UTILIZATION OF ANALYTIC PROGRAMMING FOR THE STABILIZATION OF HIGH ORDER OSCILLATIONS OF CHAOTIC LOGISTIC EQUATION

Roman Senkerik[(a)], Zuzana Oplatkova[(a)], Ivan Zelinka[(b)], Donald Davendra[(b)], Michal Pluhacek[(a)]

[(a)]Tomas Bata University in Zlin , Faculty of Applied Informatics, Nam T.G. Masaryka 5555, 760 01 Zlin, Czech Republic
[(b)] Technical University of Ostrava, Faculty of Electrical Engineering and Computer Science, 17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic

[(a)]senkerik@fai.utb.cz, [(b)]ivan.zelinka@vsb.cz

ABSTRACT
This research deals with a utilization of a modern tool for symbolic regression, which is analytic programming, for the purpose of the evolutionary synthesis of a feedback controller for the chaotic system. This synthesized chaotic controller secures the stabilization of high periodic orbit – oscillations between several values of discrete chaotic system, which is Logistic Equation. The paper consists of the descriptions of analytic programming as well as chaotic system, used heuristic and cost function. For experimentation, Self-Organizing Migrating Algorithm (SOMA) and Differential evolution (DE) were used.

Keywords: Chaos Control, Analytic programming, optimization, evolutionary algorithms.

1. INTRODUCTION

During the past five years, usage of new intelligent systems in engineering, technology, modeling, computing and simulations has attracted the attention of researchers worldwide. The most current methods are mostly based on soft computing, which is a discipline tightly bound to computers, representing a set of methods of special algorithms, belonging to the artificial intelligence paradigm. The most popular of these methods are neural networks, evolutionary algorithms, fuzzy logic, and genetic programming. Presently, evolutionary algorithms are known as a powerful set of tools for almost any difficult and complex optimization problem.

The interest about the interconnection between evolutionary techniques and control of chaotic systems is spread daily. First steps were done in (Senkerik et al.; 2010a), (Zelinka et al., 2009), where the control law was based on Pyragas method: Extended delay feedback control – ETDAS (Pyragas, 1995). These papers were concerned to tune several parameters inside the control technique for chaotic system. Compared to previous research, this paper shows a possibility how to generate the whole control law (not only to optimize several parameters) for the purpose of stabilization of a chaotic system. The synthesis of control is inspired by the Pyragas's delayed feedback control technique (Just, 1999), (Pyragas, 1992). Unlike the original OGY control method (Ott et al., 1990), it can be simply considered as a targeting and stabilizing algorithm together in one package (Kwon, 1999). Another big advantage of the Pyragas method for evolutionary computation is the amount of accessible control parameters, which can be easily tuned by means of evolutionary algorithms (EA).

Instead of EA utilization, analytic programming (AP) is used in this research. AP is a superstructure of EAs and is used for synthesis of analytic solution according to the required behaviour. Control law from the proposed system can be viewed as a symbolic structure, which can be synthesized according to the requirements for the stabilization of the chaotic system. The advantage is that it is not necessary to have some "preliminary" control law and to estimate its parameters only. This system will generate the whole structure of the law even with suitable parameter values.

This work is focused on the expansion of AP application for synthesis of a whole control law instead of parameters tuning for existing and commonly used method control law to stabilize desired Unstable Periodic Orbits (UPO) of chaotic systems.

This work is an extension of previous research (Oplatkova et al., 2010a; 2010b), (Senkerik et al., 2010b) focused on stabilization of simple p-1 orbit – stable state and p-2 orbit. In general, this research is concerned to stabilize p-4 UPO – high periodic orbit (oscillations between four values).

Firstly, AP is explained, and then a problem design is proposed. The next sections are focused on the description of used cost function and evolutionary algorithms. Results and conclusion follow afterwards.

2. PROBLEM DESIGN

The brief description of used chaotic systems and original feedback chaos control method, ETDAS is given. The ETDAS control technique was used in this research as an inspiration for synthesizing a new feedback control law by means of evolutionary techniques.

2.1. Selected Chaotic System

The chosen example of chaotic systems was the one-dimensional Logistic equation in form (1).

$$x_{n+1} = rx_n(1 - x_n) \tag{1}$$

The Logistic equation (logistic map) is a one-dimensional discrete-time example of how complex chaotic behaviour can arise from very simple non-linear dynamical equation (Hilborn, 2000). This chaotic system was introduced and popularized by the biologist Robert May (May, 2001). It was originally introduced as a demographic model as a typical predator – prey relationship. The chaotic behaviour can be observed by varying the parameter *r*. At $r = 3.57$ is the beginning of chaos, at the end of the period-doubling behaviour. At $r > 3.57$ the system exhibits chaotic behaviour. The example of this behaviour can be clearly seen from bifurcation diagram – Figure 1.

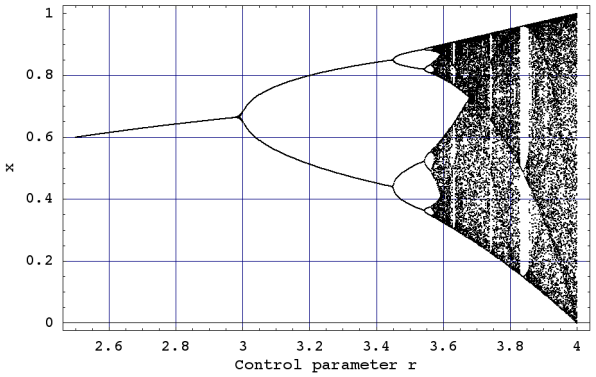

Figure 1: Bifurcation diagram of Logistic Equation

2.2. ETDAS Control Method

This work is focused on explanation of application of AP for synthesis of a whole control law instead of demanding tuning of EDTAS method control law to stabilize desired Unstable Periodic Orbits (UPO). In this research desired UPO is only p-2 (higher periodic orbit – oscillation between two values). ETDAS method was obviously an inspiration for preparation of sets of basic functions and operators for AP.

The original control method – ETDAS has form (2).

$$\begin{aligned} F(t) &= K\left[(1-R)S(t-\tau_d) - x(t)\right] \\ S(t) &= x(t) + RS(t-\tau_d) \end{aligned} \tag{2}$$

Where: *K* and *R* are adjustable constants, *F* is the perturbation; *S* is given by a delay equation utilizing previous states of the system and τ_d is a time delay.

The original control method – ETDAS in the discrete form suitable for one-dimensional Logistic equation has the form (3).

$$\begin{aligned} x_{n+1} &= rx_n(1 - x_n) + F_n \\ F_n &= K\left[(1-R)S_{n-m} - x_n\right] \\ S_n &= x_n + RS_{n-m} \end{aligned} \tag{3}$$

Where: *m* is the period of *m*-periodic orbit to be stabilized. The perturbation F_n in equations (3) may have arbitrarily large value, which can cause diverging of the system outside the interval $\{0, 1.0\}$. Therefore, F_n should have a value between $-F_{\max}$, $F_{\max}$. In this preliminary study a suitable $F_{\max}$ value was taken from the previous research. To find the optimal value also for this parameter is in future plans.

Previous research concentrated on synthesis of control law only for p-1 orbit (a fixed point). An inspiration for preparation of sets of basic functions and operators for AP was simpler TDAS control method (4) and its discrete form suitable for logistic equation given in (5).

$$F(t) = K\left[x(t-\tau) - x(t)\right] \tag{4}$$

$$F_n = K\left(x_{n-m} - x_n\right) \tag{5}$$

2.3. Cost Function

Proposal for the cost function comes from the simplest Cost Function (CF). The core of CF could be used only for the stabilization of p-1 orbit. The idea was to minimize the area created by the difference between the required state and the real system output on the whole simulation interval – τ_i.

But another universal cost function had to be used for stabilizing of higher periodic orbit and having the possibility of adding penalization rules. It was synthesized from the simple CF and other terms were added. In this case, it is not possible to use the simple rule of minimizing the area created by the difference between the required and actual state on the whole simulation interval – τ_i, due to many serious reasons, for example: degrading of the possible best solution by phase shift of periodic orbit.

This CF is in general based on searching for desired stabilized periodic orbit and thereafter calculation of the difference between desired and found actual periodic orbit on the short time interval - τ_s (40 iterations) from the point, where the first min. value of difference between desired and actual system output is found. Such a design of CF should secure the successful stabilization of either p-1 orbit (stable state) or higher periodic orbit anywise phase shifted. The CF_{Basic} has the form (6).

$$CF_{Basic} = pen_1 + \sum_{t=\tau 1}^{\tau 2} \left| TS_t - AS_t \right|, \tag{6}$$

where:
TS - target state, AS - actual state
τ_1 - the first min value of difference between TS and AS
τ_2 – the end of optimization interval ($\tau_1 + \tau_s$)
$pen_1 = 0$ if $\tau_i - \tau_2 \geq \tau_s$;
$pen_1 = 10*(\tau_i - \tau_2)$ if $\tau_i - \tau_2 < \tau_s$ (i.e. late stabilization).

3. ANALYTIC PROGRAMMING

Basic principles of the AP were developed in 2001 (Zelinka et al. 2005). Until that time only Genetic Programming (GP) and Grammatical Evolution (GE) had existed. GP uses Genetic Algorithms (GA) while AP can be used with any EA, independently on individual representation. To avoid any confusion, based on the nomenclature according to the used algorithm, the name - Analytic Programming was chosen, since AP represents synthesis of analytical solution by means of EA.

The core of AP is based on a special set of mathematical objects and operations. The set of mathematical objects is a set of functions, operators and so-called terminals (as well as in GP), which are usually constants or independent variables. This set of variables is usually mixed together and consists of functions with different number of arguments. Because of a variability of the content of this set, it is termed the "general functional set" – GFS. The structure of GFS is created by subsets of functions according to the number of their arguments. For example GFS_{all} is a set of all functions, operators and terminals, GFS_{3arg} is a subset containing functions with only three arguments, GFS_{0arg} represents only terminals, etc. The subset structure presence in GFS is vitally important for AP. It is used to avoid synthesis of pathological programs, i.e. programs containing functions without arguments, etc. The content of GFS is dependent only on the user. Various functions and terminals can be mixed together (Zelinka et al. 2005, Zelinka et al. 2008, Oplatkova et al. 2009).

The second part of the AP core is a sequence of mathematical operations, which are used for the program synthesis. These operations are used to transform an individual of a population into a suitable program. Mathematically stated, it is a mapping from an individual domain into a program domain. This mapping consists of two main parts. The first part is called Discrete Set Handling (DSH) (See Figure 2) (Zelinka et al. 2005, Lampinen and Zelinka 1999) and the second one stands for security procedures which do not allow synthesizing pathological programs.

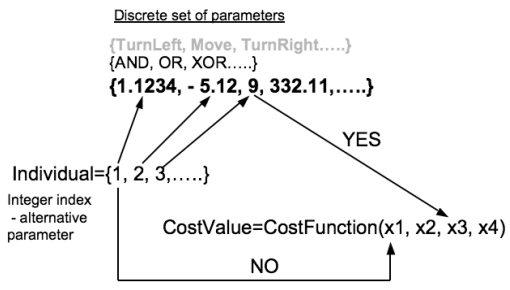

Figure 2: Discrete set handling

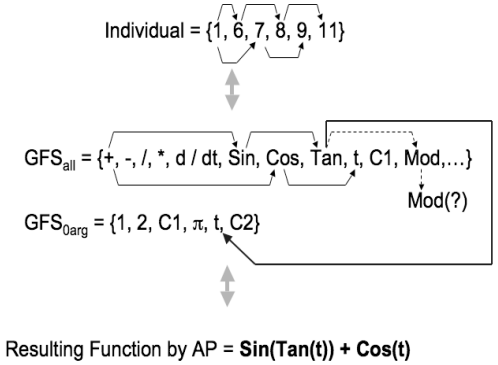

Figure 3: The main principles of AP

The method of DSH, when used, allows handling arbitrary objects including nonnumeric objects like linguistic terms {hot, cold, dark…}, logic terms (True, False) or other user defined functions. In the AP, DSH is used to map an individual into GFS and together with security procedures creates the above-mentioned mapping, which transforms arbitrary individual into a program.

AP needs some EA (Zelinka et al. 2005) that consists of a population of individuals for its run. Individuals in the population consist of integer parameters, i.e. an individual is an integer index pointing into GFS. The creation of the program can be schematically observed in Figure 3. The individual contains numbers which are indices into GFS. The detailed description is represented in (Zelinka et al. 2005, Zelinka et al. 2008, Oplatkova et al. 2009).

AP exists in 3 versions – basic without constant estimation, AP_{nf} – estimation by means of nonlinear fitting package in *Mathematica* environment and AP_{meta} – constant estimation by means of another evolutionary algorithms; meta implies metaevolution.

4. USED EVOLUTIONARY ALGORITHMS

This research used two evolutionary algorithms: Self-Organizing Migrating Algorithm (Zelinka 2004) and Differential Evolution (Price and Storn 2001, Price 2005). Future simulations expect a usage of soft computing GAHC algorithm (modification of HC12) (Matousek 2007) and a CUDA implementation of HC12 algorithm (Matousek 2010).

4.1. Self Organizing Migrating Algorithm – SOMA

SOMA is a stochastic optimization algorithm that is modelled on the social behaviour of cooperating individuals (Zelinka 2004). It was chosen because it has been proven that the algorithm has the ability to converge towards the global optimum (Zelinka 2004) and due to the successful applications together with AP (Varacha and Zelinka 2008, Varacha and Jasek 2011).

SOMA works with groups of individuals (population) whose behavior can be described as a competitive – cooperative strategy. The construction of a new population of individuals is not based on evolutionary principles (two parents produce offspring) but on the behavior of social group, e.g. a herd of

animals looking for food. This algorithm can be classified as an algorithm of a social environment. To the same group of algorithms, Particle Swarm Optimization (PSO) algorithm can also be classified sometimes called swarm intelligence. In the case of SOMA, there is no velocity vector as in PSO, only the position of individuals in the search space is changed during one generation, referred to as 'migration loop'.

The rules are as follows: In every migration loop the best individual is chosen, i.e. individual with the minimum cost value, which is called the Leader. An active individual from the population moves in the direction towards the Leader in the search space. At the end of the crossover, the position of the individual with minimum cost value is chosen. If the cost value of the new position is better than the cost value of an individual from the old population, the new one appears in new population. Otherwise the old one remains there. The main principle is depicted in Figures 4 and 5.

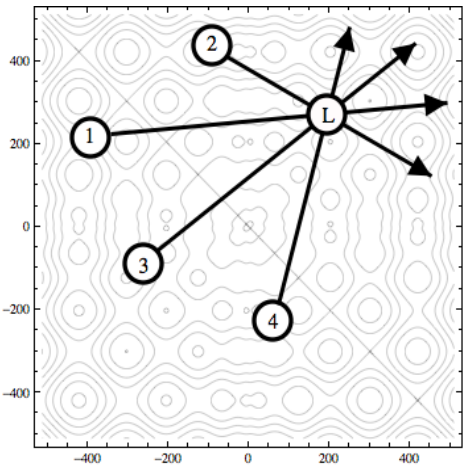

Figure 4: Principle of SOMA, movement in the direction towards the Leader

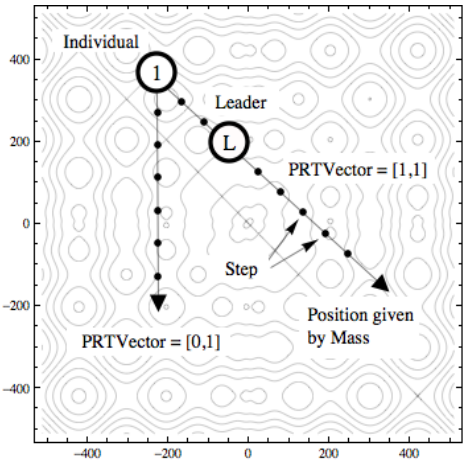

Figure 5: Basic principle of crossover in SOMA, PathLength is replaced here by Mass

4.2. Differential Evolution

DE is a population-based optimization method that works on real-number-coded individuals (Price, 2005). A schematic is given in Figure 6. There are essentially five sections to the code. Section 1 describes the input to the heuristic. D is the size of the problem, G_{max} is the maximum number of generations, NP is the total number of solutions, F is the scaling factor of the solution and CR is the factor for crossover. F and CR together make the internal tuning parameters for the heuristic.

Section 2 outlines the initialization of the heuristic. Each solution $x_{i,j,G=0}$ is created randomly between the two bounds $x^{(lo)}$ and $x^{(hi)}$. The parameter j represents the index to the values within the solution and i indexes the solutions within the population. So, to illustrate, $x_{4,2,0}$ represents the fourth value of the second solution at the initial generation.

After initialization, the population is subjected to repeated iterations in section 3.

Section 4 describes the conversion routines of DE. Initially, three random numbers r_1, r_2, r_3 are selected, unique to each other and to the current indexed solution i in the population in 4.1. Henceforth, a new index j_{rand} is selected in the solution. j_{rand} points to the value being modified in the solution as given in 4.2. In 4.3, two solutions, $x_{j,r1,G}$ and $x_{j,r2,G}$ are selected through the index r_1 and r_2 and their values subtracted. This value is then multiplied by F, the predefined scaling factor. This is added to the value indexed by r_3.

However, this solution is not arbitrarily accepted in the solution. A new random number is generated, and if this random number is less than the value of CR, then the new value replaces the old value in the current solution. The fitness of the resulting solution, referred to as a perturbed vector $u_{j,i,G}$., is then compared with the fitness of $x_{j,i,G}$. If the fitness of $u_{j,i,G}$ is greater than the fitness of $x_{j,i,G}$., then $x_{j,i,G}$. is replaced with $u_{j,i,G}$; otherwise, $x_{j,i,G}$. remains in the population as $x_{j,i,G+1}$. Hence the competition is only between the new *child* solution and its *parent* solution.

1. Input: $D, G_{\max}, NP \geq 4, F \in (0,1+), CR \in [0,1]$, and initial bounds: $\vec{x}^{(lo)}, \vec{x}^{(hi)}$.

2. Initialize: $\begin{cases} \forall i \leq NP \wedge \forall j \leq D : x_{i,j,G=0} = x_j^{(lo)} + rand_j[0,1] \bullet \left(x_j^{(hi)} - x_j^{(lo)}\right) \\ i = \{1,2,...,NP\},\ j = \{1,2,...,D\},\ G = 0,\ rand_j[0,1] \in [0,1] \end{cases}$

$\begin{cases} \text{3. While } G < G_{\max} \\ \quad \forall i \leq NP \begin{cases} \text{4. Mutate and recombine:} \\ \quad \text{4.1 } r_1, r_2, r_3 \in \{1,2,.....,NP\}, \text{ randomly selected, except: } r_1 \neq r_2 \neq r_3 \neq i \\ \quad \text{4.2 } j_{rand} \in \{1,2,....,D\}, \text{ randomly selected once each } i \\ \quad \text{4.3 } \forall j \leq D, u_{j,i,G+1} = \begin{cases} x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) \\ \qquad \text{if } (rand_j[0,1] < CR \vee j = j_{rand}) \\ x_{j,i,G} \quad \text{otherwise} \end{cases} \\ \text{5. Select} \\ \vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G+1} & \text{if } f(\vec{u}_{i,G+1}) \leq f(\vec{x}_{i,G}) \\ \vec{x}_{i,G} & \text{otherwise} \end{cases} \end{cases} \\ G = G + 1 \end{cases}$

Figure 6: DE Schematic

DE is quite robust, fast, and effective, with global optimization ability. It does not require the objective function to be differentiable, and it works well even

with noisy and time-dependent objective functions. Description of used DERand1Bin strategy is presented in (7). Please refer to (Price and Storn 2001, Price 2005) for the description of all other strategies. These strategies differ in the way of calculating the perturbed vector $u_{j,i,G}$.

$$u_{i,G+1} = x_{r1,G} + F \cdot \left(x_{r2,G} - x_{r3,G}\right) \quad (7)$$

5. SIMULATION RESULTS

As described in section about Analytic Programming, AP requires some EA for its run. In this paper AP_{meta} version was used. Meta-evolutionary approach means usage of one main evolutionary algorithm for AP process and second algorithm for coefficient estimation, thus to find optimal values of constants in the evolutionary synthesized control law.

SOMA algorithm was used for main AP process and DE was used in the second evolutionary process. Settings of EA parameters for both processes were based on performed numerous experiments with chaotic systems and simulations with AP_{meta} (Table 1 and Table 2).

Table 1: SOMA settings for AP

PathLength	3
Step	0.11
PRT	0.1
PopSize	50
Migrations	4
Max. CF Evaluations (CFE)	5345

Table 2: DE settings for meta-evolution

PopSize	40
F	0.8
CR	0.8
Generations	150
Max. CF Evaluations (CFE)	6000

Compared to previous research with stabilization of stable state - p-1 orbit, the data set for AP required only constants, operators like plus, minus, power and output values x_n and x_{n-1}. Due to the recursive attributes of delay equation *S* utilizing previous states of the system in discrete ETDAS (3), the data set for AP had to be expanded and cover longer system output history, thus to imitate inspiring control method for the successful synthesis of control law securing the stabilization of higher periodic orbits.

Basic set of elementary functions for AP:

GFS2arg= +, -, /, *, ^
GFS0arg= $data_{n-11}$ to $data_n$, K

Total number of 200 simulations was carried out. The most simulations were successful and have given new synthesized control law, which was able to stabilize the system at required behaviour (p-4 orbit) within short simulation interval of 200 iterations.

Total number of cost function evaluations for AP was 5345, for the second EA it was 6000, together 32.07 millions per each simulation. All experiments were performed in the Wolfram Mathematica environment. One experiment (simulation) took approx. 72 hours. See Table 3 for simple CF values statistic.

Table 3: Cost Function values

Min	0.0314
Max	8.9088
Average	0.6178

The novelty of this approach represents the synthesis of feedback control law F_n (8) (perturbation) for the Logistic equation inspired by original ETDAS control method.

$$x_{n+1} = rx_n\left(1 - x_n\right) + F_n \quad (8)$$

Following Table 4 and Figure 7 contains examples of synthesized control laws. Obtained simulation results can be classified into 2 groups, based on the quality and durability of stabilization at real p-4 UPO, which for unperturbed Logistic equation has following values: $x_1 = 0.3038$, $x_2 = 0.8037$, $x_3 = 0.5995$, $x_4 = 0.9124$. More about this phenomenon is written in the conclusion section.

Table 4 covers direct output from AP – synthesized control law without coefficients estimated, further the notation with simplification after estimation by means of second algorithm DE, corresponding CF value, average error between actual and required system output, and identification of figure with simulation results.

Table 4: Simulation results

Control Law	Control Law with coefficients	CF Value	Avg. output error	Figure
$F_n = -\frac{x_{n-4}}{K_3\left(x_n^{K_2} - K_1\right)}$	$F_n = \frac{0.0200224 x_{n-4}}{x_n^{\frac{1}{50.5239}} - 92.1071}$	0.0314	0.0006	7a
$F_n = x_{n-2}^{x_n^{K_1}}$	$F_n = x_{n-2}^{x_n^{34.6383}}$	0.1007	0.0020	7b
$F_n = \left(x_{n-5}^{x_{n-2}} + x_{n-2}\right) x_{n-2}^{\left(x_n^{K_1} - x_{n-11}\right)}$	$F_n = \left(x_{n-5}^{x_{n-2}} + x_{n-2}\right) x_{n-2}^{\left(x_n^{\frac{1}{35.5805}} - x_{n-11}\right)}$	0.1139	0.0023	7c
$F_n = -\frac{x_{n-3}}{K_1 - x_{n-1}\left(\frac{x_{n-5} x_{n-1}}{K_3} - K_2 K_4 x_n\right)}$	$F_n = -\frac{x_{n-3}}{-x_{n-1}\left(0.017878 x_{n-5} x_{n-1} - 325.295 x_n\right) - 70.0758}$	0.4814	0.0096	7d

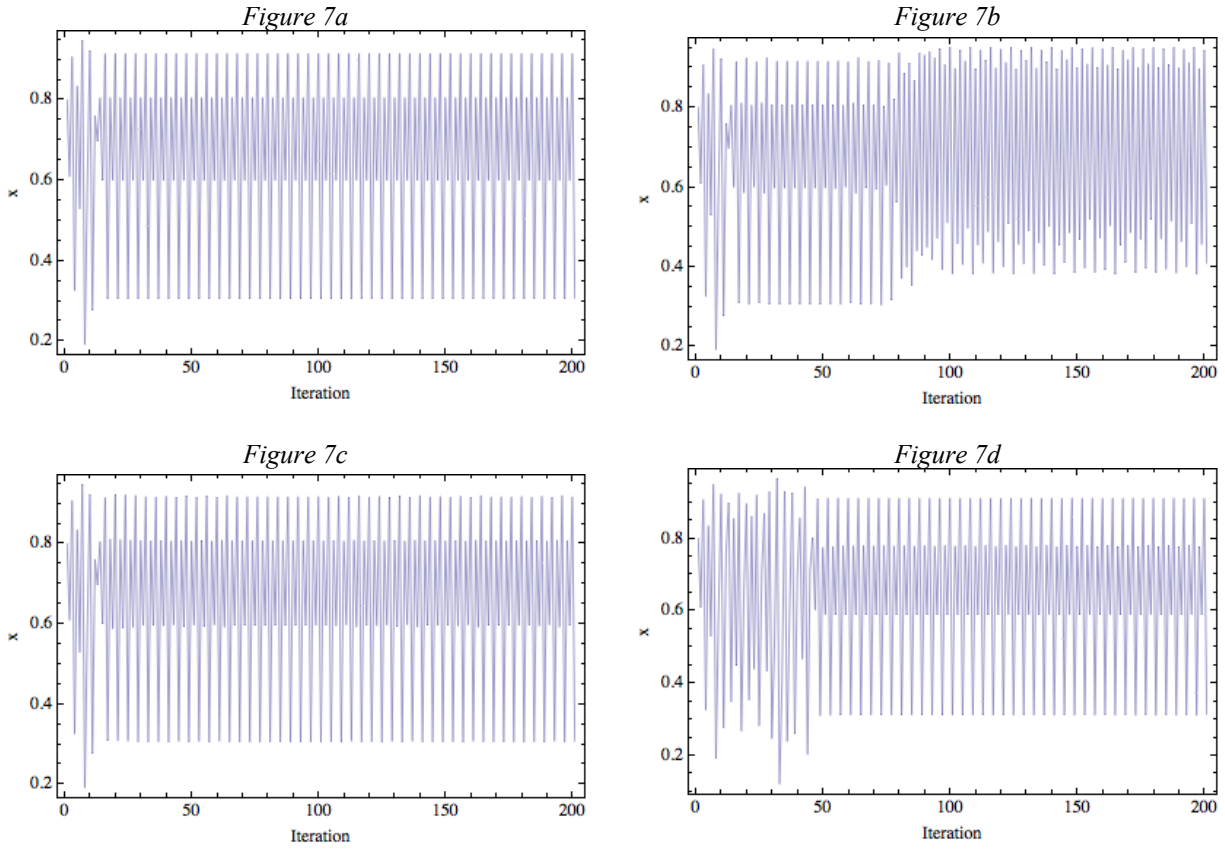

Figure 7: Examples of results – stabilization of p-4 orbit for Logistic equation by means of control laws given in Table 4.

6. CONCLUSION

This paper deals with a synthesis of a control law by means of AP for stabilization of selected chaotic system at high periodic orbit. Logistic equation as an example of one-dimensional discrete chaotic system was used in this research.

In this presented approach, the analytic programming was used instead of tuning of parameters for existing control technique by means of EA's as in the previous research.

Obtained results reinforce the argument that AP is able to solve this kind of difficult problems and to produce a new synthesized control law in a symbolic

way securing desired behaviour of chaotic system and stabilization.

Presented four simulation examples show two different results. Low CF values indicating precise, but unfortunately sometimes unstable and only temporary stabilization, together with simple control law in the first two cases. And according to the higher CF values not very precise, but very stable and relatively complex notation of chaotic controller in the next two cases. This phenomenon is caused by the design of CF, which was borrowed from the previous research focused on the simpler cases, which were stabilization of stable state and p-2 orbit, and it has given satisfactory results Nevertheless this fact lends weight to the argument, that AP is a powerful symbolic regression tool, which is able to strictly and precisely follow the rules given by cost function and synthesizes any symbolic formula, in the case of this research – the feedback controller for chaotic system.

The question of energy costs and more precise and faster stabilization will be included into future research together with development of better cost functions, different AP data set, and performing of numerous simulations to obtain more results and produce better statistics, thus to confirm the robustness of this approach.

Future research will be also aimed at the time-continuous systems, not only discrete chaotic maps.

ACKNOWLEDGMENTS

This work was supported by the European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089 and project IT4Innovations Centre of Excellence No. CZ.1.05/1.1.00/02.0070 and and by Internal Grant Agency of Tomas Bata University under the project No.IGA/FAI/2012/037.

REFERENCES

Hilborn R.C., 2000. Chaos and Nonlinear Dynamics: An Introduction for Scientists and Engineers, Oxford University Press, 2000, ISBN: 0-19-850723-2.

Just W., 1999, "Principles of Time Delayed Feedback Control", In: Schuster H.G., Handbook of Chaos Control, Wiley-Vch, ISBN 3-527-29436-8.

Kwon O. J., 1999. "Targeting and Stabilizing Chaotic Trajectories in the Standard Map", Physics Letters A. vol. 258, 1999, pp. 229-236.

Lampinen J., Zelinka I., 1999, "New Ideas in Optimization – Mechanical Engineering Design Optimization by Differential Evolution", Volume 1, London: McGraw-hill, 1999, 20 p., ISBN 007-709506-5.

Matousek R., 2007, „GAHC: Improved GA with HC station", In WCECS 2007, San Francisco, pp. 915-920. ISBN: 978-988-98671-6-4.

Matousek R., 2010, „HC12: The Principle of CUDA Implementation". In MENDEL 2010, Mendel Journal series, pp. 303-308. ISBN: 978-80-214-4120- 0. ISSN: 1803- 3814.

May R.M., 2001, "Stability and Complexity in Model Ecosystems", Princeton University Press, ISBN: 0-691-08861-6.

Oplatková, Z., Zelinka, I.: 2009. Investigation on Evolutionary Synthesis of Movement Commands, Modelling and Simulation in Engineering, Volume 2009 (2009), Article ID 845080, 12 pages, Hindawi Publishing Corporation, ISSN: 1687-559.

Oplatkova Z., Senkerik R., Zelinka I., Holoska J., 2010a, Synthesis of Control Law for Chaotic Henon System - Preliminary study, ECMS 2010, Kuala Lumpur, Malaysia, p. 277-282, ISBN 978-0-9564944-0-5.

Oplatkova Z., Senkerik R., Belaskova S., Zelinka I., 2010b, Synthesis of Control Rule for Synthesized Chaotic System by means of Evolutionary Techniques, Mendel 2010, Brno, Czech Republic, p. 91 - 98, ISBN 978-80-214-4120-0.

Ott E., C. Greboki, J.A. Yorke, 1990. "Controlling Chaos", Phys. Rev. Lett. vol. 64, 1990, pp. 1196-1199.

Price, K. and Storn, R. (2001), *Differential evolution homepage*, [Online]: http://www.icsi.berkeley.edu/~storn/code.html, [Accessed 29/02/2012].

Price K., Storn R. M., Lampinen J. A., 2005, "Differential Evolution : A Practical Approach to Global Optimization", (Natural Computing Series), Springer; 1 edition.

Pyragas K., 1992, "Continuous control of chaos by self-controlling feedback", Physics Letters A, 170, 421-428.

Pyragas K., 1995. "Control of chaos via extended delay feedback", Physics Letters A, vol. 206, 1995, pp. 323-330.

Senkerik R., Zelinka I., Davendra D., Oplatkova Z., 2010a, "Utilization of SOMA and differential evolution for robust stabilization of chaotic Logistic equation", Computers & Mathematics with Applications, Volume 60, Issue 4, pp. 1026-1037.

Senkerik R., Oplatkova Z., Zelinka I., Davendra D.,Jasek R., 2010b, "Synthesis Of Feedback Controller For Chaotic Systems By Means Of Evolutionary Techniques,", Proceeding of Fourth Global Conference on Power Control and Optimization, Sarawak, Borneo, 2010.

Varacha P; Jasek, R., "ANN Synthesis for an Agglomeration Heating Power Consumption Approximation". In: Recent Researches in Automatic Control. Montreux : WSEAS Press, p. 239-244. ISBN 978-1-61804-004-6.

Varacha P., Zelinka I., 2008, "Distributed Self-Organizing Migrating Algorithm Application and Evolutionary Scanning". In: Proceedings of the 22nd European Conference on Modelling and Simulation ECMS 2008, p. 201-206. ISBN 0-9553018-5-8.

Zelinka I., 2004. "SOMA – Self Organizing Migrating Algorithm", In: *New Optimization Techniques in Engineering*, (B.V. Babu, G. Onwubolu (eds)), chapter 7, 33, Springer-Verlag, 2004, ISBN 3-540-20167X.

Zelinka I.,Oplatkova Z, Nolle L., 2005. *Boolean Symmetry Function Synthesis by Means of Arbitrary Evolutionary Algorithms-Comparative Study*, International Journal of Simulation Systems, Science and Technology, Volume 6, Number 9, August 2005, pages 44 - 56, ISSN: 1473-8031.

Zelinka I., Senkerik R., Navratil E., 2009, "Investigation on evolutionary optimization of chaos control", Chaos, Solitons & Fractals, Volume 40, Issue 1, pp. 111-129.

Zelinka, I., Guanrong Ch., Celikovsky S., 2008. Chaos Synthesis by Means of Evolutionary algorithms, International Journal of Bifurcation and Chaos, Vol. 18, No. 4 (2008) 911–942

TRANSFORMATION ALGORITHM FROM AN ALTERNATIVES AGGREGATION PETRI NET TO A COMPOUND PETRI NET. TWO REPRESENTATIONS OF AN UNDEFINED PETRI NET WITH A NON-EMPTY SET OF EXCLUSIVE ENTITIES

Juan Ignacio Latorre-Biel[(a)], Emilio Jiménez-Macías [(b)]

[(a)] Public University of Navarre. Deptartment of Mechanical Engineering, Energetics and Materials. Campus of Tudela, Spain
[(b)] University of La Rioja. Industrial Engineering Technical School. Department of Electrical Engineering. Logroño, Spain

[(a)]juanignacio.latorre@unavarra.es, [(b)]emilio.jimenez@unirioja.es

ABSTRACT
Simulation, analysis, decision making, and control of discrete event systems, are examples of very common applications in industrial and technological fields. All these operations require the representation of the discrete event system in an appropriate formal language, that is to say obtaining the best suited model for the current application. A very common family of formalisms is the paradigm of the Petri nets. Different Petri net-based formalisms present different modelling power and diverse features, which make them especially suited for a given operation. In the field of decision making, where there exist a number of alternative structural configurations, the alternatives aggregation Petri nets and the compound Petri nets, lead to compact models for describing a discrete event system. This paper describes a transformation algorithm between them and an example to illustrate the application of the different steps. This transformation algorithm allows a fast transformation between both formalisms for applications related to decision making, since it is not necessary to perform a previous transformation to an intermediate set of alternative Petri nets to afford the construction of a compound Petri net from an alternatives aggregation Petri net.

Keywords: Petri nets, transformation, alternatives aggregation Petri nets, compound Petri nets, decision making

1. INTRODUCTION

Petri nets constitute one of the best suited formalisms for representing discrete event systems with complex behaviour. Petri nets (PN) are in fact a family of formalisms, each one of which have been developed for being more suited for a given application. The expressiveness and modelling power of the different formalisms may be related to the constraints imposed to their definitions. Some of the formalisms introduce exogenous elements such as time, in interpreted Petri nets, or random variables, in generalized stochastic Petri nets, while others transfer information of the static structure from the elements of the incidence matrices, weight of the arcs, to features of the tokens, in coloured Petri nets (Jensen and Kristensen, 2009; David and Alla, 2005; Silva, 1993).

The applications of these particular Petri net-based formalisms range from structural analysis to performance analysis and the compact representation of large systems with shared subsystems. The transformation algorithms allow translating a model of a discrete event system form a given formalism to a different one or simplifying the representation of a given model. This translation is useful for performing certain operations in a model represented by a formalism that is not suited for the aimed application.

Alternatives aggregation Petri nets (AAPN) and compound Petri nets are two Petri net-based formalisms that have been defined for decision making. Both of them are well suited for representing in a compact way a model of a system with alternative structural configurations (Latorre *et al.* 2011b, Latorre *et al.* 2009). In real applications of decision making related to discrete event systems with alternative structural configurations it is common to represent the system by means of a set of alternative Petri nets (Tsinarakis *et al.* 2005, Zimmerman *et al.* 2001). Algorithms have been described to transform a set of alternative Petri nets into an alternatives aggregation Petri net (Latorre *et al.* 2009) and into a compound Petri net (Latorre *et al.* 2011a). In this paper, an algorithm for transforming an alternatives aggregation Petri net to a compound Petri net is described, as well as an example of application to illustrate the different steps. This algorithm is aimed to allow a direct and fast transformation of a model between these two formalisms for applications such as the comparison of the performance of a given model represented in both formalisms when integrated in a decision problem.

In section 2, the definitions that are relevant for the application of the transformation algorithm are given. The section 3 is focussed on the transformation

algorithm itself. Some considerations on a reduction rule to simplify the compound Petri net obtained from the algorithm are presented in the section 4. One important step in the transformation algorithm is explained in the section 5: the translation of the set of exclusive entities associated to the model of the discrete event system from their representation as a set of choice variables to a set of feasible combinations of values for the undefined structural parameters of the resulting compound Petri net. An example of application is detailed in the section 6, while the following section is devoted to the conclusions and future research lines. Finally, the last section lists the bibliographical references of this paper.

2. DEFINITIONS

An alternatives aggregation Petri net can be defined in the following way:

Definition 1. Alternatives aggregation Petri net system.

An alternatives aggregation Petri net system, $\boldsymbol{R^A}$, is defined as the 8-tuple:

$$R^A = \langle P, T, \text{pre}, \text{post}, \mathbf{m}_0, S_\alpha, S_{val\alpha}, S_A, \text{f}_\text{A} \rangle$$

where,

- P is the set of places.
- T is the set of transitions.
- pre is the pre-incidence matrix, also called input incidence matrix.
- post is the post-incidence matrix, also called output incidence matrix.
- $\mathbf{m}_0$ is the initial marking that represents the initial vector of state and is usually a function of the choice variables.
- S_α is a set of undefined parameters.
- $S_{val\alpha}$ is the set of feasible combination of values for the undefined parameters in S_α.
- S_A is a set of choice variables such that $S_A \neq \varnothing$ and $|S_A| = n$.
- f_A: $T \rightarrow \text{f}(a_1, \ldots, a_n)$ assigns a function of the choice variables to each transition t such that type[$\text{f}_\text{A}(t)$] = Boolean.

□

Where a set of choice variables is given by:

Let $c_{str} \in C_{str} = \{1, 2, \ldots, m_{strq}\} \subseteq \text{N}^*$.

A set of choice variables can be defined as $S_A = \{a_1, a_2, \ldots, a_{mstrq} \mid \exists!\ a_i=1, i \in C_{str} \wedge a_j=0\ \forall\ j \neq i, j \in C_{str}\}$

Furthermore, the dynamic behaviour of an alternatives aggregation Petri net is given by an enabling rule that differs slightly from most of the formalisms based on Petri nets. The firing rule is the one of a generalized Petri net.

Definition 2. Enabled transition.

Given an alternatives aggregation Petri net R^A with an associated set of choice variables $S_A = \{ a_1, a_2, \ldots, a_n \}$, let us consider the following decision:

$$a_i = 1 \Rightarrow a_i = 0\ \forall\ j \in \text{N}^* \text{ such that } 1 \leq j \leq n \wedge j \neq i$$

A transition $t_j \in T$ in an alternatives aggregation Petri net is said to be enabled if

$$m_i \geq \text{pre}(p_i, t_j)\ \forall\ p_i \in {}^\text{o}t_j \wedge \text{f}_\text{A}(t_j) = 1$$

□

On the other hand, a compound Petri net can be defined from a parametric point of view, as in (Latorre *et al*, 2011c).

Moreover, a more classic approach (Silva, 1993) for the definition of a compound Petri net can be given as stated below:

Definition 3. Compound Petri net.

A compound Petri net is a 7-tuple

$$R^c = \langle P, T, F, \text{w}, \mathbf{m_0}, S_\alpha, S_{val\alpha} \rangle, \text{ where}$$

i) S_α is the set of undefined parameters of R^c.

ii) $S_{str\alpha} \neq \varnothing$ is the set of undefined structural parameters of R^c, such that $S_{str\alpha} \subseteq S_\alpha$. Notice that S_α is the set of undefined parameters of R^c.

iii) $S_{val\alpha}$ is the feasible combination of values for the undefined parameters .

□

A compound Petri net can be considered as a parametric Petri net with undefined structural parameters.

The structural parameters refer to the elements of the incidence matrix of a Petri net. If a Petri net has undefined structural parameters it has a structure with certain freedom degrees that should be specified by a decision from the set of feasible combinations of values for them.

In summary, the undefined structural parameters are present in models that correspond with DES with undefined structure, in process of being designed, modified or controlled.

3. TRANSFORMATION FROM AN AAPN TO A COMPOUND PN

In (Latorre et al. 2011b) it was described a transformation algorithm to obtain an alternatives aggregation Petri net from a compound Petri net. The algorithm presented in this paper solves the opposite transformation and verifies that it is possible to perform a double transformation, using both algorithms sequentially, to return to the initial representation of the discrete event system. As a conclusion it is possible to state that both transformations are reversible.

The algorithm to perform a direct and fast transformation from an alternatives aggregation Petri net RA to a compound Petri net is presented in the following.

Algorithm.

Step 1.

Create a set of variables $S_{valstr\alpha}(R^c) = \{ cv_1, cv_2, \ldots, cv_{n_r} \}$ such that $|S_{valstr\alpha}(R^c)| = |S_A|$,

where $S_A = \{ a_1, a_2, \ldots, a_{n_r} \}$ is the set of choice variables of R^A.

Create a bijection between $S_{valstr\alpha}(R^c)$ and S_A.

This set $S_{valstr\alpha}(R^c)$ will contain the feasible combinations of values for the undefined structural parameters of the resulting compound Petri net.

Step 2.

Apply reduction rules to the columns of the incidence matrix of the AAPN, R^A, which have elements in common and are associated to different choice variables, aiming to obtain a more compact matrix.

Step 3.

For every transition $t_i \in T(R^A)$, with a function of choice variables associated to it, $f_A(t_i,a_j)$, transform this function into the sets $S_{str\alpha}(t_i)$ and $S_{valstr\alpha}(t_i)$.

Step 4.

Represent the resulting compound Petri net R^c.

□

4. TRANSFORMATION FROM AN AAPN TO A COMPOUND PN

Reduction rules have been developed for the simplification of Petri net models in order to perform structural analysis or performance analysis in an easier or more efficient way. See for example (Berthelot, 1987) and (Haddad and Pradat-Peyre, 2006). One of the reduction rules is based on the reduction of several identical transitions to a single one (Berthelot, 1987) and (Silva, 1993).

In order to apply this rule to an alternatives aggregation Petri net, it is necessary to consider two or more columns of the incidence matrix of the alternatives aggregation Petri net associated to functions of choice variables, which do not have any choice variable in common. It is possible to merge the mentioned columns by the creation of the appropriate undefined structural parameters if there are elements belonging to different columns but to the same row that are not equal and modifying the function of choice variables.

Furthermore, a simplication rule can also be applied, since according to the Boole algebra if the function of choice variables includes every choice variable in the form $a_1 + a_2 + \ldots + a_n$, where $|S_A| = n$, then the function can be removed since it is true after any decision that selects one of the choice variables.

5. TRANSFORMATION OF THE FUNCTIONS OF CHOICE VARIABLES INTO UNDEFINED STRUCTURAL PARAMETERS.

This step is complementary to the previous one. The difference between both operations is that the previous one merges columns of the incidence matrix of the AAPN aiming to obtain a more compact incidence matrix, while this operation manages to eliminate the functions of choice variables and to convert the AAPN into a compound alternative Petri net

In order to proceed as explained, this step develops a reverse operation to a replication of the transitions with associated function of choice variables. Taken a column with a function of choice variables that does not include a certain choice variable, a new isolated transition is added (a columns of zeros) and associated to this missing choice variable (Latorre et al. 2011b). Then, both transitions are merged by the creation of the appropriate choice variables and increasing the sets of feasible values for the undefined structural parameters if necessary.

On the other hand, the function of choice variables acquires the choice variable of the merged isolated transition.

The resulting function of choice variables might include all the choice variables. In this case, by the application of the simplification rule mentioned in the section 4, the function can be removed. Otherwise, another operation of creation of an isolated transition associated to another missing choice variable and the merge of it can be performed and so on.

As a consequence of the previous explanations it is possible to see that the operation described in this section 5 can be decomposed in the following steps:

a) Replication of the transition associated to functions of choice variables to isolate the individual choice variables. This operation is the opposite to the reduction rule of the transformation described in the previous section.

b) Addition of isolated transitions to complete the choice variables in every transition of the original net.

c) Merging of the transitions with different choice variables and with arcs to the same places and which complete the set of choice variables SA.

Notice that all the operations described in this section are the opposite operation to those applied in the reverse transformation from a compound Petri net into an alternatives aggregation Petri net (Latorre *et al*. 2011b). Due to the fact that the equivalence between the nets before and after the operations are the same and that they are reversible, their application can be performed in this algorithm.

6. EXAMPLE OF APPLICATION.

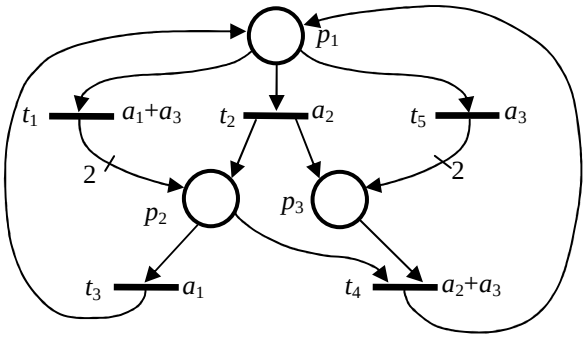

Fig. 1. Graphical representation of the AAPN to be converted into a compound PN.

This example will describe the application of the different steps of the algorithm described in the section

3 for the transformation of an alternatives aggregation Petri net into a compound Petri net. The original alternatives aggregation Petri net to be transformed is shown in the figure 1 in its graphical form and its matrix-based representation is given in the figure 2.

$$\mathbf{W}(R^A) = \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{pmatrix} -1 & -1 & 1 & 1 & -1 \\ 2 & 1 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 & 2 \end{pmatrix} \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array} \\ \begin{array}{ccccc} & a_2 & a_1 & & a_3 \end{array} \\ \begin{array}{ccc} a_1+a_3 & & a_2+a_3 \end{array} \end{array}$$

Fig. 2 Matrix-based representation of the AAPN to be converted into a compound PN.

Step 1.

The set of choice variables of the AAPN is S_A = { a_1, a_2, a_3 }.

As a consequence, the set of variables associated to the feasible combination of values for the undefined structural parameters of the resulting compound Petri net, R^c, is created:

$S_{valstr\alpha}(R^c)$ = { cv_1 , cv_2 , cv_3 } such that $|S_{valstr\alpha}(R^c)|$ = $|S_A|$ = 3.

On the other hand, a bijection between $S_{valstr\alpha}(R^c)$ and S_A is defined and their elements are made correspond, resulting in the following pairs:

(cv_1 , a_1), (cv_2 , a_2) and (cv_3 , a_3)

Step 2.

Apply reduction rules to the columns of the incidence matrix of the AAPN, R^A that have elements in common and are associated to different choice variables, aiming to obtain a more compact matrix.

The first couple of columns to be merged are the 1st and the 2nd ones. In the first case the associated function of choice variables is $f_A(t_1, a_i) = a_1 + a_3$. Moreover, the second column is associated to $f_A(t_2, a_i) = a_2$.

In order to merge both columns, the elements that are placed in the same row are compared and if they are different an undefined structural parameter is created:

Row 1: $w_{11} = w_{12} = -1 \Rightarrow w'_{11} = -1$

Row 2: $w_{21} = 2$, $w_{22} = 1 \Rightarrow w'_{21} = \alpha_4$, where $S_{val\alpha_4}$ = { 1, 2 }

Due to the fact that $w_{21} = 2$ was associated to $a_1 + a_3$ and that there is a bijection that makes the pairs (cv_1 , a_1) and (cv_3 , a_3), then $\alpha_4 = 2$ will be associated to the following combinations of choice variables: cv_1 and cv_3.

On the other hand, $w_{22} = 1$ was associated to a_2 and the bijection defines the pair (cv_2 , a_2), then $\alpha_4 = 1$ will be associated to the combination of choice variables cv_2.

Row 3: $w_{31} = 0$, $w_{32} = 1 \Rightarrow w'_{31} = \alpha_7$, where $S_{val\alpha_7}$ = { 0, 1 } and $\alpha_7 = 0$ is associated to cv_1 and cv_3, whereas $\alpha_7 = 1$ is associated to cv_2.

The result of this first merging of columns can be seen in the figure 3.

$$\mathbf{W}(R_1^A) = \begin{array}{c} \begin{array}{cccc} t_1 & t_3 & t_4 & t_5 \end{array} \\ \begin{pmatrix} -1 & 1 & 1 & -1 \\ \alpha_4 & -1 & -1 & 0 \\ \alpha_7 & 0 & -1 & 2 \end{pmatrix} \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array} \\ \begin{array}{ccc} & a_1 & a_3 \end{array} \\ \begin{array}{cc} a_1+a_3+a_3 & a_2+a_3 \end{array} \end{array}$$

Fig. 3. First reduction of transitions.

Where $S_{valstr\alpha}$ = { cv_1, cv_2, cv_3 } = { (…, $\alpha_4 = 2$, …, $\alpha_7 = 0$, …) , (…,$\alpha_4 = 1$, …, $\alpha_7 = 1$, …) , (…,$\alpha_4 = 2$, …, $\alpha_7 = 0$, …) }

Furthermore, the function of choice variables associated to the resulting transition, called t_1 in the figure 3, can be removed since it contains all the choice variables, as it is justified in the section 4.

Another couple of columns in the incidence matrix can be merged. They are the ones associated to the transitions t_3 and t_4. In the first case the associated function of choice variables is $f_A(t_3, a_i) = a_1$. Moreover, the second column is associated to $f_A(t_4, a_i) = a_2 + a_3$.

In order to merge both columns, the elements that are placed in the same row are compared and if they are different an undefined structural parameter is created:

Row 1: $w_{12} = w_{13} = 1 \Rightarrow w'_{12} = 1$

Row 2: $w_{22} = w_{23} = -1 \Rightarrow w'_{22} = -1$

Row 3: $w_{32} = 0$, $w_{33} = -1 \Rightarrow w'_{32} = -\alpha_8$, where $S_{val\alpha_8}$ = { 0, 1 } and $\alpha_8 = 0$ is associated to cv_1, whereas $\alpha_8 = 1$ is associated to cv_2 and cv_3.

The result of this second reduction of transitions can be seen in the figure 4.

$$\mathbf{W}(R_2^A) = \begin{array}{c} \begin{matrix} t_1 & t_3 & t_5 \end{matrix} \\ \begin{pmatrix} -1 & 1 & -1 \\ \alpha_4 & -1 & 0 \\ \alpha_7 & -\alpha_8 & 2 \\ & & a_3 \end{pmatrix} \\ \begin{matrix} a_1+a_3+a_3 & a_1+a_3+a_3 & \end{matrix} \end{array} \begin{matrix} \\ p_1 \\ p_2 \\ p_3 \end{matrix}$$

Fig. 4. Second reduction of transitions.

Where
$S_{valstr\alpha}$ = { cv_1, cv_2, cv_3 } = { (…, α_4 = 2 , …, α_7 = 0 , α_8 = 0, …) ,
(…,α_4 = 1 , …, α_7 = 1 , α_8 = 1 , …) , (…,α_4 = 2 , …, α_7 = 0 , α_8 = 1 , …) }

Furthermore,
the function of choice variables associated to the resulting transition can be removed since it contains all the choice variables, as it is justified in the section 4.

Step 3.

For every transition $t_i \in T(R^A)$, transform the function of choice variables associated to it, $f_A(t_i,a_j)$, into the sets $S_{str\alpha}(t_i)$ and $S_{valstr\alpha}(t_i)$.

In the case of t_1 and t_3, the associated function of choice variables has been removed and the subsequent sets $S_{str\alpha}(t_i)$ and $S_{valstr\alpha}(t_i)$ have already been obtained.

On the contrary, t_5 has an associated function of choice variables, which is $f_A(t_5,a_i) = a_3$.

In order to develop this step, two new isolated transitions will be added, associated to the choice variables a_1 and a_2 respectively (Latorre *et al.* 2011a). The result can be seen in the resulting incidence matrix written in the figure 5.

$$\mathbf{W}(R_3^A) = \begin{array}{c} \begin{matrix} t_3 & t_5 & t_6 & t_7 & \end{matrix} \\ \begin{pmatrix} -1 & 1 & -1 & 0 & 0 \\ \alpha_4 & -1 & 0 & 0 & 0 \\ \alpha_7 & -\alpha_8 & 2 & 0 & 0 \\ & & a_3 & a_1 & a_2 \end{pmatrix} \end{array} \begin{matrix} \\ p_1 \\ p_2 \\ p_3 \end{matrix}$$

Fig.5. Addition of isolated transitions to remove of the function $f_A(t_5,a_i)$.

At this stage, it is possible to reduce the transitions t_5, t_6 and t_7.

In order to merge the three columns of the incidence matrix related to these transitions, the elements that are placed in the same row are compared and if they are different an undefined structural parameter is created:

Row 1: $w_{13} = -1$ and $w_{14} = w_{15} = 0 \Rightarrow w'_{13} = -\alpha_3$, where $S_{val\alpha_3}$ = { 0, 1 } and $\alpha_3 = 0$ is associated to cv_1 and cv_2, whereas $\alpha_3 = 1$ is associated to cv_3.

Row 2: $w_{23} = w_{24} = w_{25} = 0 \Rightarrow w'_{23} = 0$.

Row 3: $w_{33} = 2$ and $w_{34} = w_{35} = 0 \Rightarrow w'_{33} = \alpha_9$, where $S_{val\alpha_9}$ = { 0, 2 } and $\alpha_9 = 0$ is associated to cv_1 and cv_2, whereas $\alpha_9 = 2$ is associated to cv_3.

The result of this step can be seen in the incidence matrix given in the figure 6.

$$\mathbf{W}(R_3^A) = \begin{array}{c} \begin{matrix} t_1 & t_3 & t_5 & t_6 & t_7 \end{matrix} \\ \begin{pmatrix} -1 & 1 & -1 & 0 & 0 \\ \alpha_4 & -1 & 0 & 0 & 0 \\ \alpha_7 & -\alpha_8 & 2 & 0 & 0 \\ & & a_3 & a_1 & a_2 \end{pmatrix} \end{array} \begin{matrix} \\ p_1 \\ p_2 \\ p_3 \end{matrix}$$

Fig.5. Addition of isolated transitions to remove of the function $f_A(t_5,a_i)$.

At this stage, it is possible to reduce the transitions t_5, t_6 and t_7.

In order to merge the three columns of the incidence matrix related to these transitions, the elements that are placed in the same row are compared and if they are different an undefined structural parameter is created:

Row 1: $w_{13} = -1$ and $w_{14} = w_{15} = 0 \Rightarrow w'_{13} = -\alpha_3$, where $S_{val\alpha_3}$ = { 0, 1 } and $\alpha_3 = 0$ is associated to cv_1 and cv_2, whereas $\alpha_3 = 1$ is associated to cv_3.

Row 2: $w_{23} = w_{24} = w_{25} = 0 \Rightarrow w'_{23} = 0$.

Row 3: $w_{33} = 2$ and $w_{34} = w_{35} = 0 \Rightarrow w'_{33} = \alpha_9$, where $S_{val\alpha_9}$ = { 0, 2 } and $\alpha_9 = 0$ is associated to cv_1 and cv_2, whereas $\alpha_9 = 2$ is associated to cv_3.

The result of this step can be seen in the incidence matrix given in the figure 6.

$$\mathbf{W}(R_4^A) = \begin{array}{c} \begin{matrix} t_1 & t_3 & t_5 \end{matrix} \\ \begin{pmatrix} -1 & 1 & -\alpha_3 \\ \alpha_4 & -1 & 0 \\ \alpha_7 & \alpha_8 & \alpha_8 \\ & a_3 & a_1 \quad a_2 t_1 \end{pmatrix} \end{array} \begin{matrix} \\ p_1 \\ p_2 \\ p_3 \end{matrix}$$

Fig.6. Result of the reduction of the transitions t_5, t_6 and t_7 into a single transition named t_5.

Where $S_{str\alpha}$ = { α_3 , α_4 , α_7 , α_8 , α_9 } and $S_{valstr\alpha}$ = { cv_1, cv_2, cv_3 } = { (0,2,0,0,0), (0,1,1,1,0), (1,2,0,1,2) }

Furthermore, the function of choice variables associated to the resulting transition can be removed since it contains all the choice variables, as it is justified in the section 4.

Step 4.

Represent the resulting compound Petri net R^c.

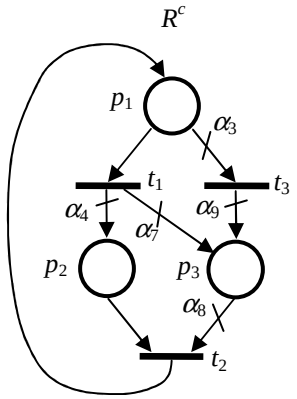

Fig. 7.Graphical representation of the resulting compound Petri net.

$$\mathbf{W}(R^c) = \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{pmatrix} -1 & 1 & -\alpha_3 \\ \alpha_4 & -1 & 0 \\ \alpha_7 & -\alpha_8 & \alpha_9 \end{pmatrix} \end{array} \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array}$$

Fig. 8. Matrix-based representation of the resulting compound Petri net.

The result of this transformation algorithm can be seen in the graphical and matrix-based representations of the resulting compound Petri net, which is equivalent to the original alternatives aggregation Petri net. These representations are shown in the figures 7 and 8 respectively.

$|S_{valstr\alpha}(R^c)| = |\{ (0,2,0,0,0) , (0,1,1,1,0) , (1,2,0,1,2) \}|=3$

7. CONCLUSIONS

As a conclusion of this paper it can be stated that with this algorithm it has been completed the set of transformations between three common Petri net-based formalisms to represent Petri nets with alternative structural configurations for the main purpose of developing automatic decision support systems: the set of alternative Petri nets, the compound Petri net and the alternatives aggregation Petri net. It is now possible to perform any direct transformation between any pair of formalisms belonging to the mentioned group.

As open research lines it can be considered the analysis of the freedom degrees in these algorithms to adjust them in order to obtain the most compact models for the development of the most efficient optimization problems to solve the original decision problems.

ACKNOWLEDGMENTS

This paper has been partially supported by the project of the University of La Rioja and Banco Santander (grant number API12-11) 'Sustainable production and productivity in industrial processes: integration of energy efficiency and environmental impact in the production model for integrated simulation and optimization'.

REFERENCES

R., Alla. H., 2005. *Discrete, Continuous and Hybrid Petri nets*, Springer.

Jensen, K., Kristensen, L.M., 2009. *Coloured Petri nets. Modelling and Validation of Concurrent Systems*. Springer.

Berthelot, G., 1987. Transformations and decompositions of nets. In: Brauer, W., Reisig, W., and Rozenberg, G., eds. *PetriNets: Central Models and Their Properties, Advances in Petri Nets. Lecture Notes in Computer Science*, vol. 254-I, pp. 359–376. Springer, 1987.

Haddad, S. and Pradat-Peyre, J.F., 2006. New Efficient Petri Nets Reductions for Parallel Programs Verification. *Parallel Processing Letters*, pages 101-116, World Scientific Publishing Company.

Silva, M., 1993. Introducing Petri nets. In: Di Cesare, F., ed. *Practice of Petri Nets in Manufacturing*, pp. 1-62. Ed. Chapman&Hall.

Latorre, J.I., Jiménez, E., Pérez, M., 2009. Decision taking on the production strategy of a manufacturing facility. An integrated methodology. *Proceedings of the 21st European Modelling and Simulation Symposium* (EMSS 09). Puerto de la Cruz, Spain, vol. 2, pp. 1-7.

Latorre, J.I., Jiménez, E., Pérez, M., 2011. Matrix-based operations and equivalence classes in alternative Petri nets. *Proceedings of the 23rd European Modelling and Simulation Symposium* (EMSS 11). Rome, Italy, pp. 587-592.

Latorre, J.I., Jiménez, E., Pérez, M., 2011. Petri net transformation for decision making: compound Petri nets to alternatives aggregation Petri nets. *Proceedings of the 23rd European Modelling and Simulation Symposium* (EMSS 11). Rome, Italy, pp. 613-618.

Latorre, J.I., Jiménez, E., Pérez, M., 2011. Petri nets with exclusive entities for decision making. *International Journal of Simulation and Process Modeling, Special Issue on the I3M 2011 Multiconference*. Inderscience Publishers.

Tsinarakis, G. J., Tsourveloudis, N. C., and Valavanis, K. P., 2005. Petri Net Modeling of Routing and Operation Flexibility in Production Systems. *Proceedings of the 13th Mediterranean Conference on Control and Automation*, pages 352-357.

Zimmermann, A.; Freiheit, J.; Huck, A. 2001. A Petri net based design engine for manufacturing systems. *International Journal of Production Research*, Vol. 39, No. 2, pages 225-253.