Microcontroller Implementation of a Multi Objective Genetic Algorithm for Real-Time Intelligent Control

Martin Dendaluce[2], Juan José Valera[2], Vicente Gómez-Garay[2], Eloy Irigoyen[1,2], and Ekaitz Larzabal[2]

[1] Computational Intelligence Group
Department of Systems Engineering and Automatic Control
University of the Basque Country (UPV/EHU), ETSI, 48013 Bilbao
[2] Intelligent Control Research Group
Department of Systems Engineering and Automatic Control
University of the Basque Country (UPV/EHU), ETSI, 48013 Bilbao
{mdendaluce001,elarzabal001}@ikasle.ehu.es
{juanjose.valera,vicente.gomez,eloy.irigoyen}@ehu.es

Abstract. This paper presents an approach to merge three elements that are usually not thought to be combined in one application: evolutionary computing running on reasonably priced microcontrollers (μC) for real-time fast control systems. A Multi Objective Genetic Algorithm (MOGA) is implemented on a 180MHz μC.A fourth element, a Neural Network (NN) for supporting the evaluation function by predicting the response of the controlled system, is also implemented. Computational performance and the influence of a variety of factors are discussed. The results open a whole new spectrum of applications with great potential to benefit from multivariable and multiobjective intelligent control methods in which the hybridization of different soft-computing techniques could be present. The main contribution of this paper is to prove that advanced soft-computing techniques are a feasible solution to be implemented on reasonably priced μC -based embedded platforms.

Keywords: Soft-Computing, Intelligent Control, Multi Objective Genetic Algorithm, NSGA-II, Artificial Neural Network, Microcontroller.

1 Introduction

Soft-computing techniques such as Genetic Algorithms (GA), as well as other intelligent techniques like NNs and Fuzzy Logic (FL), have already proven to be a serious alternative not only to complement but also to replace conventional algorithms in a wide diversity of applications[1].

Intelligent control schemes based on soft-computing techniques are showing to be an excellent solution for dealing with the difficulty of optimizing nonlinear and multi objective control systems. They also adapt conveniently to diverse real-world issues such as system parameter alterations, operation point variations, measurement imprecision, noise, unforeseen perturbations and the overall present nonlinearities and non-convexities[1, 2]. Trying to control these aspects through conventional

Á. Herrero et al. (eds.), *International Joint Conference SOCO'13-CISIS'13-ICEUTE'13*, Advances in Intelligent Systems and Computing 239,
DOI: 10.1007/978-3-319-01854-6_8,

hard-computing methods usually will increase the solution complexity up to unviable levels, forcing the designer to accept assumptions and simplifications on the control problem. Therefore, the control problem transformations –which are needed to solve the problemby conventional hard-computing methods in reasonable sampling times - can ensure stability on the closed-loop response under some assumptions, but can lose optimality in their response when variable and frequent changes in the system operation point are required. All these aspectsare likely to deteriorate the final result and could impede the development of an enhanced controller which would provide added value to the final control system.

NN and FL have already been implemented on embedded platforms for real-time control applications, often also in hybrid algorithms combining them witheach other (e.g. Neuro-Fuzzy) or with classical control techniques like PI controllers[3–6].

GAs are also well known for being able to offer great results when correctly tuned, but unlike the previous cases,they are usually used for offline optimization of a wide variety of designs and parameter determination[7, 8]. It is not likely to see GAs running on an embeddedtarget platforms, and when done so, they typically control slow systemsor optimize parameters of another controller according to variations [9–13].

One of the main drawbacks for the application of GAs and Evolutionary Algorithms(EA) on real-time control applicationsis their high computational cost, their complexity and their variability, which may also make it difficult to guarantee the searching convergence, i.e. the control action convergence inside the sampling time window.Additionally, ensuringstability of the closed-loop response becomes an arduous task (but however a challenge)that should be done by using Finite Markov Chain analysis[14]. These difficulties, together with the fact of lower demand for complex optimization methods running on targets, have prevented them from moving onto embedded platforms, hindering the appearance of new applications.

The core of this work is the MOGA which is desired to be proven as feasible for fast real-time intelligent control applications under the umbrella of a nonlinear model predictive strategy [15]. A NN for supporting the evaluation function by predicting the response of the control systems in a short horizon is also implemented. The aim of our research is to contribute to the applicability of intelligent & hybrid control schemes - using soft-computing techniques – but considering constricted electronic control units based on μC or digital signal processors (DSP).

2 A Real-Time MOGA Based Hybrid Controlscheme

GAs are stochastic optimization algorithms inspired in evolutionary genetics and natural selection, in which a population of solutions evolves through various generations under the influence of randomness and specimen survival, reproduction and mutation [16, 17].The resultisa method that handles very well nonlinear and complex systems, does not require excessive model simplifications and has less tendency to get stuck in local optima as happens with conventional methods, therefore being very successful in finding optimum or near-optimum solutions. Nevertheless, it is an iterative method in whicha series of operators have to be applied to a population of individuals (candidates) that also need to be evaluated at each iteration, which means the

computational complexity is a problem. The application of GAs over multiobjective optimization problems leads to MOGA techniques and methods where fitness functions with multiple objectives are considered[18, 19].

For the implementation approach presented in this work, the well-knownfast and elitist MOGA algorithm NSGA-II (Nondominated Sorting Genetic Algorithm) has been selected[20], see Fig. 1. Apart from its high relevancy in the current state of the art, its reasonable and foreseeable computational cost together with its fast convergence propertymaintaining a good spread of solutions makes it very suitable for working as solver in real-time optimization problems.The algorithm complexity of the NSGA-II is given by the expression$O(MN^2)G$, where Mstands for the amount of objectives, $N$for the population size and $G$for the number of generations. According to the previous expression the time T required to execute the searching procedure is given by $T = C(MN^2)G$where the factor C represents all the ignored aspects after using the asymptotic approach,O, which in this case is dependant of the platform performance, the fitness functions used and the issues related to the programming and implementation of the algorithm. As the impact of the population number N is squared, this variable should be tried to be kept low, thusimproving the searching results and convergence by adequately tuning the MOGA operators and the number of generations required, as far as the characteristics of the problem to be solved allow it.

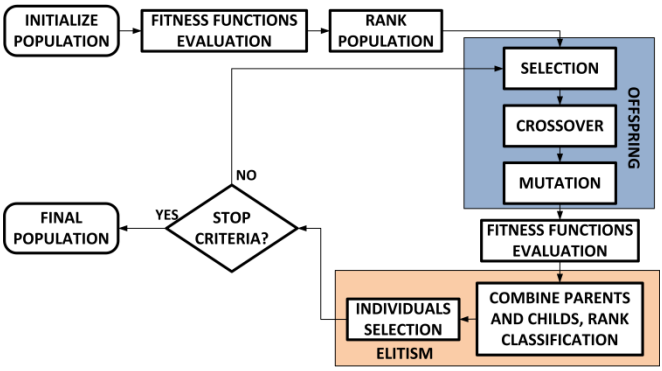

Fig. 1. Diagram representing the NSGA-II Multi Objective Genetic Algorithm

The interesting concept of using a MOGA optimizer as main controller and supporting it on NN based predictions as fitness functions is a feasible idea for handling the multiobjective nonlinear model predictive control [15]. Further enhancement of the controller can be obtained by implementing a final solution selection mechanism (among the Pareto's set) based on FL or Expert Systems as proposed in[15], see Fig. 2. Both mechanisms (NN and FL) can be implemented in an efficient and uncomplicated manner on µC platforms.

The implementation of the NN is a very critical point that must be implemented with high efficiency, as it needs to be executed hundreds or thousands of times per second, therefore consuming a considerable part of the total computation time.

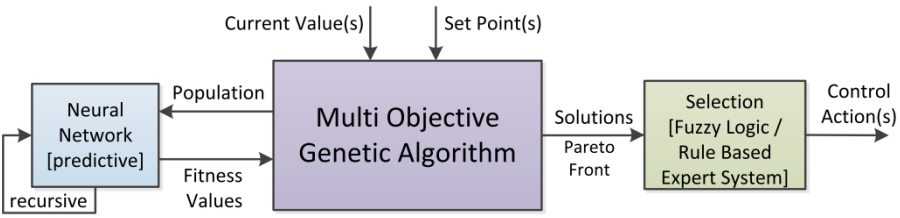

Fig. 2. The proposed Real-Time MOGA based Intelligent Hybrid scheme

Supporting the previous concept by transferring the task of executing the NN to a FPGA platform is a very interesting approach, due to their parallel processing capabilities with potential to rapidly run predictions of various solutions simultaneously [21]. The need for this hybrid hardware architecture solution will depend on the complexity of the NN and the time and cost boundary conditions.

Another interesting approach is the usage of downsized GAs. Micro-GAs offer a lightweight alternative for cases where higher computational costs are not acceptable[22]. They represent an optimization solution able to offer a compromise between results and time constraints which might be adequate in certain applications. Also in this direction, techniques based on small population combining Particle Swarm Optimization with evolutionary mechanisms have been proposed[23]. When adequately set up and tuned, which usually is a delicate task in GAs, smaller and simpler algorithms can also provide satisfying results.

3 Microcontroller Implementation

For the NSGA-II implementation and testing, a high performance floating-pointµC was selected(ARM® Cortex™-R4F - 180 MHz - 3 MB of flash memory - 256 KB RAM). This is one of the µCs that were considered as interesting – in addition to its processing capabilities and performance -because it is available as a redundant dual-core controller conceived,among others, for safety critical control functions, as it occurs in many automotive applications. This makes it well suited for promising application fields in the transportation sector, such as the relevant trend of Advanced Driver Assistance Systems(ADAS). However, other µCsand DSPs offering performances up to GHz range and parallel processing can be candidates forapplications in which higher data processing speeds are required[24].

An important consideration for the software development was to avoid bounding the solution to a specific target platform and/or software development tool which could create undesirable dependencies and workflow limitations. Therefore a simple programmingscheme has been selectedin order to provide a high code portability and flexibility to work with other platforms and algorithms. A single primary task executes the algorithm on a timer-based determinist and time-constrained loop.

The programming abstraction has been kept low: it is written in C and no RTOS has been used.Algorithms coming from a code generation tool canalso be easily integrated.

During the adaptation of the original code into embedded platform friendly C, diverse functions were suppressed or substituted, for example those regarding to interface, visualization and storage operations. Programming optimizations such as reducing data type casting operations and avoiding unnecessary intermediate results have also been applied. Regarding to the compiler, optimizations affecting registers, local code and global code were selected, together with function inlining and the most speed oriented speed-VS-code-size balance.

Regarding to the evaluation function, different approaches were taken. First two non-representative multiobjective optimization problems were defined: simple (1) and heavy (2) cost functions. Each one is composed by two fitness (objective) functions (f_0, f_1) with different mathematical complexity.

$$f_0 = x_0^2 + 5x_0x_1^2 + 4x_2 + {}^{x_3x_4}/_{x_5} \quad ; \quad f_1 = x_5^4 + 8x_2x_3^2 + 9x_1 + {}^{x_0x_1}/_{x_2} \tag{1}$$

$$f_0 = e^{\sqrt{x_0^2+5x_0x_1^2}/x_5} + \sqrt{x_2^2 + \sqrt{x_3x_4}} \cdot \left(\frac{x_5}{123456.789}\right)^5 \tag{2}$$

$$f_1 = e^{543212345\,/x_1\sqrt{x_5^2+5x_4^2/x_3}} + \sqrt{x_0^2 + \sqrt{x_4x_5}} \cdot {x_4}^{1/x_5}$$

A function which represents a practical problem was also tested: the model of a helicopter-like Twin Rotor MIMO System (Feedback Instruments Ltd.). This system has two voltage inputs to control the power of each of the two rotors which are rotated 90º respect to each other. As the structure is fixed to the ground through an articulation, it offers 2 degrees of freedom: the outputs are the pitch and the yaw angles.

This system is represented not by simple mathematical expressions, but bymore complex functions for which two different approaches have been implemented. In both of them*H*recursive function calls are executed for predicting the response over the prediction horizon.There are two objectives and their functions are chosen to be the calculation of the quadratic error over H regarding to the pitch and yaw angle setpoints. This can be easily extended by, for example, adding the control action energy consumption. The MOGA chromosome contains 10 real variables, as the MOGA has to obtain 2 control actions over a selected horizon of $H = 5$ steps-ahead.

The first approachis a relatively complex nonlinear Simulink™ model which represents the differential equations and was translated into a C function using code-generation. It contains 3 transfer functions, 4 integrators and a series of nonlinear mathematical expressions andtrigonometric functions.

The second and especiallyrelevant approach, which leads to the interesting hybrid concept with a NN, isimplemented instead of the previous model in order to be trained with the real system inputs and outputs so that it better reflects the non linearities. It is based on a NARX topology containing a layer with 8 neurons and a hidden layer with 12 neurons, plus the corresponding input and output layers. It is also integrated into the recursive execution based predictive context.

Inevitably the µC implementation must be adapted to the specific application considering the control specifications and requirements. This not only does mean optimizing the size and complexity of the algorithm itself, but also the required range and resolution of the variables being used. Once theplatform is established, platform specific low level coding must be considered in order to optimize the execution performance.Especially on routines that are executed many times per iteration- such as the fitness function or many of the evolutionary operations in EAs or GAs - should be taken care to be efficient, as they represent a significant fraction of the overall computation time. On high performance devices, the correct usage of resources such as the cache and the pipeline, and when possible executing more than one instruction per cycle(or even fullparallel processing)should beimplemented. Combining the previous actions with compiler/linker level optimizations and other actions, such as function inlining, significant performance improvements are obtained.

Aiming to test and analyze different soft-computing algorithms and intelligent control strategies, several systems would be needed. As setting up physical models would suppose an excessive effort, simulating the corresponding plant models on a parallel hardware platform was chosen as an agile, cost effective and safe alternative using the Hardware in the Loop (HiL) testing approach, see Fig. 3. For the first tests, a cost effective 32 bit floating point DSC running at 150 MHz was selected.It offers the possibility of directly embedding the automatically generated code from Simulink™ models, and additionally monitor and interact with them through a PC based GUI. It will enable to run the plant model in real-time with differential integration steps between one and three orders of magnitude faster than the controller sample time.

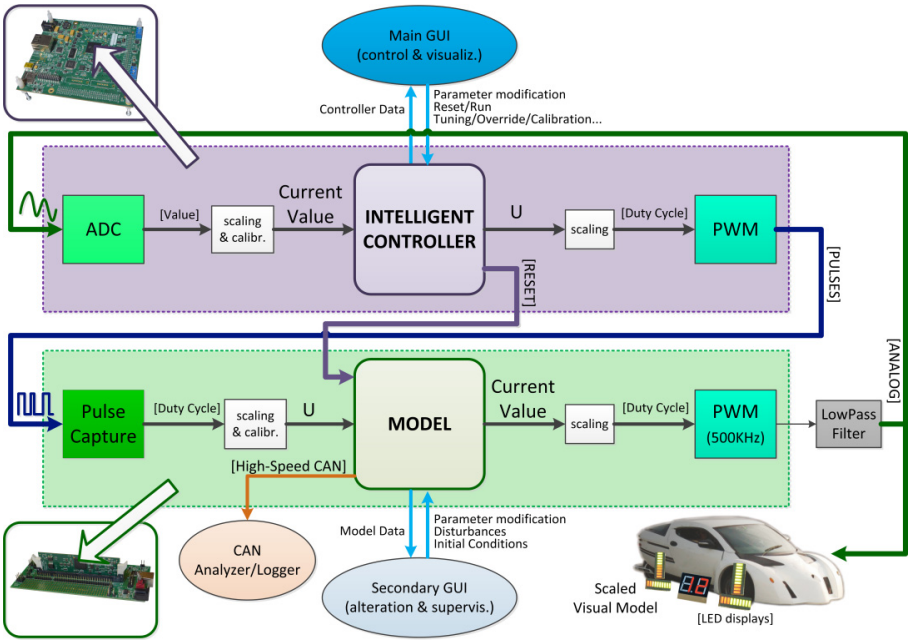

Fig. 3. HiL based Testing Setup. ARM® Cortex™-180 MHz as Intelligent Controller; DSC-150 MHz as Emulator of the system to be controlled

4 Results

As it was expected, the results obtained from this implementation have shown that MOGAsare computationally costly compared to other algorithms such as FL, NN or PID controllers, of which some simple versions were also implemented in a separate experiment for testing and comparative purposes, with resulting times in the units to hundreds of μs range. More specifically, the NN implemented for the hybrid-predictive-fitness scheme averaged at 34.7μswith its NARX topology and 8 neurons in the main layer and 12 neurons in the hidden layer.

The numbers corresponding to the most representative cases of the MOGA are presented in Table 1.A *big* and a *small*MOGA were adequately dimensioned thinking of complex and simpler problems, whereas the MOGA for the two rotor system was adjusted to allow a fast sample rate of 100mswith a prediction horizon of 5 steps.

Table 1. Summary of the most significant results of the MOGA implementation

	Big MOGA test		**Small MOGA test**		**Twin Rotor test**	
Fitness function	**Simple(1)**	**Heavy(2)**	**Simple(1)**	**heavy(2)**	**Nonlinear Model**	**NN**
Objectives	2	2	2	2	2	2
Population	**60**	**60**	**15**	**15**	**15**	**15**
Generations	**60**	**60**	**20**	**20**	**24**	**24**
Chromosome size	6	6	6	6	10	10
Prediction horizon	**-**	**-**	**-**	**-**	**5**	**5**
Constraints	-	-	-	-	-	-
Code size	79.0 KB	79.7KB	78.4 KB	79.2 KB	101.8 KB	104.3 KB
RAM occupation	85.5 KB	85.5 KB	11.6 KB	11.6 KB	14.9 KB	13.5KB
T_{cycle}Avg./Worst [ms]	**922**/970	**998**/1085	**21.5**/24.2	**28.1**/32.3	**74.6**/82.1	**88.1**/96.1
C factor (see section 2)	0.002134	0.002310	0.002389	0.003122	0.006907	0.008157
t_{eval}Average	0.335 μs	21.8 μs	0.335 μs	21.8 μs	136.2 μs	173.5 μs
C' factor(see eq. 3)	0.002130	0.002130	0.002378	0.002378	0.002371	0.002371

It can be observedthat more RAM than Flash memory is needed, in spite of the speed oriented and flash consuming optimization settings. More generations barely increase Flash nor RAM usage, but a bigger population once again dramatically increases the RAM consumption: without including the Twin Rotor nonlinear model (or the NN), the maximum possible population is 113.

Regarding to the cycle times - the really critical point - the nonlinear model and NN based predictive evaluation functions are considerably more costly that the *Simple* and *Heavy* mathematical expressions (eqs. 1 and 2). Nevertheless, as the most significant part of the computation time is still taken by the MOGA itself, the relative growth of the total computation time is still acceptable.It can be also noted that, although the model is slightly faster than the NN in this implementation, it has some simplifications so that the NN might be able to reflect nonlinearities better.

A major advantage of having a generic expression of the computational cost is the fact that instead of relying on large tables constructed through extensive testing, the execution time of a MOGA with some specific parameters can be extrapolated with sufficient precision basing on a reduced set of test results and the expression shown in section 2: $T = C(MN^2)G$. The resulting C values obtained are shown in Table 1. The difference between C valuesis due to the fact that the N^2 factor applies to the evolutionary algorithm, but the fitness function evaluation is only called $M \cdot N \cdot G$ times. Therefore this should be considered if the execution cost of the algorithm as a whole, including the fitness functions, is to be considered.

Consequently the following new expression is proposed:

$$T = C'(MN^2)G + C'(NF)G \;\rightarrow\; \boldsymbol{T = C'[N[MN + F)]G} \tag{3}$$

Being $F = t_{eval}/C$and $t_{eval}$the total computation time of one fitness function call (including all the objectives and therefore not$M$ dependant in this representation). Here the C'value corresponds to the MOGA running without any fitness evaluation.

In this way, knowing the computation time of the fitness expression (which can be relatively easily obtained in diverse manners) and the C' value which reflects the platform/implementation performance, the computation time of a MOGA of any desired population, generation and objective number can be easily and reliably obtained.

A small difference between the new C' values can still be seen, which is explained by the fact that there are a series of operations in the loops that introduce offsets and different dependencies to the M, N, G magnitudes, hereby slightly distorting the relation. Their values can be obtained by a combination of code analysis and measurements, but it turns into anunnecessarilycomplex analysis.

The results obtained in this µC implementation may be compared with those obtained in a NSGA-II implementation on industrial systems by Larzabal et al. [25]. This was done on industrial platforms with much higher performance, size and cost. The resulting cycle time for a 1 GHz Intel® Celeron® PLC was under 0.15s for a population of 70, 100 generations and 1 objective. This gives $C = 0.00031$. Faster-platforms obviously provide a lower C number due to their higher operating frequency. Therefore representing these values in a frequency independent unit could provide an interesting benchmark. Calculating $B = C \times f_{processor}(MHz)$ the result is $B = 0.38 \sim 0.43$ for the 180 MHz ARM® based µC and $B = 0.31$ for the 1 GHz Industrial PLC platform. The overall performance of the PLC is better, in spite of the additional cost of running a RTOS, as it benefits from higher processing capacity and other advantages such as cache and faster memory.

5 Conclusions and Future Work

The experimental results obtained in this work prove that the idea of embedding a NSGA-II MOGA on a reasonably priced µC - which hasconsiderable restrictions in terms of computational power and memory - is possible and reasonable. Therefore, this contribution is meant to support the future use of MOGAs not only for off-line optimization, but also as main controller in fast real-time control systems.

After a first phase where rather simple mathematical equations with reduced capacity to represent real world systems were used as fitness function, more complex nonlinear models have been introduced in order toenable to control dynamic systems,based on a recursive predictive strategy. Anonlinear model with differential equation systemswas inserted into this scheme. Finally, following the hybrid soft-computing concept,a NN which can representcomplex nonlinearities with considerable accuracy was also implemented into the predictive structure.

The resulting cycle times are reasonably low and encouragingto follow this line and exploit its potential, as they enable the application of complex intelligent control methods on relatively fast dynamic systems.

Sufficient degrees of freedom are still available to improve the application of complex soft-computing algorithms and techniques, thus enabling the development of new systems and control concepts. Fine algorithm tuning and the code optimization for specific μCs or DSPs could still push the overall performance.

A new and more detailed expression to predict - in a reliable and simple manner - the total computation time for the MOGA has also been presented. In this formula the influence of the fitness functions time cost has been treated independently and included as a new term in the expression.

Future work will develop a more refined implementation and a more exhaustive study of alternative solutions regarding both to the algorithmia and the μC or DSP based platforms. Following the intelligent control research line, next works will also focus on hybrid intelligent controllers' implementation and their validation through the HiL setup. Transferring time consuming parallel operations to a FPGA to optimize the controller implementation will also be investigated.

References

1. Rudas, I.J., Fodor, J.: Intelligent Systems. International Journal of Computers, Communications and Control 3(Spl. Iss.), 132–138 (2008)
2. Zadeh, L.A.: Fuzzy Logic, Neural Networks, and Soft Computing. Communications of the ACM 37(3), 77–84 (1994)
3. Del Campo, et al.: Efficient Hardware/Software Implementation of an Adaptive Neuro-Fuzzy System. IEEE Transactions on Fuzzy Systems 16(3), 761–778 (2008)
4. Velagic, et al.: Microcontroller Based Fuzzy-PI Approach Employing Control Surface Discretization. In: 2012 20th Mediterranean Conference on Control Automation, MED, Piscataway, NJ, USA, pp. 638–645 (2012)
5. Jung, S., Su Kim, S.: Hardware Implementation of a Real-Time Neural Network Controller With a DSP and an FPGA for Nonlinear Systems. IEEE Transactions on Industrial Electronics 54(1), 265–271 (2007)
6. Ravi, S., Sudha, M., Balakrishnan, P.A.: Design and Development of a Microcontroller Based Neuro Fuzzy Temperature Controller. In: 2012 International Conference on Informatics, Electronics &Vision (ICIEV), Piscataway, NJ, USA, pp. 103–107 (2012)
7. Yousefpoor, et al.: THD Minimization Applied Directly on the Line-to-Line Voltage of Multilevel Inverters. IEEE Transactionson Industrial Electronics 59, 373–380 (2012)
8. Fleming, P., Purshouse, R.: Evolutionary Algorithms in Control Systems Engineering: a Survey. Control Engineering Practice 10(11), 1223–1241 (2002)

9. Mamdoohi, et al.: Realization of Microcontroller-Based Polarization Control System with Genetic Algorithm. In: 2009 IEEE 9th Malaysia International Conference on Communications (MICC), Piscataway, NJ, USA, pp. 774–779 (2009)
10. Krishnan, et al.: Parallel Distributed Genetic Algorithm Development Based on Microcontrollers Framework. In: First International Conference on Distributed Framework and Applications, DFmA 2008, Piscataway, NJ, USA, pp. 35–40 (2008)
11. Mininno, E., et al.: Real-Valued Compact Genetic Algorithms for Embedded Microcontroller Optimization. IEEE Transaction on Evolutionary Computing 12, 203–219 (2008)
12. Cao, et al.: DSP Implementation of the Particle Swarm and Genetic Algorithms for Real-Time Design of Thinned Array Antennas. IEEE Antennas and Wireless Propagation Letters 11, 1170–1173 (2012)
13. Kwak, M., Shin, T.S.: Real-Time Automatic Tuning of Vibration Controllers for Smart Structures by Genetic Algorithm. In: Proc. of SPIE, USA, vol. 3667, pp. 679–690 (1999)
14. Goldberg, D., Segrest, P.: Finite Markov Chain Analysis of Genetic Algorithms. In: Proceedings of the Second International Conference on Genetic Algorithms and their Application, pp. 1–8. L. Erlbaum Associates Inc., Hillsdale (1987)
15. Valera Garcia, J.J., et al.: Intelligent Multi-Objective Nonlinear Model Predictive Control (iMO-NMPC): Towards the 'On-line' Optimization of Highly Complex Control Problems. Expert Systems with Applications 39(7), 6527–6540 (2012)
16. Holland, J.H.: Adaptation in Natural and Artificial Systems:. Univ. Michigan Press (1975)
17. Goldberg, D., Holland, J.: Genetic Algorithms and Machine Learning. Machine Learning 3(2-3), 95–99 (1988)
18. Coello Coello, C.A.: Evolutionary Multi-Objective Optimization: a Historical View of the Field. IEEE Computational Intelligence Magazine 1(1), 28–36 (2006)
19. Konak, et al.: ulti-Objective Optimization using Genetic Algorithms: A tutorial. Reliability Engineering & System Safety 91(9), 992–1007 (2006)
20. Deb, et al.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
21. Martin, P.: A hardware implementation of a genetic programming system using FPGAs and Handel-C. Genetic Programming and Evolvable Machines 2(4), 317–343 (2001)
22. Toscano Pulido, G., Coello Coello, C.A.: The Micro Denetic Algorithm 2: Towards Online Adaptation in Evolutionary Multiobjective Optimization. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 252–266. Springer, Heidelberg (2003)
23. Cabrera, J.C.F., Coello Coello, C.A.: Micro-MOPSO: A Multi-objective Particle Swarm Optimizer that uses a very small Population Size. In: Nedjah, N., dos Santos Coelho, L., de Macedo Mourelle, L. (eds.) Multi-Objective Swarm Intelligent Systems. SCI, vol. 261, pp. 83–104. Springer, Heidelberg (2010)
24. Li, Q., He, J.: A sophisticated architecture for evolutionary multiobjective optimization utilizing high performance dsp. In: Kang, L., Liu, Y., Zeng, S. (eds.) ICES 2007. LNCS, vol. 4684, pp. 415–425. Springer, Heidelberg (2007)
25. Larzabal, E., Cubillos, J.A., Larrea, M., Irigoyen, E., Valera, J.J.: Soft computing Testing in Real Industrial Platforms for Process Intelligent Control. In: Snasel, V., Abraham, A., Corchado, E.S. (eds.) SOCO Models in Industrial & Environmental Appl. AISC, vol. 188, pp. 221–230. Springer, Heidelberg (2013)