RESEARCH PAPERS
FACULTY OF MATERIALS SCIENCE AND TECHNOLOGY IN TRNAVA
SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

2011 Number 30

EVENT PROCESSING AND VARIABLE PART OF SAMPLE PERIOD DETERMINING IN COMBINED SYSTEMS USING GA

Maximilián STRÉMY[1], Pavol ZÁVACKÝ[2], Martin JEDLIČKA[3]

Abstract

This article deals with combined dynamic systems and usage of modern techniques in dealing with these systems, focusing particularly on sampling period design, cyclic processing tasks and related processing algorithms in the combined event management systems using genetic algorithms.

Key words

events, sample period, genetic algorithms, combined dynamic systems

Purpose of the article

This article deals with combined dynamic systems and usage of modern techniques in dealing with these systems, focusing mainly on sampling period design, cyclic processing tasks and related processing of the control algorithms in the combined control dynamic systems using genetic algorithms.

Combined dynamic systems

The fusion result of time-driven and event activated systems are combined dynamic systems, called also hybrid systems. Since the notion "hybrid system" is used also in connection e.g. between distributed control system and programmable logical controllers, neural networks, genetic algorithms and fuzzy logics, or in combination of electric and

[1] Maximilián Strémy, Ph.D. – Institute of Applied Informatics, Automation and Mathematic, Faculty of Materials Science and Technology, Slovak University of Technology Bratislava, Hajdóczyho 1, 917 24 Trnava, Slovak Republic, e-mail: maximilian.stremy@stuba.sk
[2] Pavol Závacký, Ph.D. – Division of Communication and Information Systems, Faculty of Materials Science and Technology, Slovak University of Technology Bratislava, Paulínska 16, 917 24 Trnava, Slovak Republic, e-mail: pavol.zavacky@stuba.sk
[3] Martin Jedlička, Ph.D. – University of SS, Cyril and Methodius in Trnava, nám.Jozefa Herdu 2, 917 01 Trnava

mechanical power units, the concept of "combined dynamic system" was introduced in order to better predict and better identify the time-driven and event activated systems.

If the change is dependent on events occurring at discrete time moments, we talk about discrete event dynamic systems. In case the conditions change is dependent from periodic processing, monitoring and system evaluation in discrete time moments, we talk about discrete time-driven systems. Since it is management of real time systems, it is very important to set up right sampling time. By time-driven systems, events are processed through cyclic monitoring, processing and evaluating inputs and system states. In discrete event dynamic systems, this part is implemented by calling internal interrupts by occurrence of any event. Given the sampling period, we classify combined dynamic systems into two groups:

a) Constant sampling time – by setting up, it is necessary to determine the ratio of event component T_P and time-driven system component T_R.
b) With varying sampling time – it is necessary to determine the impact, respectively relationship between T_R a T_P.

Dynamic control system works with sampling time T_0 (of cycle processing), corresponding to Shannon-Kotelnik theorem of time, needed to process all processes in each cycle of execution.

Generally in dynamic systems, sampling period T_0 is considered constant and includes only time T_R processing of all necessary control and cyclically recurring processes in the system. On the other hand, combined dynamic systems means a sampling period, no matter whether constant or varying, enlarged of event time constant T_P, needed to carry out random events in the system.

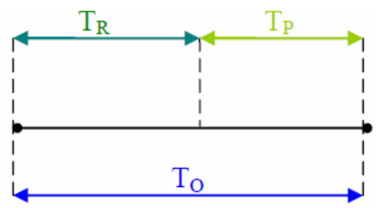

Fig. 1 *Sampling time T_0*

To determine the time T_P, and in order to carry out servicing routines of random events in the cycle, I used statistical evaluation of emerging events and their probability estimate, and determined the most probable time constant T_P, using one of the probability distribution.

Generally, we may suppose that time T_A needed to process all events in a combined dynamic system is [1]:

1. $T_A<T_0$ – less then total sampling period T_0, while
   a) $T_R>>T_A$ – constant of time-driven system T_R is much bigger than time needed to process all events T_A, then this time T_A is negligible,
   b) $T_R \approx T_A$ – constant of time-driven system T_R is approximately equal to time needed for processing all events T_A,
   c) $T_R << T_A$ – time T_A needed to process all events is much bigger then constant T_R of time-driven system.

2. $T_A > T_0$ – is much bigger then total sampling period, than minimal number of steps K_{min} needed to carry out services; it is determined by the equation $K_{min} \geq T_A/T_P$, or $T_P=T_A/K_{min}$, while $T_A = T_{A1} + T_{A2} + ... + T_{AK}$.

According to the abovementioned approach, the total time needed to carry out all servicing routines lies in the number of cycles according to the value of step K_{min}.

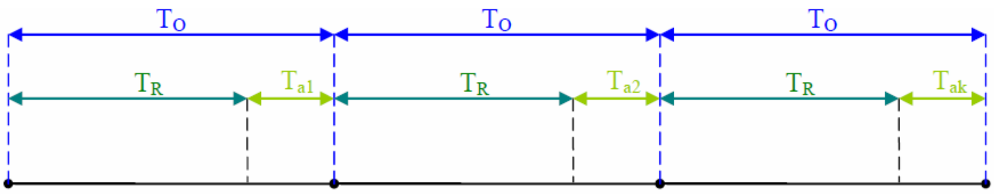

Fig. 2 *Decomposition of T_A into several steps with respect to T_P*

This article describes an appropriate part of control algorithm that can process generated events and also determine which of the events will be processed in a given cycle and which ones in the next cycle according to the rules based on GA. This is important when sum of times needed to process all events in the system is bigger than value T_0 and we need minimum steps K_{min}. In this case, it is necessary to select those events which should be processed in the current cycle time, and to process as many events as possible in the specified time.

Genetic algorithms

Algorithm begins with solving group (represented by chromosomes) also called population. Solutions from one population are selected and used to create the new population. This is motivated by the fact that new population will be better that the old one. Solutions, which are later selected to form a new population, are selected on the basis of their suitability. The more convenient they are, the greater the likelihood of reproduction. This is repeated until the fulfilment of any conditions. Flowchart of GA is illustrated in the following Figure:

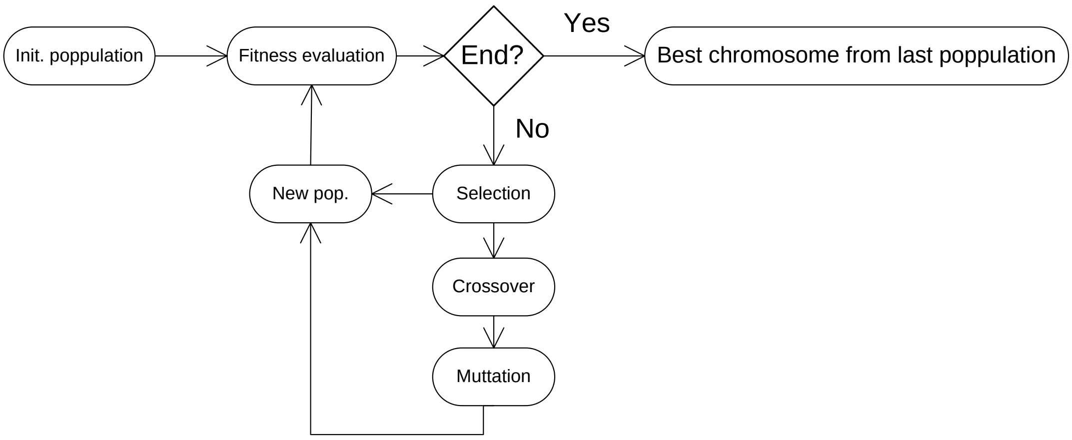

Fig. 3 *Flowchart of GA*

The idea is to select better parents in the hope that better parents will create better offspring. The process of natural evolution is simple, but very powerful and robust, universally valid for simple and complex. There is no feedback to it. The organisms are able to get out of the local extremes and move to global optima. Searching is carried out simultaneously in several directions. They do not need information on the development of solutions such as gradient of objective function [2]. They want only to assess the objective function at each point area [3]. They are able to solve optimization problems with hundreds of variables within the most challenging time solutions.

Genetic algorithm in control algorithm

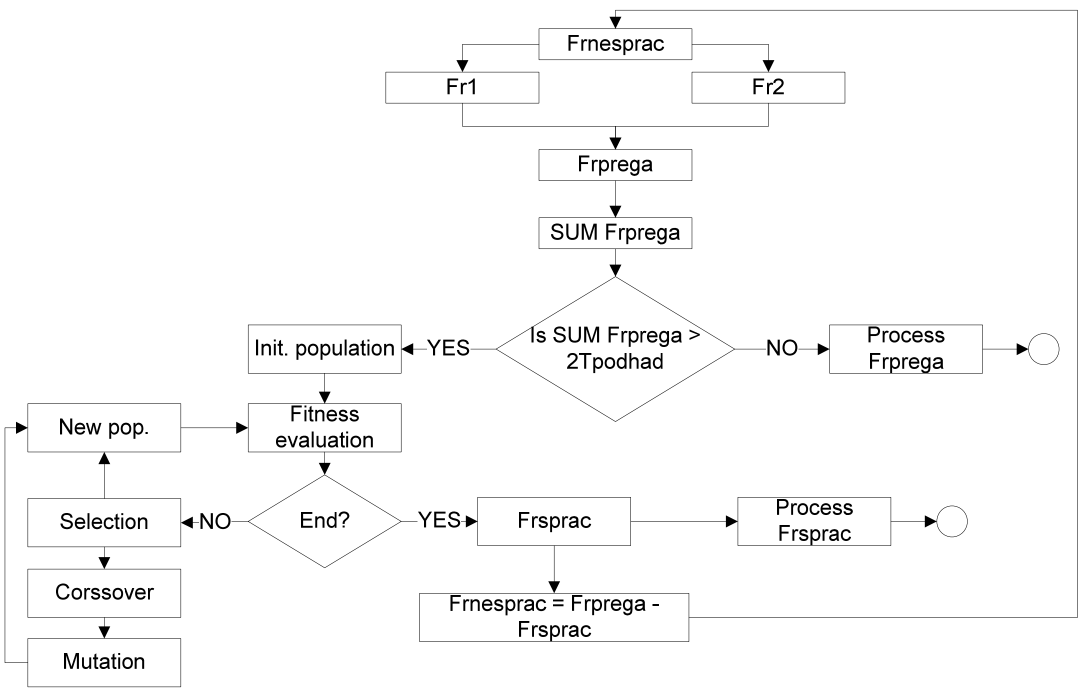

Fig. 4 *Flowchart of GA, part of draft algorithm*

The proposed algorithm uses two main queues (F_{r1} and F_{r2}) to which emerging events are collected [1]. In actually inactive queue, there are events which occur in the system and should be processed in the next cycle.

In actually active queue, accumulated events are processed. The choice of the event to be processed is based on the genetic algorithm. It is necessary for the 1st population to create a queue with the events from actually an active queue, and events which were not processed before by the GA. Given that the algorithm is a variant designed to perform all statistical forecast of the events in the system, the maximum sampling period is determined by the equation:

$$T_{0max} = T_R + 2T_{Podhad} . \qquad (1)$$

Queue F_{rprega} contains all the raw events from the previous cycle, and all events of the currently active event queue, regardless of whether the time required to perform all the abovementioned events is bigger or smaller then $2T_{Podhad}$. When the switch between active and inactive queues comes on, this queue is read-out and processed by the genetic algorithm.

Before the GA processing occurs, sum of all individual times required for processing the events is created. If the sum is less than $2T_{Podhad}$, queue F_{rsprac} is automatically filled and the events in the queue are processed. If the sum is more than $2T_{Podhad}$, it is necessary to select the events with GA, which should be processed. The main goal is to process as many events from the queue F_{rprega} as possible in maximum time $2T_{Podhad}$.

This means that each chromosome is a solution, and each gene in chromosome is one event, respectively the time required for its processing. Lengths of chromosomes are generated randomly, and also events occurring in different genes are generated randomly. The fitness function will be applied to evaluate the new chromosomes:

$$fitness = \frac{number\ of\ genes\ in\ chromosome}{\sum time\ in\ each\ gene} \cdot 100\% \qquad (2)$$

The fitness function was chosen on the basis that our aim is to handle the largest number of events in the shortest time, but not longer than $2T_{Podhad}$. The function comprises also an integrated classification to order the events from the view of the time requirements - from the largest to the smallest – while using genetic operations like selection, crossover and mutation. The selection will be based on elitism, the set % of the best individuals will be automatically transferred to the next generation. Applying crossover operations, two situations can occur:

- one event in chromosome can occur multiple times,
- one event in chromosome can occur only once.

By genetic operation mutation, we will mutate individual chromosomes in a way that we add or remove genes in each chromosome. If total length of chromosome is not bigger then $2T_{Podhad}$, we will add random event; on the other hand, if it is bigger than $2T_{Podhad}$, we will remove unnecessary genes.

Thus after a set number of generations, the chromosome which best suits our needs and individual events contained in this chromosome will be placed in queue F_{rsprac} to be processed. Raw events are transferred to the queue $F_{rnesprac}$ which will be included once again in the queue of events F_{rprega} to start. If the queue $F_{rnesprac}$ is empty, the whole cycle of the algorithm is processed, but if this front is not empty, the value Count is set-up. If this value Count is bigger than the declared value N, redetermination of time event component T_{Podhad} occurs and sets a new sampling period of the controller. The actual value of the sampling period in this control process is fully dependent on the number of generated events in the system and related service routines.

Conclusions

The proposed control algorithm part of combined dynamic systems should be tested. Testing and simulation of the proposed algorithm will be made in Matlab / Simulink and the toolboxes SimEvents for generating stochastic events in random times and GAToolBox for operations with genetic algorithm. It is necessary to determine how many populations are needed to find a suitable solution, and it is also necessary to test the overall functionality and accuracy of the proposed solutions. The solution presents a new approach to organizing the combined event dynamic systems, since the algorithm handles all the events generated in the system, when time needed to their processing is not bigger than the allowed time to process all stochastic generated events in the system and to avoid undesirable conditions such as rejection or failure events, which could lead to emergency situations.

References:

[1] STRÉMY, M. *Combined discrete dynamic systems*. Trnava: AlumniPress, 2010. ISBN 978-80-8096-113-8

[2] HURAJ L., SILÁDI, V. Authorization through Trust Chains in Ad hoc Grids. In *Proceedings of the 4th ACM EATIS annual international conference on Telematics and Informatics: New Opportunities to increase Digital Citizenship (EATIS '09)*. Prague: 2009, pp. 68-71. ISBN 978-1-60558-398-3

[3] TANUŠKA, P. et al. The Proposal of Step Stress Testing Automation. In *International Journal of Computer Theory and Engineering*, 2010, Vol. 2, No 6, p. 860. ISSN 1793-8201

Reviewers:

Pavol Tanuška, Assoc. Professor, Ph.D. - Institute of Applied Informatics, Automation and Mathematic, Faculty of Materials Science and Technology, Slovak University of Technology Bratislava

Anton Kachaňák, Assoc. Professor, Ph.D. – Faculty of Machining, Slovak University of Technology Bratislava