputs of process.

Simulation results were presented in Figs. 2(a) and 2(b) showed that the non-dominated solutions of best run of 30 runs obtained by MOPSO with 59 solutions and IMOPSO with 290 solutions. It is observed that the IMOPSO dominated the solutions obtained by MOPSO. Furthermore, other important information is about the mean of Pareto solutions in 30 runs. In this work, the MOPSO obtained mean of 35 solutions and the IMOPSO obtained mean of 88 solutions in Pareto front.

The metric of spacing (S) gives an indication of how evenly the solutions are distributed along the discovered front. The spacing of Pareto front (mean of 30 runs) of MOPSO was 1.5889. On the other hand, the spacing of IMOPSO was 5.7998. In terms of spacing, the IMOPSO maintains a relatively good spacing metric and obtained a better distribution that the MOPSO of non-dominated solutions in Pareto front.

A good compromise solution in terms of harmonic mean of f_1 and f_2 values for the IMOPSO (with f_1 = 119.6016 and f_2 = 165.5758) is presented in Fig. 3. The gains obtained by IMOPSO in this case were Kp_1=37.8455, Kp_2 =-0.4821, Ti_1=239.7113 and Ti_2 = 210.4079.

In this context, an important comment related the multiobjective optimization must be mentioned. A multi-objective optimization problem differs from a single-objective optimization problem because it contains several objectives that require optimization. In case of single-objective optimization problems, the best single design solution is the goal. But for multi-objective problems, with several and possibly conflicting objectives, there is usually no single optimal solution. Therefore, the decision maker is required to select a solution from a finite set by making compromises. A suitable solution should provide for acceptable performance over all objectives (Panda, 2009).



(a) MOPSO (b) IMOPSO

Figure 2. Pareto front of MOPSO and IMOPSO approaches.

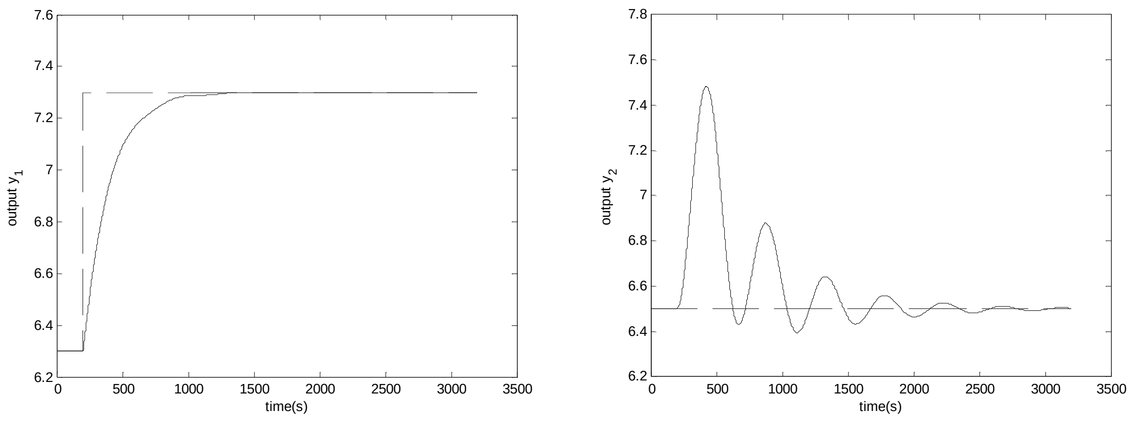


Figure 3. Best result in terms of harmonic mean of f_1 and f_2 for the IMOPSO.

5. CONCLUSION

PSO is a powerful metaheuristic approach inspired by observing the bird flocks and fish schools. Recent works (Sierra and Coello, 2006; Abido, 2009) showed that basic PSO algorithm can be modified to accommodate the problem formulation of multiobjective problems, which is to search for a well extended, uniformly distributed, and near-optimal Pareto front.

In this paper, the MOPSO (Raquel and Naval, 2005) and the proposed IMOPSO design presented promising results to tune the decoupled PI controllers when applied to a quadruple process. The IMOPSO allows the discovery of a well-distributed and diverse solution set for PI tuning without compromising the convergence speed of the algorithm. Furthermore, the MOPSO presented competitive results in terms of proximity, diversity, and distribution with the MOPSO for the studied case.

The proposed IMOPSO method is expected to be extended to other multivariable processes with parameter uncertainties and perturbations. The aim of future works is to investigate the use of MOPSO and IMOPSO approaches tune model-based predictive controllers.

6. ACKNOWLEDGEMENTS

This work was supported by the National Council of Scientific and Technologic Development of Brazil — CNPq — under Grant 309646/2006-5/PQ and by the Institutional Program of Scientific Initiation Grants from the Pontifical Catholic University of Paraná — PIBIC/PUCPR.

7. REFERENCES

Abido, M.A., 2009, “Multiobjective Particle Swarm Optimization for Environmental/Economic Dispatch Problem”, Electric Power Systems Research, Vol. 79, No. 7, pp. 1105-1113.

Ayala, H.V.H. and Coelho, L.S., 2008, “A Multiobjective Genetic Algorithm Applied to Multivariable Control Optimization”, ABCM Symposium Series in Mechatronics, Vol. 3, pp. 736-745.

Carvalho, J.R.H. and Ferreira, P.A.V., 1995, “Multiple-Criterion Control: A Convex Programming Approach”, Automatica, Vol. 31, pp. 1025-1029.

Coelho, L.S. and Krohling, R.A., 2005, “Predictive controller tuning using modified particle swarm optimisation based on Cauchy and Gaussian distributions,” in Soft Computing: Methodologies and Applications, F. Hoffmann, M. Köppen, and R. Roy (eds.), Springer Engineering Series in Advances in Soft Computing, Springer, London, UK, pp. 287-298.

Coello, C.A.C. Coello, Pulido, G.T., and Lechuga, M.S., 2004, “Handling Multiple Objectives with Particle Swarm Optimization”, IEEE Transactions on Evolutionary Computation, Vol. 8, No. 3, pp. 256-279.

Coello, C.A.C., 1999, “A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization”, Knowledge and Information Systems, Vol. 1, No. 3, pp. 269-308.

Coello, C.A.C., Van Veldhuizen, D.A. and Lamont, G.B., 2002, “Evolutionary Algorithms for Solving Multi-Objective Problems”, Kluwer Academic Publishers, New York, NY, USA.

Deb, K., 2001, “Multi-Objective Optimization using Evolutionary Algorithms” Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons.

Deb, K., Pratap, A., Agrawal, S. and Meyarivan, T., 2002, “A Fast and Elitist Multiobjective Genetic Algorithms: NSGA-II”, IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197.

Eberhart R.C. and Kennedy, J.F., 1995, “A New Optimizer Using Particle Swarm Theory”, Proceedings of International Symposium on Micro Machine and Human Science, Japan, pp. 39-43.

Hwang, C.L. and Masud, A S M., 1979, “Multiple Objective Decision Making Methods and Applications: A State of the Art Survey,” Berlin, Heidelberg: Springer-Verlag.

Johansson, K.H., 2000, “The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero”, IEEE Transactions on Control Systems Magazine, Vol. 8, No. 3, pp. 456-465.

Kennedy, J. F. and Eberhart, R.C., 1995, “Particle Swarm Optimization”, Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942-1948.

Khare, V., 2002, “Performance Scaling of Multi-Objective Evolutionary Algorithms”, Master thesis, School of Coputer Science, The University of Birmingham, Birmingham, UK.

Liao, L.-Z. and Li, D., 2002, “Adaptice Differential Dynamic Programming for Multiobjective Optimal Control”, Automatica, Vol. 38, pp. 1003-1015.

Liu, W. and Wang, G., 2000, “Auto-Tuning Procedure for Model-Based Predictive Controller”, Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, Nashville, Tennessee, USA, Vol. 5, pp. 3421-3426.

Lu, H. and Yen, G.G., 2003, “Rank-density-based multiobjective genetic algorithm and benchmark test function study,” IEEE Transactions on Evolutionary Computation, vol. 7, no. 4, pp. 325-343.

Moore, J. and Chapman, R., 1999, “Application of Particle Swarm to Multiobjective Optimization”, Department of Computer Science and Software Engineering, Auburn University.

Panda, S., 2009, “Multi-objective Evolutionary Algorithm for SSSC-based Controller Design”, Electric Power Systems Research, Vol. 79, No. 6, pp. 937-944.

Raquel, C.R. and Naval, P.C.Jr., 2005, “An Effective Use of Crowding Distance in Multiobjective Particle Swarm Optimization”, In Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2005), Washington DC, USA, 2005.

Ratnaweera, A., Halgamuge, S.K. and Watson, H.C., 2004, “Self-organizing Hierarchical Particle Swarm Optimizer with Time Varying Acceleration Coefficients,” IEEE Transactions on Evolutionary Computation, Vol. 8, No. 3, pp. 240-255.

Sierra, M.R. and Coello, C.A.C., 2006, “Multi-objective Particle Swarm Optimizers: A Survey of the State-of-the-art”, International Journal of Computational Intelligence Research, vol. 2, no. 3, pp. 287-308.

Van Veldhuizen, D.A. and Lamont, G.B., 2000, “Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art”, Evolutionary Computation, Vol. 8, No. 2, pp. 125-147.

Zambrano, D. and Camacho, E. F., 2002, “Application of MPC with Multiple Objective for a Solar Refrigeration Plant”, Proceedings of the IEEE International Conference on Control Applications, Glasgow, Scotland, UK, pp. 1230-1235.

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.