INFOTEH-JAHORINA Vol. 12, March 2013.

Permuted Sequence Implemented as an Upgraded Petri Net for Solving N-Queens Problem

Perica S. Štrbac, Jovo Arežina
Faculty of Computer Science
Belgrade, Serbia
strbac68@gmail.com jovo.arezina@gmail.com

Zoran Banjac
School of Electrical Engineering and Computer Science
Belgrade, Serbia
zoran.banjac@viser.edu.rs

Abstract — **The objective of this paper is modeling, simulation and analysis of Upgraded Petri Net (UPN) model which implements permuted sequence for solving N-queens problem. The UPN was developed for simulation and analysis of processes, particularly at the register transfer level. Original software for modeling and simulations of UPN, PeM (Petri Net Manager), is developed and used for all models described in this paper. This software supports: UPN formal theory, graphical modeling, simulation and analysis of an UPN model. This paper includes UPN theory, the UPN models of high level representation for solving N-queens problem and model which includes three sub models based on alternating position crossover (AP), order based crossover (OX2) and partially mapped crossover (PMX), respectively, their simulation and analysis. The executions of UPN are based on parallel firing of a group of transitions. The suitability of UPN for modeling of the systems based on permuted sequence is examined and established.**

Key words — Permuted Sequence; Upgraded Petri Net; Genetic Algorithm; AP; OX2; PMX; Modeling; Simulation; Analysis;

I. Introduction

Upgraded Petri nets are a formal mathematical apparatus which enables modeling, simulation and process analysis [1]. They enable interactive monitoring of process operations and its gradual improvement from the initial phase, all the way to the final version.

The hierarchical structure of an UPN gives wide possibilities for abstraction. This feature of the UPN provides the model implementation consisting 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). Unlike other classes of Petri Net, the UPN can be able to generate numerical results through its model execution at RTL level.

This paper presents the usage of UPN in the example of modeling, simulation and analysis of a genetic algorithm [3] which solves n-queen problem where chromosomes coded as permuted sequence. A chromosome is determined as a permuted sequence where one element in the sequence refers to the column position and its index refers to the appropriate row position of one queen on the chessboard. Main goal is to place N-queens on N×N chess board in such a way that no one attacks each other.

The algorithm finds one or more solutions using basic genetic algorithm principles. Initial population of random queen positions is generated.

The chromosomes in the population will be evaluated by fitness function, recombined and mutated until the problem is solved.

The fitness function is determined by the smallest number of conflicts attack between queens on the chessboard.

The recombination is done by using: alternating position crossover (AP), order based crossover (OX2) and partially mapped crossover (PMX) between two permuted sequences to generate two different offspring.

The mutation is done by exchanging positions of two values in permuted sequence.

During a life cycle the population is sorted by fitness, some 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 rest chromosomes.

As an example, the first UPN model implements the high level representation of the solution and the second UPN model implements the recombination process.

For given input parameters determined by: initial marking, multiplicity of arcs, transition function, transition firing level, place attributes, and UPN conflict solving the model through its execution can check the recombination process [1].

II. An UPN formal theory

Upgraded Petri nets formal theory is based on functions [1]. 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 μ_i it can be used the sign $\mu(p_i)$.

The timing function θ 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 α_j operation assigned to the t_j transition executes where the order of operands in the operation α_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 α_j must be equal to true. (3)

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

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

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 FOR SOLVING N-QUEENS PROBLEM

In this section two UPN models are described. The first one refers to high level representation of the genetic algorithm which has used for solving N-queens problem and the second one refers to recombination of chromosomes implemented as permuted sequence inside N-queens solution. [3], [4], [5], [6], [7], [8].

The first UPN model of a high level representation of the genetic algorithm is shown in Error! Reference source not found.. Initial marking is $\mu_{(p-8)} = 1$. By firing of enabled t-8 transition, which models process of chromosome encoding, a new marking becomes $\mu_{(p-1)} = 1$, $\mu_{(p-8)} = 0$. This state of the model refers to ready state for generating the first population. By firing of transition t-1, a new marking becomes $\mu_{(p-2)} = 1$, $\mu_{(p-1)} = 0$ and transition t-2 becomes enabled. Further, by firing of transition t-2, which models calculation of fitness for chromosomes in the population, a new marking is $\mu_{(p-3)} = 1$, $\mu_{(p-2)} = 0$ and transition t-3 becomes enabled. Now, firing of this transition models selection process with respect to fitness. Transition t-4 becomes enabled, and marking is $\mu_{(p-4)} = 1$, $\mu_{(p-3)} = 0$. The next event is process of the recombination which was modeled by firing of transition t-4. At this point transition t-5 is enabled and marking is $\mu_{(p-5)} = 1$, $\mu_{(p-4)} = 0$. By firing of enabled t-5 transition, which models process of mutation the population, a new marking become $\mu_{(p-6)} = 1$, $\mu_{(p-5)} = 0$ and transitions t-2 and t-7 become enabled. This conflict situation means that there are two ways. The first one is firing of transition t-6, which means that main goal is not reached, and the second is firing of transition t-7 which means that main goal reached. If the first one case occurs the fitness evaluation over the new population starts, and the iteration of genetic algorithm started. In the opposite, if the transition t-7 fired there is at least one solution in the population ($\mu_{(p-7)} = 1$). Finally, the results will be decoded and converted into graphic representation of solutions.

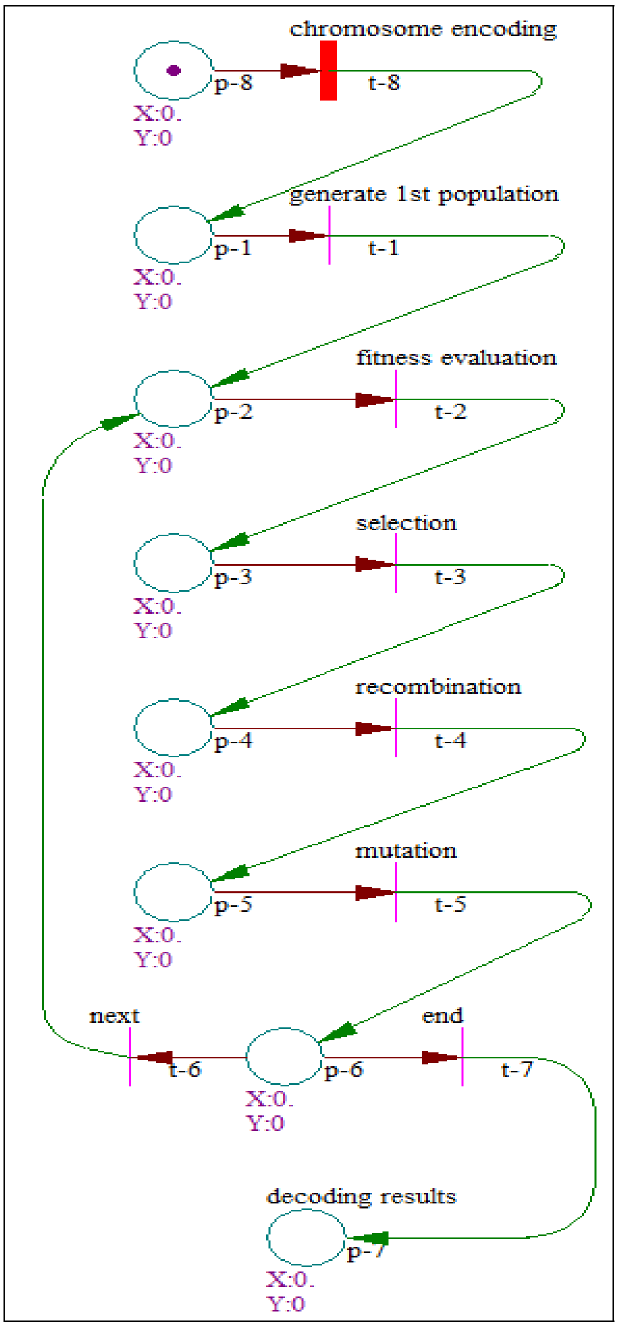

Figure 1. High level representation UPN model for solving N-queens problem

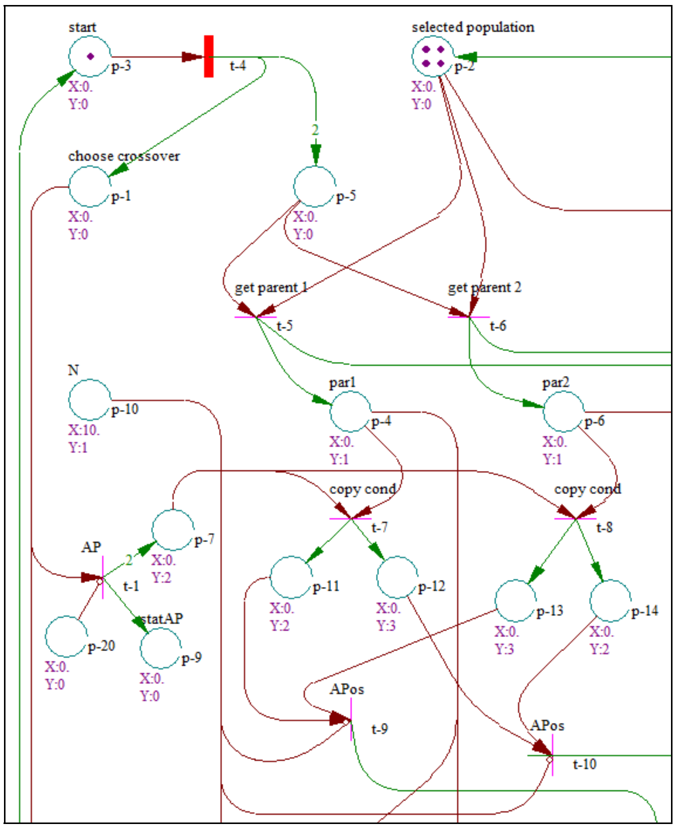

Figure 2. UPN model of recombination of encoded sequence (part 1)

The second model refers to the recombination of two encoded sequences. This model contents of the three sub models based on alternating position crossover (AP), order based crossover (OX2) and partially mapped crossover (PMX), respectively.

The first part of the second model is shown in Error! Reference source not found.. Initial marking is $\mu(p\text{-}3) = 1$, $\mu(p\text{-}2) = 4$. The first value enables transition *t-4*, so UPN can starts. The second value represents how many chromosomes are in the selected population set. By firing of transition *t-4*, a new marking becomes $\mu(p\text{-}1) = 1$, $\mu(p\text{-}5) = 2$ (because of multiplicity of output arcs $B(t\text{-}4,p\text{-}5)=2$), $\mu(p\text{-}3) = 1$, $\mu(p\text{-}2) = 4$ and transitions *t-1, t-2, t-3, t-5* and *t-6* become enabled. Transition *t-5* get the first parent from selected population, while transition *t-6* get the second parent. These two transitions will be fired in parallel. On the other side, only one transition of the parallel set of transitions *{t-1, t-2, t-3}* will be fired, because of $\mu(p\text{-}1) = 1$and $F(t\text{-}1, p\text{-}1) = F(t\text{-}2, p\text{-}1) = F(t\text{-}3, p\text{-}1) = 1$. Each of the places *p-20, p-21* and *p-22* have inhibitor arc to the appropriate transitions *t-1, t-2* and *t-3,* respectively. On this way we can enable or disable any combination of crossover (AP, OX2 and PMX) by setting appropriate marking i.e. $\mu(p\text{-}20) = 1$disables AP crossover. In the model only one of three algorithms will be used per iteration. Every single algorithm has probability of 1/3 to be used per iteration.

By firing of transitions *{t-5, t-6, t-1}* places *p-4* and *p-6* take the first and second parent from the selected population, respectively, and now marking $\mu(p\text{-}2) = 2$, $\mu(p\text{-}1) = 0$, $\mu(p\text{-}16) = \mu(p\text{-}17) = 1$, On the other side marking $\mu(p\text{-}7) = 2$ and set of transitions *{t-7, t-8, t-11, t-12}* is enabled and increment of the marking $\mu(p\text{-}9)$ = 1 occurs. The last marking represents statistic data how many times the AP crossover was used per iteration. Places p-16 and p-17 prepare to save selected parents into the new population.

By firing of transitions {t-7, t-8} places p-11 and p-12 have copies of the first parent while places p-13 and p-14 have copies of the second parent, on the other side by firing of transitions {t-11, t-12} selected parents is added to the new population (these transitions are shown on **Fig. 3**). At this moment the marking $\mu(p\text{-}7) = \mu(p\text{-}4) = \mu(p\text{-}6) = 0$ and $\mu(p\text{-}11) = \mu(p\text{-}12) = \mu(p\text{-}13) = \mu(p\text{-}14) = 1$, $\mu(p\text{-}16) = \mu(p\text{-}17) = 0$, $\mu(p\text{-}18) = 2$ (the new population grows up by 2 parents) $\mu(p\text{-}19) = 2$. The transitions {t-9, t-10} are enabled. These transitions have transition function *APos* which creates offspring by using alternating position crossover. This *TF* has transition firing level equal to 3. The inputs into this function are as follows: length of the permuted sequence (implemented as place p-10 which has an attribute x=N and an attribute y=1), first permuted sequence (implemented as place which has an attribute x equal to address of the first permuted sequence and an attribute y=2) and the second permuted sequence (implemented as place which has an attribute x equal to address of the second permuted sequence and an attribute y=3).

By firing of transitions {t-9, t-10} a new marking becomes $\mu(p\text{-}11) = \mu(p\text{-}12) = \mu(p\text{-}13) = \mu(p\text{-}14) = 0$, $\mu(p\text{-}18) = \mu(p\text{-}19) = 2$ and $\mu(p\text{-}16) = \mu(p\text{-}17) = 1$. Now both offspring are placed into the places p-16 and p-17, respectively.

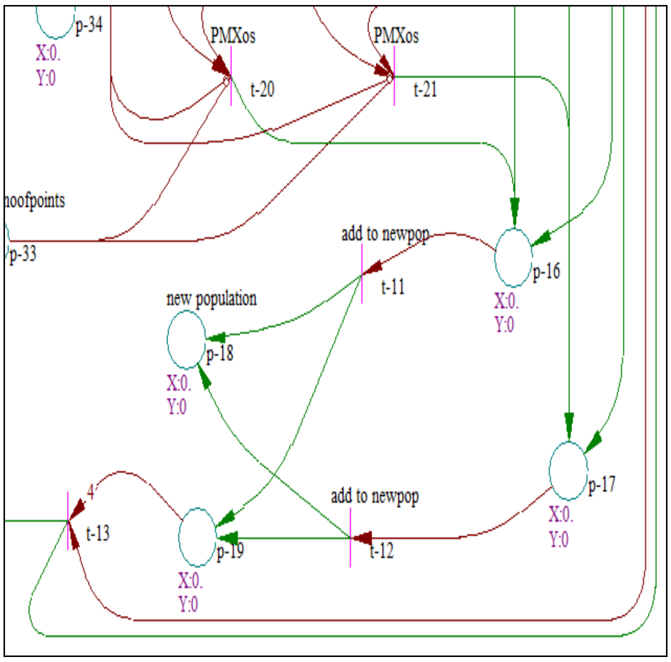

Figure 3. UPN model of recombination of encoded sequence (part 3)

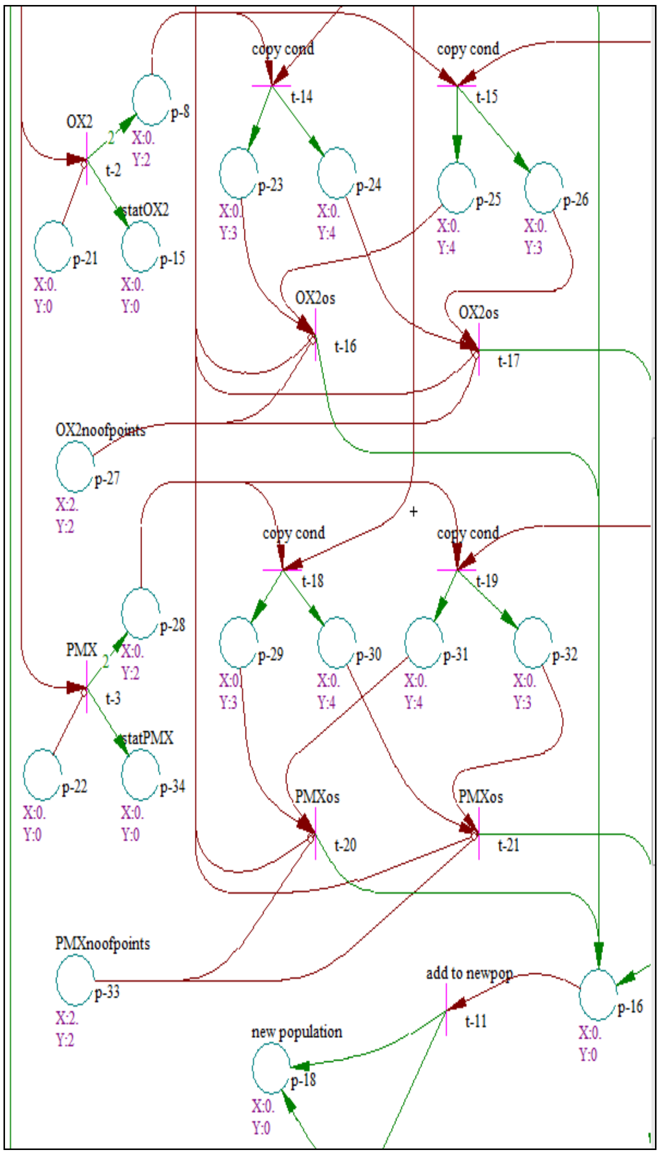

Figure 4. UPN model of recombination of encoded sequence (part 2)

By firing of transitions *{t-11, t-12}* both offspring are added to the new population. At this moment a new marking becomes $\mu(p\text{-}18) = 4$, $\mu(p\text{-}19) = 4$ and $\mu(p\text{-}16) = \mu(p\text{-}17) = 0$. The transition *t-13* is enabled. This means that there are parents in the selected population and new iteration of mating will start. By firing of the transition *t-13* a new marking becomes $\mu(p\text{-}18) = 4$, $\mu(p\text{-}19) = 0$, $\mu(p\text{-}3) = 1$ (start new iteration), and $\mu(p\text{-}2) = \mu(p\text{-}2) -1+1$ (preserve marking into the model).

Now the next iteration starts. In the second iteration let the OX2 is chosen (see **Fig. 4**). In that case after firing the set of transitions *{t-2, t-5, t-6}* two selected parents will be copied into places *p-16* and *p-17* and there is no chromosomes into the selected population, and statistic data for OX2 (place p-15) will be updated. By firing of the set of transitions *{t-14, t-15, t-11, t12}* selected parents will be copied into the places *p-23*, *p-24* for the first parent and into the places *p-25*, *p-26* for the second parent and also parents from places *p-16* and *p-17* will be added to the new population. .

The transitions *t-16* and *t-17* have transition function *OX2os* which creates offspring by using order based crossover. This *TF* has transition firing level equal to 4. The inputs into this function are as follows: length of the permuted sequence (implemented as place *p-10* which has an attribute *x=N* and an attribute *y=1*), number of crossing points (implemented as place *p-27* which has an attribute *x=2* and an attribute *y=2*), first permuted sequence (implemented as place which has an attribute x equal to address of the first permuted sequence and an attribute y=3) and the second permuted sequence (implemented as place which has an attribute x equal to address of the second permuted sequence and an attribute y=4). After firing of transitions *{t-16, t-17}* both offspring will be placed into the places *p-16* and *p-17* and after firing of transitions {t-11, t-12} new population has 8 chromosomes and model reached death node. That means that there is no enabled transition in the net i.e. in this case selected population is empty.

If in the second iteration was selected partially mapped crossover the situation would be similar as in the case of OX2. Equivalent places are *p-10, p-15, p-8, p-23, p-24, p-25, p-26* as *p-10, p-34, p-28, p-29, p-30, p31* and *p-32,* respectively. Equivalent transitions are *t-2, t-14, t-15, t-16, t-17* as *t-3, t-18, t-19, t20* and *t-21*, respectively. The transitions *t-20* and *t-21* have transition function *PMXos* which creates offspring by using partially mapped crossover. This *TF* has transition firing level equal to 4. The inputs into this function are as follows: length of the permuted sequence (implemented as place *p-10* which has an attribute *x=N* and an attribute *y=1*), number of crossing points (implemented as place *p-27* which has an attribute *x=2* and an attribute *y=2*), first permuted sequence (implemented as place which has an attribute x equal to address of the first permuted sequence and an attribute y=3) and the second permuted sequence (implemented as place which has an attribute x equal to address of the second permuted sequence and an attribute y=4).

IV. AN UPN MODEL EXECUTION

By executing the UPN model for given initial marking (shown on **Fig. 1**) the following sequence of transitions firing will happen: *{t-8}*→*{t-1}*→*{t-2}*→*{t-3}*→*{t-4}*→*{t-5}*→*{t-6, t-7}*.

By executing the UPN model for given initial marking shown on UPN model of recombination of encoded sequence (**Fig.** 2, **Fig.** 3 and **Fig.** 4) the next sequence of transitions firing will happen: the first *{t-4}* then one of the three sets of the transitions *{t-1, t-5, t-6}*, *{t-2, t-5, t-6}*, *{t-3, t-5, t-6}* will fired.

After the sequence *{t-1, t-5, t-6}* the next sequence will occur: *{t-7, t-8, t-11, t-12}*→ *{t-9, t-10}*→ *{t-11, t-12}*→

{t-13}→ *{t-4}*. This sequence refers to alternating position crossover.

When the sequence *{t-2, t-5, t-6}* occurred the next sequence is: *{t-14, t-15, t-11, t-12}*→ *{t-16, t-17}*→ *{t-11, t-2}*→ *{t-13}*→ *{t-4}*. This sequence refers to order based crossover.

After the sequence *{t-3, t-5, t-6}* the next sequence will occur: *{t-18, t-19, t-11, t-12}*→ *{t-20, t-21}*→ *{t-11, t-12}*→ *{t-13}*→ *{t-4}*. This sequence refers to partially mapped crossover.

When UPN model reached state that there is no parents into the selected population ($\mu(p\text{-}2) = 0$) the transition t-13 will stay disabled after execution of the preceding sequences (up to *{t-11, t-12}*). This represents the dead node which in shown model refers to end of process of generating the new population.

V. The UPN model analysis

The UPN models which are presented in the paper show great potential for parallel operation through their parallel firing of transitions. This has observed and implemented in software program using *C++* language.

The results given by the UPN models are expected and can be checked by executing the program which refers to the models.

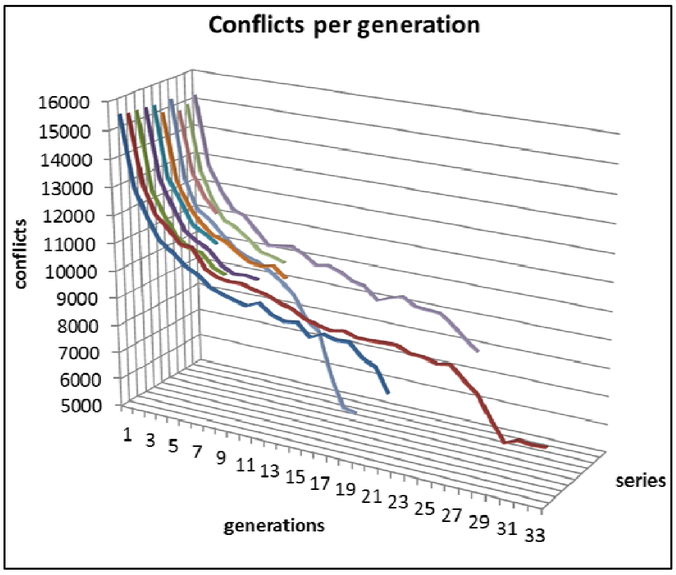

Figure 5. Total conflicts per generation

The changing of total conflicts in all chromosomes in the population per generation is shown on **Fig. 5**. These results have obtained with the parameters as follows: the population includes 1000 chromosomes, mutation factor is equal to 0.8, maximum number of generation is set to 1000 with chess board size is equal to 12×12 and with randomly choosing one of three crossover algorithm per generation. With respect to the second UPN model (see **Fig.** 2, **Fig.** 3. and **Fig.** 4) it is possible to set up which algorithm will be selected for the current generation. It can be selected one, two or all three algorithms which competing to be used over the current generation. All three crossover algorithms were used (alternating position crossover, order based crossover and partially mapped crossover) to produced data which shown on **Fig. 5**.

VI. Conclusion

Two Upgraded Petri net models are presented. The first one refers to high level of presentation of genetic algorithm which solve N-queens problem. The second model refers to the recombination of two encoded sequences which includes three sub models. The sub models are refer to: alternating position crossover (AP), order based crossover (OX2) and partially mapped crossover (PMX). Given UPN models can generate results for given input data and initial marking. Suitability of given UPN model was checked by execution of the model and the results generated dynamically during this execution. UPN can be used for creating models based on genetic algorithms due to their parallel nature and the need for random inputs. UPN can be able to generate numerical results through its model execution at RTL level. This level is suitable for analyzing hardware implementation of the model and also for checking given results. 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] Perica Strbac, Milan Tuba, “An Upgraded Petri Net Model, Simulation and Analysis of An 8x8 Sub-Image for JPEG Image Compression,” Recent Advances in Applied Informatics and Communications, Recent Advances in Computer Engineering, WSEAS Press, ISBN 978-960-474-107-6, Moscow, Russia, pp. 167-172, August 2009.

[2] J. L. Peterson, “Petri Net Theory and Modeling of Systems”, Englewood Cliffs , Prentice Hall, New York 1981.

[3] K. F. Man, K. S. Tang and S. Kwong, “Genetic Algorithms: Concepts and Applications,” IEEE Transactions on Industrial Electronics, Vol. 43, No. 5, pp. 519-534, October 1996.

[4] 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.

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

[6] 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.

[7] W. Zhong, J. Liu, and L. Jiao, “Evolutionary Agent for n-Queen Problems”, Lecture Notes in Computer Science, 2005, Volume 3612/2005, pp. 366-373, 2005.

[8] 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.