# Genetic Algorithm Implemented as an Upgraded Petri Net for Searching Maximum of a Given Function

Perica S. Štrbac
Faculty of Computer Science
Belgrade, Serbia
strbac68@gmail.com

Nikola Davidović
Faculty of electrical engineering
Istočno Sarajevo, RS, BiH
nikola.davidovic@etf.unssa.rs.ba

*Abstract* — **The objective of this paper is modeling, simulation and analysis of Upgraded Petri Net (UPN) model which implements a genetic algorithm for searching maximum of a given function. Original software for modeling and simulations of UPN, PeM (Petri Net Manager), is used for all models described in this paper. The paper includes UPN theory and the UPN model of genetic algorithm for searching a maximum of a given function inside given two dimensional area which refers to: chromosome encoding, the first population generating, fitness evaluation, *k*-tournament selection, crossing, mutation and the next population generating. The paper also represents simulation and analysis of the UPN model. The executions of UPN are based on parallel firing of a group of transitions. The suitability of UPN for modeling the genetic algorithm for searching maximum of a given function is examined and established.**

***Key words — Upgraded Petri Net; Genetic Algorithm; Modeling; Simulation; Analysis;***

## I. INTRODUCTION

This paper presents the usage of Upgraded Petri nets in the example of modeling, simulation and analysis of a genetic algorithm which searching for a maximum of a given function in a given area.

Upgraded Petri nets are a formal mathematical apparatus which enables modeling, simulation and process analysis [1] through interactive monitoring of Petri net execution from the initial phase, all the way to the final version. The UPN was developed for simulation and analysis of processes, particularly at the register transfer level. The hierarchical structure of an UPN enables that a model consists at the same time of elaborate pieces essential for the analysis at a certain level, and also of some general pieces whose details are irrelevant for the analysis at the given level of abstraction [2]. The UPN is an extension of an ordinary Petri Net and a formal modeling tool appropriate for simulation and analysis of processes, particularly at the register transfer level (RTL) [3].

In this paper chromosomes are coded as triplet of: $x$ coordinate, $y$ coordinate and calculated value of the given function over these two coordinates. These $x$ and $y$ coordinates belong to the given area. As the first, we make initial population of chromosomes. Main goal is to find maximum of the given function over the set of chromosomes through $c$ iterative cycles. The cycle includes: $k$-tournament selection, crossing, mutation and creating the next population of chromosomes.

The chromosomes in the population will be evaluated by fitness function, recombined and mutated until the $c$ iterative cycles is finished. The fitness function is determined by the greatest value of the third part of the triplet of the chromosome. The phases of a life cycle are: the population is sorted by fitness, $n$ of the best evaluated chromosomes go to the next generation as elitism and the rest population will be populated by recombination and mutation of the selected rest chromosomes. Selection is based on $k$-tournament where we choose two best parents of the $k$ randomly selected chromosomes. The crossing is done by choosing two random position between positions of the parents to generate two different offspring chromosomes. The mutation is done by moving position determined by $x$ and $y$ value of the chromosome by random two dimensional vector. As an example, the UPN model implements chromosome encoding, the first population generating, fitness evaluation, $k$-tournament selection, crossing, mutation and the next population generating.

For given input parameters the UPN model through its execution simulates described genetic algorithm [3].

## II. AN UPN FORMAL THEORY

Upgraded Petri nets formal theory is based on functions [3]. Upgraded Petri net is a 9-tuple:

$C = (P, T, F, B, \mu, \theta, TF, TFL, PAF)$

where:

$P=\{p_1, p_2, p_3, ..., p_n\}, n>0$
- a finite nonempty set of places $p_i$

$T=\{t_1, t_2, t_3, ..., t_m\}, m>0$
- a finite nonempty set of transitions $t_j$

$F: T\times P\rightarrow N_0$ - Input Function;
$B: T\times P\rightarrow N_0$ - Output Function;
$\mu: P\rightarrow N_0$ - Marking Function:
$\theta: T\times\Delta\rightarrow\lambda$ - Timing Function;
$TF: T\rightarrow A$ - Transition Function;
$TFL: T\rightarrow N_0$ - Transition Firing Level;
$PAF: P\rightarrow(x, y)$ - Place Attributes Function;

The input function assigns a non-negative number to an ordered pair $(t_i,p_j) \in T \times P$. The assigned non-negative number defines how many times the place pi is input as compared to the transition $t_i$. $N_0$ represents the set of non-negative integers. The set of places which are input as compared to the transition $t_j$ is presented as follows $*t_j = \{ p_i \in P, F(t_j, p_i) > 0 \}$. For the presentation of the place $p_i \in *t_j$ which have the standard input compared to the $t_j$, the sign $*t_j^S$ will be used, and sign $F^S(t_j, p_i)$ will be used for such input function. For the places $p_i \in *t_j$ with inhibitor input in relation to the $t_j$ transition, the sign $*t_j^I$ will be used and sign $F^I(t_j, p_i)$ for the input function.

The output function gives a non-negative integer to the ordered pair $(t_i,p_j)$. The assigned non-negative integer shows how many times the place $p_i$ is input in relation to the $t_i$ transition. The set of places which are input in relation to the $t_j$ transition is presented as follows $t_j* = \{ p_i \in P, B(t_j, p_i) > 0 \}$.

The marking function assigns a non-negative integer to the pi place. The marking function can be defined as n-dimension vector (marking): $\mu=(\mu_1,\mu_2,...,\mu_n)$, where $n=|P|$. Instead of sign $\mu_i$ it can be used the sign $\mu(p_i)$.

The timing function $\theta$ assigns the probability $\lambda_{ij} \in [0,1]$ to an ordered pair $(t_i, j) \in T \times N_0$, i.e., $\lambda_{ij} = \theta(t_i, j)$.

The transition function gives an operation $\alpha_j \in A$ to the $t_j$ transition. Sign $A$ is the set of operations which can be assigned to the transition.

The firing level function of the transition gives a non-negative integer to the transition $t_j$. If this number not equals zero, it shows the number of $p_i \in *t_j$ places takes part in the transition firing, and if this number equals zero, then all the places $p_i \in *t_j$ affect the $t_j$ transition firing.

Place attributes function assigns an ordered pair $(x,y)$ to the place $p_i$. The $x$ component is a real number called $x$ attribute, and $y$ is a non-negative integer called $y$ attribute (i.e. $x \in R$, $y \in N_0$). Over the $x$ attribute belonging to the $p_i \in *t_j$ places, the $\alpha_j$ operation assigned to the $t_j$ transition executes where the order of operands in the operation $\alpha_j$ is defined by the $y$ attributes which belong to the $p_i$ place in accordance to $TFL$ function take part in the transition firing.

An Operation Assigned to a Transition: Function $TF$ assigns to a transition $t_j$ one operation. This operation can be: arithmetical operation, logical operation or file operation [1]. Inside the suite *PeM* a file which is a target of file operation function has an **.mem* extension. This **.mem* file is a text file and it is used for simulation of computer system memory. One line inside the **.mem* file refers to context of one memory location of computer system that we are modeling.

An arithmetical operation $\alpha_j \in A$, which is assigned to the transition $t_j \in T$, uses attributes $x$ which belong to the places $p_i \in *t_j$ as operands of that operation. A result of an arithmetical operation $\alpha_j \in A$ will be placed into the attributes $x$ which belong to the places $p_i \in t_j*$. The order of an operand (i.e. order of attributes $x$ which belong to the places $p_i \in *t_j$) in an arithmetical operation $\alpha_j \in A$ is defined by attributes $y$ which belong to the places $p_i \in *t_j$.

A logical operation $\alpha_j \in A$ which is assigned to the transition $t_j \in T$, uses attributes x which belong to the places $p_i \in *t_j$ as operands of that operation. If a result of the logical operation $\alpha_j \in A$ is logical false the transition $t_j \in T$ is disabled and will stay in that state until the result of this logical operation $\alpha_j \in A$ becomes logical true. The order of an operand (i.e. order of attributes $x$ which belong to the places $p_i \in *t_j$) in a logical operation $\alpha_j \in A$ is defined by attributes $y$ which belong to the places $p_i \in *t_j$.

A file operation $\alpha_j \in A$ which is assigned to the transition $t_j \in T$ performs over the context of a file which extension is equal to **.mem*. A File Operation $\alpha_j \in A$ addresses context of a **.mem* by using attribute $x$ which belongs to the places $p_i \in *t_j$. A result of this operation $\alpha_j \in A$ changes the value of attributes $x$ which belong to the places $p_i \in t_j*$. The result also can change attributes $y$ which belong to the places $p_i \in t_j*$, or can change context of addressed line into the **.mem* file.

An UPN Graph: Upgraded Petri Net is represented via formal mathematical apparatus or graphically. An UPN is represented by bipartite multigraph as is in Petri-net.

An Upgraded Petri Net executing represents change of system state from the current state to the next state. This migration from one state to the other one is triggered by firing of the transitions. By UPN executing: marking vector can be changed, contents of **.mem* file can be changed, and attributes which belong to the places $p_i \in t_j*$ of enabled transition $t_j$ can be changed.

A transition $t_j \in T$ can be enabled in Upgraded Petri Net: $C = (P, T, F, B, \mu, \theta, TF, TFL, PAF)$ if the next 3 conditions are satisfied:

1° If the timing function $\lambda_{jk} = \theta(t_j, k) > 0$; (1)

2° If $TFL(t_j) > 0$ then $(\#p_i(S)) + (\#p_i(I)) = TFL(t_j)$, (2)
and if $TFL(t_j) = 0$ then $(\#p_i(S)) + (\#p_i(I)) = |*t_j|$,
where
$\#p_i(S)$ is a number of places $p_i \in *t^S_j$ such that
$\mu(p_i) \geq F^S(t_j, p_i)$, and
$\#p_i(I)$ represents a number of places $p_i \in *t^I_j$ for which
$\mu(p_i) = 0$;

3° If a logical operation $\alpha_j \in A$ assigned to the transition $t_j$, then the result of the operation $\alpha_j$ must be equal to true. (3)

A marking vector $\mu$ will be changed to new marking vector $\mu'$ by firing of transitions $t_j$, where:

$\mu'(p_i) = \mu(p_i) - F(t_j, p_i) + B(t_j,p_i)$, for places $p_i \in *t_j^S$
$\mu'(p_k) = \mu(p_k) + B(t_j,p_i)$, for places $p_k \in *t_j^I$

By firing of the transition $t_j$ an arithmetic operation is executed or a file operation is executed with respect to the operation that is assigned to $t_j$ by function $TF(t_j)$.

A logical operation which assigned to the transition tj by function $TF(t_j)$ will be executed if conditions (1) and (2) related to $t_j$ are equal to true.

A conflict in Upgraded Petri Net influences UPN executing. A conflict in UPN is the same as the conflict in Petri-net.

An UPN reachability tree graphically represents all possible marking vectors which can occurs during an UPN execution for given initial marking. Reachability tree shows all states which model can reach from the initial state. The UPN reachability tree is the same as the Petri Net reachability tree.

An UPN executing refers to a concurrent firing of the enabled. An UPN execution generates an UPN flammability tree. This tree is such tree where a node of the tree is a set of the transitions which are enabled at the same time. If there is the same node as the current node in the flammability tree then generating of the flammability tree will be stopped. There are four types of nodes in a flammability tree: root node, double node, dead node, and inner node.

## III. UPN MODELS AND THEIR EXECUTION FOR FINDING MAXIMUM OF A FUNCTION

In this section an UPN model is described. The model refers to genetic algorithm which searching for maximum of a given function inside the given area. [4], [5]. The UPN model which refers to initiation of the population is shown in *Figure 1*. Initial marking is $\mu_{(p-6)} = 12$, $\mu_{(p-7)} = 1$, $\mu_{(p-26)} = 4$. Places p-6, p-7 and p-26 refer to number of chromosome which will be generated, synchronizer and number of life cycles of genetic algorithm, respectively. By firing of enabled *t-3* transition a new marking becomes $\mu_{(p-6)} = 11$, $\mu_{(p-26)} = 4$, $\mu_{(p-5)} = 1$. This state of the model refers to ready state for generating the *x* and *y* coordinates of a new chromosome and the transitions *{t-1, t-2}* become enabled because of the marking and inhibitors arcs $F^I(t\text{-}1, p\text{-}1)=1$, $F^I(t\text{-}1, p\text{-}2)=1$, and $F^I(t\text{-}2, p\text{-}3)=1$, $F^I(t\text{-}2, p\text{-}4)=1$. By firing of this transitions a new marking becomes $\mu_{(p-6)} = 11$, $\mu_{(p-26)} = 4$, $\mu_{(p-8)} = 1$ and $\mu_{(p-9)} = 1$. After this, *x* and *y* coordinates of the new chromosome are generated. Now, transition t-4 is enabled and after firing of this transitions given function for the new chromosomes will calculated by using the generated coordinates *x* and *y*. At this moment we have the marking $\mu_{(p-6)}=11$, $\mu_{(p-26)}=4$ and $\mu_{(p-10)} =3$ and transitions *{t-5, t-6, t-7}* are enabled because of inhibitor

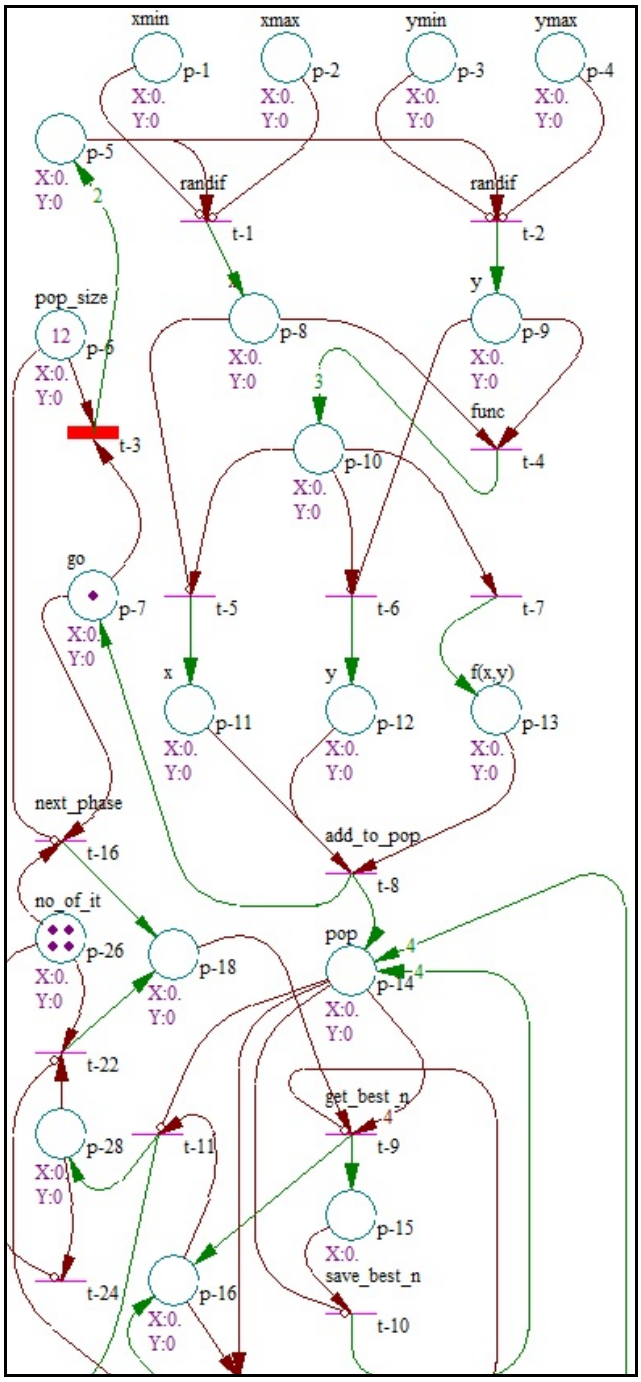


Figure 1. UPN model of genetic algorithm which searches the maximum of a given function, initiation of the population

arcs $F^I(t\text{-}5, p\text{-}8)=1$ and $F^I(t\text{-}6, p\text{-}9)=1$ and $\mu_{(p-10)} = 3$. By firing of these transitions a new marking is $\mu_{(p-6)} = 11$, $\mu_{(p-26)} = 4$, $\mu_{(p-11)} = 1$, $\mu_{(p-12)} = 1$ and $\mu_{(p-13)} = 1$ and transition *t-8* is enabled. By firing of this transition the new chromosome will be added to the population (place *p-14*) and a new marking is $\mu_{(p-6)} = 11$, $\mu_{(p-26)} = 4$, $\mu_{(p-7)} = 1$ and $\mu_{(p-14)} = 1$ so transition *t-3* is enabled again and the next cycle of generating a new chromosome is ready to start. The marking of the place *p-14* refers to number of generated chromosomes. After firing of

the transition *t-3* a new marking is $\mu_{(p-6)} = 10$, $\mu_{(p-7)} = 5$, $\mu_{(p-26)} = 4$ so model is going to generate the next chromosome and number of remaining chromosomes to generates *(p-6)* decreases by one.

When the initial population is generated (through previous described cycles) a *k*-tournament selection can starts which is shown in Figure 2. [6]. Now the marking is $\mu_{(p-26)} = 4$, $\mu_{(p14)} = 12$ and transition *t-16* is enabled. This means that there are 4 evolution cycles left (*p-26*), there are 12 chromosomes generated (*p-14*) and the first cycle of the selection starts. By firing of the transition *t-16* a new marking becomes $\mu_{(p-26)} = 3$, $\mu_{(p14)} = 12$, $\mu_{(p18)} = 1$, and the transition *t-4* is enabled. This transition refers to choosing *n* the best chromosomes as elite. [7], [8], [9]. By firing of this transition the marking becomes $\mu_{(p-26)} = 3$, $\mu_{(p14)} = 8$, $\mu_{(p15)} = 1$, $\mu_{(p16)} = 1$, because of multiplicity arcs $F^S(t\text{-}9, p\text{-}14)$=4. This multiplicity is equivalent to number chromosomes of elitism (in our model *n=4)*. At this moment the transition *t-12* is enabled. This transition will burn until all the remaining chromosomes from the initial population are transferred to our selection set (*p-17*). This is possible because of $F^S(t\text{-}12, p\text{-}6) = B^S(t\text{-}12, p\text{-}6)$ and $F^S(t\text{-}12, p\text{-}14) = 1$. After 4th firing of this transition a new marking is $\mu_{(p-26)} = 3$, $\mu_{(p14)} = 4$, $\mu_{(p15)} = 1$, $\mu_{(p16)} = 1$, $\mu_{(p17)} = 4$, $\mu_{(p20)} = 4$, so the transitions *{t-12, t-17}* are enabled. By firing of this set of transitions a new marking is $\mu_{(p-26)} = 3$, $\mu_{(p14)} = 3$, $\mu_{(p15)} = 1$, $\mu_{(p16)} = 1$, $\mu_{(p17)} = 5$, $\mu_{(p20)} = 1$, $\mu_{(p21)} = 1$ and the transitions *{t-12, t-18}* are enabled. By firing of these transitions a new marking is $\mu_{(p-26)} = 3$, $\mu_{(p14)} = 2$, $\mu_{(p15)} = 1$, $\mu_{(p16)} = 1$, $\mu_{(p17)} = 6$, $\mu_{(p20)} = 2$, $\mu_{(p22)} = 3$ and the transition *t-12* is enabled. After the next 2 successive firing of the transition t-12 a new marking is $\mu_{(p-26)} = 3$, $\mu_{(p15)} = 1$, $\mu_{(p16)} = 1$, $\mu_{(p17)} = 8$, $\mu_{(p20)} = 4$, $\mu_{(p22)} = 3$ and transitions *{t-10, t-11, t-17}* are enabled. A firing of the transition *t-10* our elitism chromosomes will be put in the next population ($\mu_{(p14)} = 4$). By firing of the transition *t-11* a marking $\mu_{(p-28)} = 1$ and it will be used for synchronization of the start for the next generation. After firing of the transition *t-17* a marking $\mu_{(p-21)} = 1$ which means there is one more cycle for *k*-tournament. At this moment a new marking is $\mu_{(p-26)} = 3$, $\mu_{(p14)} = 4$, $\mu_{(p28)} = 1$, $\mu_{(p19)} = 3$, $\mu_{(p17)} = 8$, $\mu_{(p21)} = 1$, $\mu_{(p22)} = 3$ and the transitions *{t-13, t-14, t-15}* are enabled. By firing of these transitions a new marking is $\mu_{(p-26)} = 3$, $\mu_{(p14)} = 4$, $\mu_{(p28)} = 1$, $\mu_{(p19)} = 3$, $\mu_{(p17)} = 8$, $\mu_{(p21)} = 1$, $\mu_{(p23)} = 1$, $\mu_{(p24)} = 1$, $\mu_{(p25)} = 1$, $\mu_{(p49)} = 1$. The place *p-49* will be used for synchronization of the start of the next cycle of *k*-tournament in the current generation. The places *p-23, p-24* and *p-25* refer to three random chosen chromosomes from the selection set of chromosomes.

The second part of the *k*-tournament selection is shown on Figure 3. The main goal is to choose two chromosomes of selected three chromosomes which have larger value of fitness function. These two chromosomes become two parents *(p-35, p-36)* for crossing and making new two offspring. There are two enabled transitions *{t-23, t-25}*. Only one of them will be fired because we are using conflict in UPN. So there are two cases.

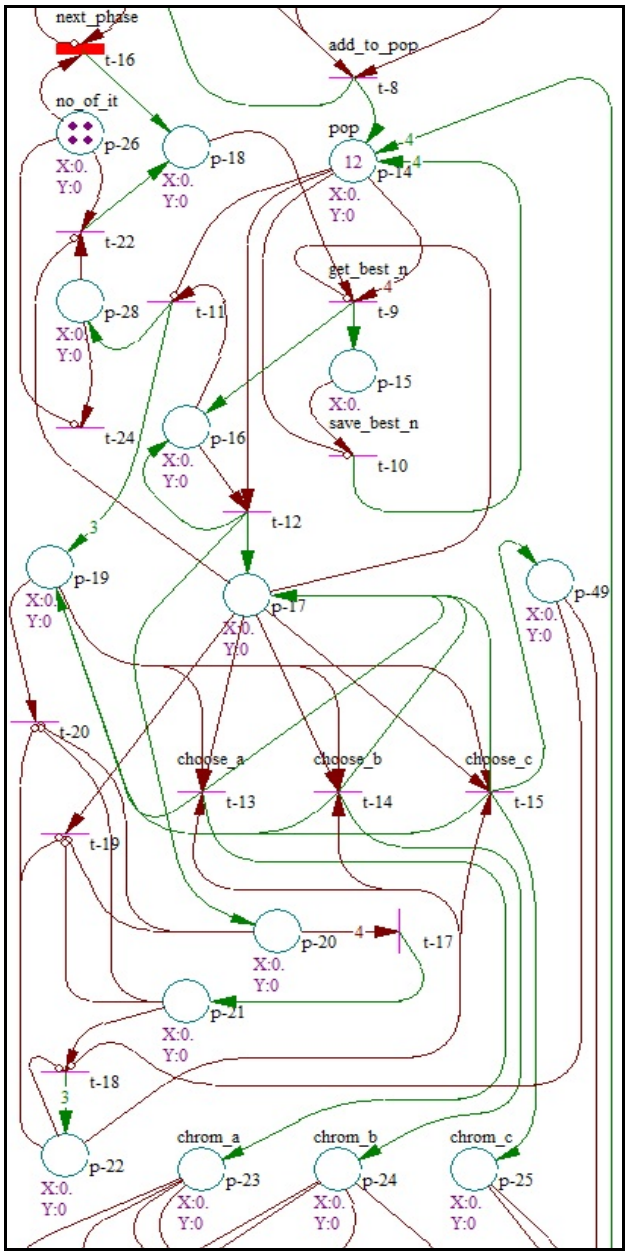


Figure 2. PN model of genetic algorithm which searches the maximum of a given function, k-tournament selection, part 1.

The first case is a firing of the transition *t-25* means that fitness of *chrom_a* is greater than the fitness of *chrom_b*. Now, the transitions *{t-29, t-26, t-27}* will fired. By firing the transition *t-29* the first parent (*chrom_1*) becomes *chrom_a*. By firing of the transition *{t-26, t-27}* the values of *chrom_b* and *chrom_c* will copied into the places *p-31* and *p-32* and *(cb', cc')* so the transitions *{t-30, t-31}* become enabled and only one of them will fired because of conflict in UPN. Firing of the transition *t-30* means that fitness of *chrom_b* is greater than the fitness of *chrom_c*, otherwise, firing of the transition *t-31* means the opposite. For the first case by firing of the transition *t-32* the second parent (*chome_2*) becomes *chrome_b* and for the second case by firing of the transition *t-33 chome_2* becomes *chrome_c*.

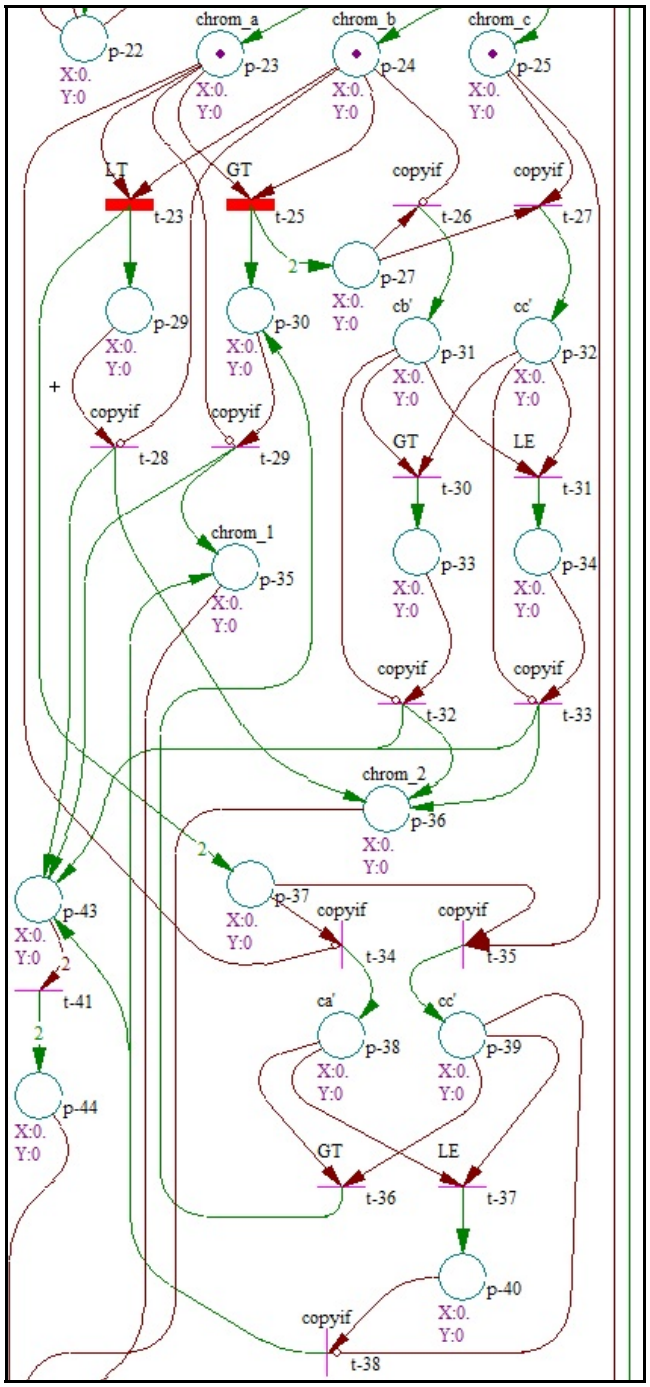


Figure 3. UPN model of genetic algorithm which searches the maximum of a given function, k-tournament selection, part 2.

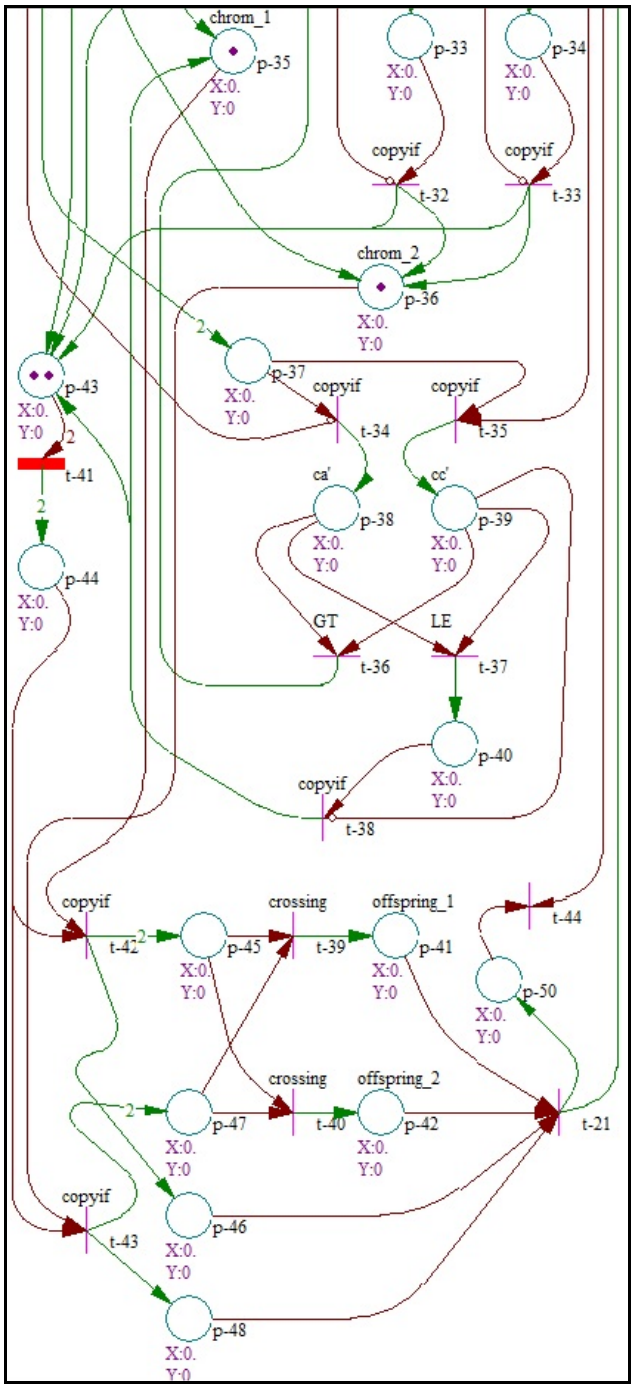


Figure 4. UPN model of genetic algorithm which searches maximum of a given function, crossing and mutation.

The second case is a firing of the transition *t-23* means that fitness of *chrom_b* is greater than the fitness of *chrom_a*. Now, the transitions *{t-28, t-34, t-35}* will fired. By firing the transition *t-28* the second parent (*chrom_2*) becomes *chrom_b*. By firing of the transition *{t-34, t-35}* the values of *chrom_a* and *chrom_c* will copied into the places *p-38* and *p-39* and *(ca', cc')* so the transitions *{t-36, t-37}* become enabled and only one of them will fired because of conflict in UPN. Firing of the transition *t-36* means that fitness of *chrom_a* is greater than the fitness of *chrom_c*, otherwise, firing of the transition *t-37* means the opposite. For the first case by firing of the transition *t-36* the first (*chome_1*) becomes *chrome_a* and for the second case by firing of the transition *t-37* *chome_1* becomes *chrome_c*.

After the end of these two cases we have a new marking shown on Figure 4. At this moment the transition *t-41* is enabled. The firing sequence *{t-41}* → *{t-42, t-43}* → *{t-39, t-*

*40}* → *{t-21}* → *{t-44}* → *{t-44}* → *{t-18}* does: synchronization for

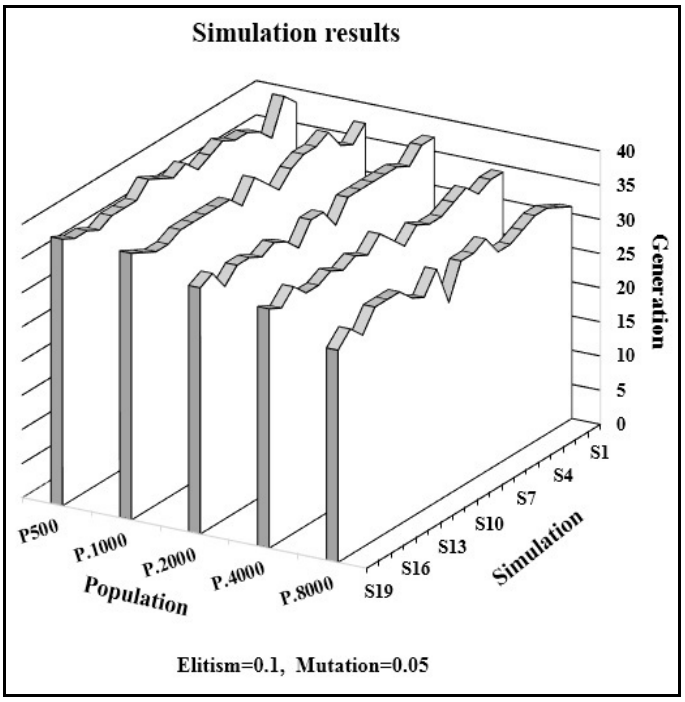


Figure 5. Simulation results

starts crossing then mutation and save new 4 chromosomes and finally starts the next iteration of the *k*-tournament for the current generation, respectively. [10]. When all cycles of the *k*-tournament will end then the selection set will cleared by successive firing of transition *t-19*. After this if $\mu_{(p-26)}>0$ the next selection for the current generation starts. Finally, the UPN model reaches dead node and execution of the UPN model stops.

## IV. THE UPN MODEL ANALYSIS

The UPN model which is presented in the paper has observed and implemented in software program using *C++* language. The behavior of the UPN model shown in previous chapter can be checked by executing this software program which refers to the model.

This software program has been tested for various values of input parameters. There are some simulation results shown on

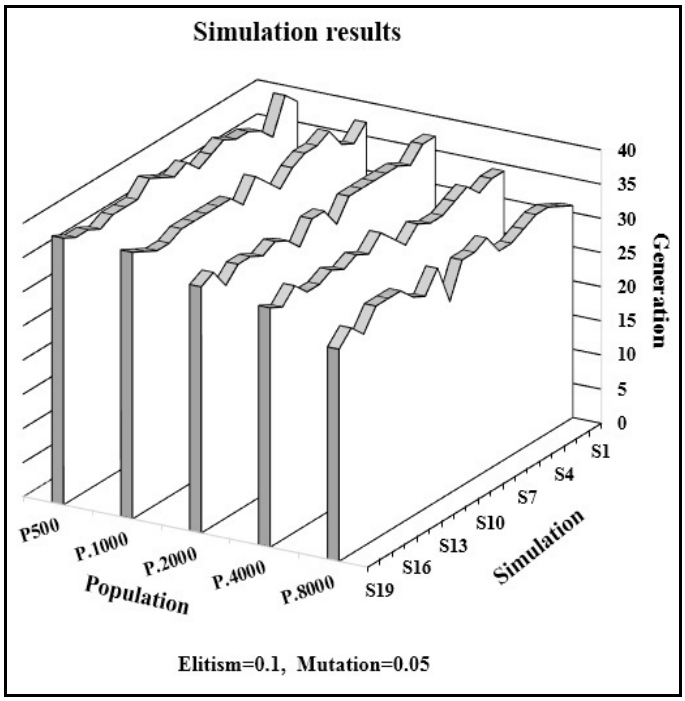


Figure 5. These results were obtained for five values of the population size (*500*, *1000*, *2000*, *4000* and *8000* chromosomes) after *20* independent simulations per each population size. Other input parameters are: interval of x coordinate ($x\in[-10,10]$), interval of y coordinate ($y\in[-10,10]$), a given function $f(x,y)=-x^2-y^2$, *elitism=0.1* and *mutation=0.05*. Height on the Figure 5. Simulation results refers to an order number of the generation in which minimum of the given function is reached.

The results given by the implemented software program were expected and confirm the validity of a given UPN model.

## V. CONCLUSION

This paper presents Upgraded Petri net model of genetic algorithm which searching for maximum of a given function in given intervals. This model represents: chromosome encoding, the first population generating, fitness evaluation, *k*-tournament selection, crossing, mutation and the next population generating. Also, parallelism, synchronization and UPN conflicts are presented in the model by using UPN formal theory shown in this paper. The UPN model was developed from the initial version to the final version through interactive monitoring of its execution from the initial marking up to the dead node. After careful analysis, the UPN model is transformed into the real software program. Suitability of given UPN model was checked by execution of the real program. UPN can be used for creating models based on genetic algorithms due to their parallel nature. Original software for modeling and simulations of Upgraded Petri Nets, PeM (Petri Net Manager), is developed and used for all models described in this paper.

## ACKNOWLEDGEMENTS

This research was supported by the Serbian Ministry of Education and Science, Republic of Serbia, Project no. III 44006

## REFERENCES

[1] Michel Diaz, "Petri Nets: Fundamental Models, Verification and Applications", John Wiley & Sons 2010.

[2] Girault, C. and Valk, R., "Petri Nets for Systems Engineering - A Guide to Modeling, Verification, and Applications", Springer-Verlag, Berlin, Heidelberg, New York, 2003.

[3] Perica Štrbac, Gradimir V. Milovanović, "Upgraded Petri net model and analysis of adaptive and static arithmetic coding", Elsevier: Mathematical and Computer Modelling Vol. 58, http://dx.doi.org/10.1016/j.mcm.2013.06.001, pp. 1548–1562, 2013.

[4] D. Dumitrescu, B. Lazzerini, L. C. Jain, and A. Dumitrescu, "Evolutionary Computation," Boca Raton, FL: CRC Press, 2000.

[5] A. J. Umbarkar M. S. Joshi, "Dual Population Genetic Algorithm (GA) versus OpenMP GA for Multimodal Function Optimization", International Journal of Computer Applications, DOI: 10.5120/10744-5516, Vol.64 No.19. pp. 29-36, 2013.

[6] A. V. Eremeev, "Genetic algorithm with tournament selection as a local search method", Diskretn. Anal. Issled. Oper., Vol.19, No.2, pp. 41–53, 2012.

[7] Chang Wook Ahn and R. S. Ramakrishna, "Elitism-Based Compact Genetic Algorithms," IEEE Transactions on Evolutionary Computation, Vol. 7, No. 4, pp. 367-385, August 2003.

[8] S. Yang, "Genetic Algorithms with Elitism-based Immigrants for Changing Optimization Problems," Lecture Notes in Computer Science, Volume 4448/2007, pp. 627-636, DOI: 10.1007/978-3-540-71805-5_69, 2007.

[9] Mishra K.K., Tiwari S., Kumar A., Misra A.K., "An approach for mutation testing using elitist genetic algorithm," International Conference on Computer Science and Information Technology 3rd IEEE (ICCSIT), pp. 426-429, 2010.

[10] Magdalena Smetek, Bogdan Trawinski, "Investigation of Genetic Algorithms with Self-Adaptive Crossover, Mutation, and Selection", Lecture Notes in Computer Science: Hybrid Artificial Intelligent Systems, Volume 6678, 2011, pp 116-123, 2011.