18

Some Results on Evolving Cellular Automata Applied to the Production Scheduling Problem

Tadeusz Witkowski[1], Arkadiusz Antczak[1],
Paweł Antczak[1] and Soliman Elzway[2]
[1]*Warsaw University of Technology*
[1]*Nasser International University*
[1]*Poland*
[2]*Libya*

1. Introduction

Production scheduling is the process of allocating the resources and then sequencing of task to produce goods. Allocation and sequencing decision are closely related and it is very difficult to model mathematical interaction between them. The allocation problem is solved first and its results are supplied as inputs to the sequencing problem. High quality scheduling improves the delivery performance and lowers the inventory cost. They have much importance in this time based competition. This can be achieved when the scheduling is done in acceptable computation time, but it is difficult because of the NP-hard nature and large size of the scheduling problem.

Based on the machine environment, sequence of operations for the jobs, etc. , the production scheduling problem is divided into the different types: one stage, one process or single machine; one stage, multiple processor or parallel machine; flow shop, job shop, open shop; static and dynamic etc. Job shop is a complex shop where there are finite number of machines, jobs and operation to be done on jobs. There is no direction of flow for jobs. The scheduling is done based on the selection of machine k to process an operation i on job j. Each job can be processed on a machine any number of times. Flexible job-shop scheduling problem (FJSP) extends the JSP by allowing each operations to be processed on more than machine. With this extension, we are now confronted with two subtask: assignment of each operation to an appropriate machine and sequencing operations on each machine.

In the literature, different approaches (tabu search, simulated annealing, variable neighborhood, particle swarm optimization, clonal selection principle etc.) have been proposed to solve this problem (Fattahi,et al., 2007; Kacem, et al., 2002; Liu, et al., 2006; Ong, et. al., 2005; Preissl, 2006; Shi-Jin, et al., 2008; Tay, et al., 2008; Yazdani, et al., 2009). The genetic algorithms (GA), genetic programming, evolution strategies, and evolutionary programming for scheduling problem are described in (Affenzeller, et. al., 2004; Back, et al., 1997; Beham, et al., 2008; Koza, 1992; Mitchell, et. al., 2005; Zomaya, et. al., 2005; Stocher, et. al., 2007; Winkler, et. al., 2009), and cellular automata are presented in (De Castro, 2006; Tomassini, 2000; Seredyński, 2002). Using GA algorithm to behavior in cellular automata (CA), evolutionary design of rule changing CA, and other problems are described in (Back,

et. al., 2005; Kanoh, et. al., 2003; Martins, et. al., 2005; Das, et. al., 1994; Sipper, 1997,1999; Subrata, et. al., 2003; Sahoo, et. al. 2007).

The difficulty of designing cellular automatons transition rules to perform a particular problem has severely limited their applications.

In (Seredyński,et. al., 2002) evolution of cellular automata-based multiprocessor scheduling algorithm is created. In learning mode a GA is applied to discover rules of CA suitable for solving instances of a scheduling problem. In operation mode discovered rules of CA are able to find automatically an optimal or suboptimal solution of the scheduling problem for any initial allocation of a program graph in two-processor system graph.

The evolutionary design of CA rules has been studied by th EVCA group in detail. A genetic algorithm GA was used to evolve CAs for the two computational tasks. The GA was shown to have discovered rules that gave rise to sophisticated emergent computational strategies. Sipper (1999) has studied a cellular programming algorithm for 2-state non-uniform CAs, in which each cell may contain a different rule. The evolution of rules is here performed by applying crossover and mutation. He showed that this method is better than uniform (ordinary) Cas with a standard GA for the two tasks. In Kanoh (2003) was proposed a new programming method of cellular computers using genetic algorithms. Authors considered a pair of rules and the number of rule iterations as a step in the computer program. This method im meant to reduce the complexity of a given problem by dividing the problem into smaller ones and assigning a distinct rule to each.

This study introduces an approach to solving evolutionary cellular automata-based FJSP. In this paper genetic programming is applied in this algorithm – rule tables undergo selection and crossover operations in the populations that follow.

The paper is organized as follows. Section 2 gives formulation of the problem. A formal definition of CA is described in section 3. Section 4 explains the details of the evolving CA-based production scheduling. Section 5 shows the computational results and the comparison of CA and GA for finding solutions in FJSP is presented. Some concluding remarks are given in section 6.

2. Problem formulation

The FJSP is formulated as follows. There is a set of jobs $Z = \{Z_i\}$, $i \in I$, where $I = \{1, 2, ..., n\}$ is an admissible set of parts, $U = \{u_k\}$, $k \in 1, m$, is a set of machines. Each job Z_i is a group of parts Π_i of equal partial task p_i of a certain range of production. Operations of technological processing of the i-th part are denoted by $\{O_{ij}\}_{j=\xi}^{H_i}$. Then for Z_i , we can write $Z_i = (\Pi_i \{O_{ij}\}_{j=\xi}^{H_i})$, where $O_{ij} = (G_{ij}, t_{ij}\ (_N))$ is the j-th operation of processing the i-th group of parts; ξ_i is the number of operation of the production process at which one should start the processing the i- th group of parts; H_i is the number of the last operation for a given group; G_{ij} is a group of interchangeable machines that is assigned to the operation O_{ij}; G is a set of all groups of machines arose in the matrix $||\{ Z_i \}||$; $t_{ij}\ (_N)$ is an elementary duration of the operation O_{ij} with one part d_i that depends on the number of machine N in the group (on the specified operations); t'_{ij} is the duration of set up before the operation O_{ij}; N_{gr} is the number of all groups of machines. The most widely used objective is to find feasible schedules that minimize the completion time of the total production program, normally referred to as makespan (C_{max}).

3. Formal definition cellular automata

A *d*-dimensional CA consists of a finite or infinite *d*-dimensional grid of cells, each of which can take on a value from a finite, usually small, set of integers. The value of each cell at time step $t + 1$ is a function of the values of small local neighborhood of cells at time *t.* The cells update their state simultaneously according to a given local rule. Formally, a CA can be defined as a quintuple (De Castro, 2006)

$$C = <S, s_0, G, d, f>$$

where S is a finite set of states, $s_0 \in S$ are the initial states of the CA, G is cellular neighborhood, $d \in Z+$ is the dimension of C, and f is the local cellular interaction rule, also referred to as the *transition function* or *transition rule*. Given the position of a cell **i**, where **i** is an integer vector in a d-dimensional space ($\mathbf{i} \in Zd$), in a regular d-dimensional uniform lattice, or grid, its neighborhood G is defined by

$$G_{\mathbf{i}} = \{\mathbf{i}, \mathbf{i}+ \mathbf{r}_1, \mathbf{i}+ \mathbf{r}_2,..., \mathbf{i}+ \mathbf{r}_n\}$$

where n is a fixed parameter that determines the neighborhood size, and $\mathbf{r}_j$ is a fixed vector in the d-dimensional space. The local transition rule f

$$f : Sn \rightarrow S$$

maps the state $s_{\mathbf{i}} \in S$ of a given cell **i** into another state from the set S, as a function of the states of the cells in the neighborhood $G_{\mathbf{i}}$. In a uniform CA, f is identical for all cells, whereas in nonuniform CA, f may differ from one cell to another, i.e., f depends on **i**, $f_{\mathbf{i}}$. For a finite-size CA of size N, where N is the number of cells in the CA, a configuration of the grid at time t is defined as

$$C(t) = (s_0(t), s_1(t),...,s_{N-1}(t))$$

where $s_{\mathbf{i}}(t)$ is the state of cell **i** at time t**.** The progression of the CA in time is then given by the iteration of the global mapping F

$$F : C(t) \rightarrow C(t+1), \quad t = 0,1,...$$

Through the simultaneous application in each cell of the local transition rule f, the global dynamics of the CA can be described as a directed graph, referred to as the CA' s state space. One-and bi-dimensional CA are the most usually explored types of CA. In the one-dimensional case, there are usually only two possible states for each cell, $S = \{0,1\}$. Thus, f is a function $f : \{0,1\}$ n $\rightarrow \{0,1\}$ and the neighborhood size n is usually taken to be $n = 2r+1$ such that

$$s_{\mathbf{i}}(t+ 1) = (s_{\mathbf{i}-r}(t),...,s_{\mathbf{i}}(t),...,s_{\mathbf{i}+r}(t))$$

where $r \in Z+$ is a parameter, known as the radius, representing the standard one-dimensional cellular neighborhood.

4. Evolving cellular automata for FJSP

4.1 Algorithm for evolving CA for FJSP

The general working principle of evolutionary algorithms is based on a program loop that involves implementations of the operators mutation, recombination, selection, and fitness evaluation on a set of candidate solutions for a given problem.

The algorithm which generates the schedule bases on two CAs. One is responsible for construction sequencing operations on individual parts, and the other for the allocation of machines to operation with interchangeably group machines.

The crossover operation is realized on the current and previous population using a definite number of the best rules in the two above-mentioned populations. Half of that definite number is taken from the current population, and the other half from the previous one.

Depending on the generated value and the determined intensity the re-writing of the values from the current table to the previous one or vice versa takes place (no operation is also possible). During the algorithm operation in a loop state changes of the CA are executed basing on the transition tables.

They define the change of the current position of an element in the state table on the basis of its current value. The repetition of the operation causes changes in the CA state, which defines the sequence of technological operations and machines used. On the basis of those state tables a proper schedule is generated (reservation of machines).

Genetic algorithm is applied in the CA algorithm – rule tables undergo selection and crossover operations in the populations that follow.

The algorithm sequences the technological operations on a given set of parts of different kinds using evolving CAs. This is realized with the use a genetic algorithm which performs a selection of the so-called transition tables (i.e. rule tables, state change tables) of the two cellular automata whose functions are described above.

The input parameters are: the number of the population of automata transition tables (rule tables - RT), the number of populations, the number of transitions, the hybrid coefficient (the number of the tables in the populations being crossed over with a given probability), the hybridization intensity (the probability of the crossover operation on given elements of the tables). Fig. 1 shows the flowchart of evolving CA used to create schedules.

The algorithm is based on two cellular automata: a) determination of machine allocation from the interchangeable group for individual operations, b) determination of part sequence for individual operations.

The CA state change is realized as follows. Let o_{i1},o_{i2} position in the state table (ST) where o_{i1}=0. We determine the value n = Operation $_{i}$ [o_{i1}], where n = Operation $_{i\,\mathrm{ST}}$ [o_{i1}], which we use to calculate D where D = Operation $_{i\,\mathrm{RT}}$ [n]. We calculate this position o_{i2} from the formula o_{i2} = (o_{i1} +D) mod N (where N – number of jobs). Next the value change in the CA state table is realized on the above mentioned positions o_{i1}, o_{i2}:

$$\text{Operation}_{i\,\mathrm{ST}}[o_{i1}] = \text{Operation}_{i\,\mathrm{ST}}[o_{i2}];$$

eg. for o_{i1} = o_{i2} for the values 0 1 2 3 4 5 6 in the previous state table we have values 5 1 2 3 4 0 6 in the new state table. After the change is done we assume o_{i1} = o_{i2}. All the last permutations obtained as a result of the CA execution for each of the operations of all the CA state tables create a schedule. The number of schedules in one iteration of the algorithm is equal to number of populations. At the next stage schedule sorting takes place on the basis of the value of the makespan as well as the their selection with a determined

hybridization coefficient. As a result of those operations new rule tables for the next iteration are obtained. The CA for machine choice in individual groups operates in a similar way.

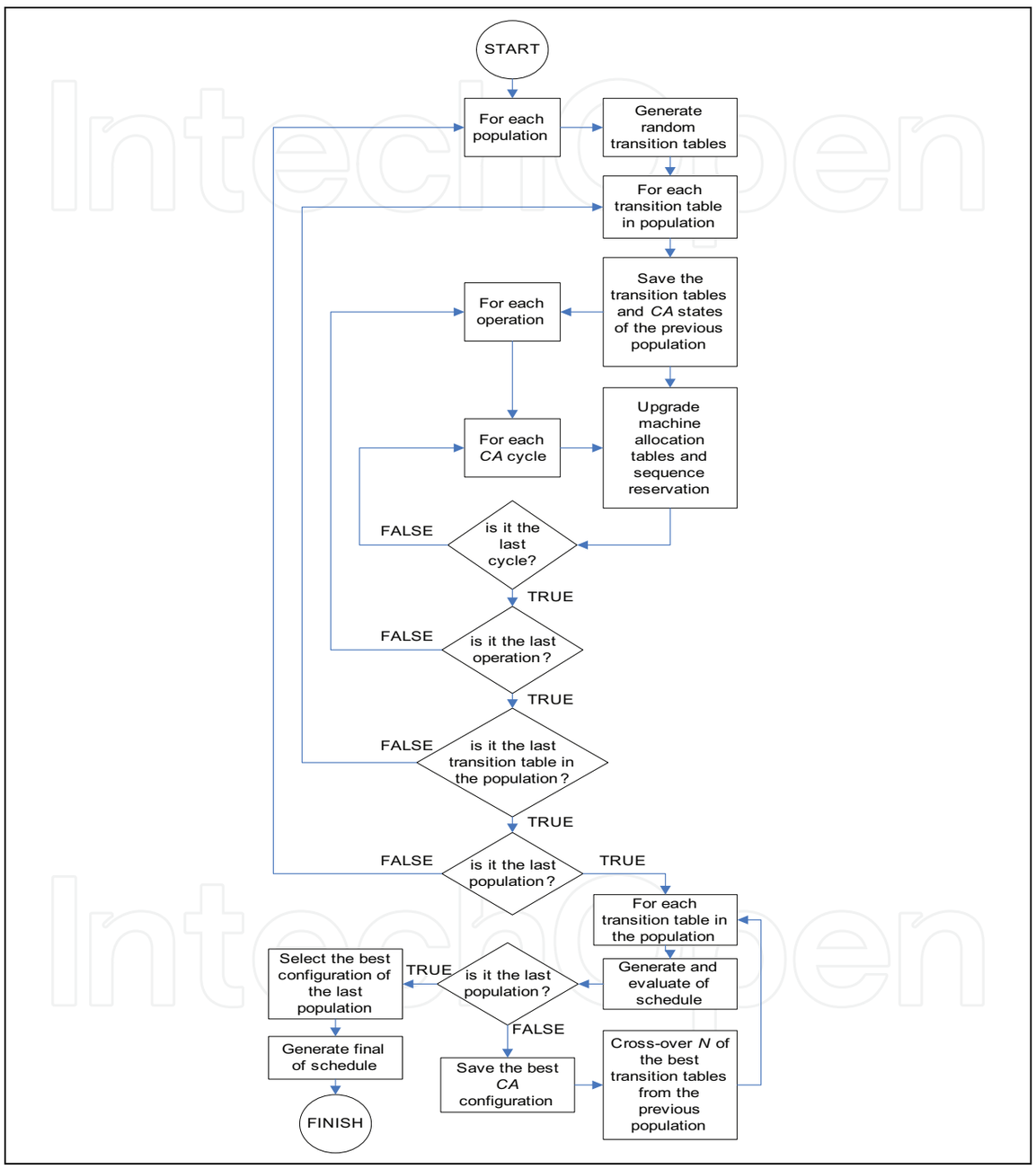


Fig. 1. Flowchart of evolving CA for flexible job shop scheduling.

4.2 Example

Examples of transition tables for the CA responsible for machine allocation from technological groups for individual operations are shown in Fig. 2.

Transition table: 0
Operation: 0 [2, 0, 5, 0, 4, 4, 2]
Operation: 1 [1, 1, 4, 5, 5, 2, 1]
Operation: 2 [5, 1, 5, 2, 6, 1, 5]
Operation: 3 [5, 2, 6, 0, 2, 2, 6]
Transition table: 1
Operation: 0 [2, 1, 6, 2, 0, 0, 3]
Operation: 1 [0, 2, 2, 5, 5, 4, 5]
Operation: 2 [1, 1, 0, 4, 0, 6, 3]
Operation: 3 [4, 3, 2, 3, 1, 3, 0]
Transition table: 2
Operation: 0 [4, 1, 2, 3, 0, 6, 1]
Operation: 1 [0, 0, 0, 2, 6, 0, 3]
Operation: 2 [6, 1, 2, 3, 5, 2, 0]
Operation: 3 [6, 5, 4, 3, 3, 6, 0]

Fig. 2. Automata transition tables (allocate machines)

Examples of transition tables for the CA responsible for operation sequence in a generated schedule are shown in Fig. 3.

Transition table: 0
Operation: 0 [5, 4, 1, 0, 1, 2, 6]
Operation: 1 [3, 2, 2, 4, 0, 6, 5]
Operation: 2 [6, 2, 1, 5, 0, 3, 1]
Operation: 3 [4, 6, 2, 5, 2, 0, 6]
Transition table: 1
Operation: 0 [6, 5, 3, 0, 3, 2, 2]
Operation: 1 [6, 4, 0, 1, 5, 0, 3]
Operation: 2 [4, 1, 5, 3, 3, 4, 5]
Operation: 3 [6, 2, 6, 3, 6, 6, 4]
Transition table: 2
Operation: 0 [5, 2, 4, 6, 3, 2, 3]
Operation: 1 [2, 2, 6, 3, 2, 0, 4]
Operation: 2 [0, 2, 4, 2, 0, 3, 1]
Operation: 3 [5, 5, 6, 6, 3, 3, 5]

Fig. 3. Automata transition tables (sequence operations)

The use of machines (CA changes for the 10 consecutive states in a cycle of each table - left column) and the operation sequence (CA changes for the next consecutive states in a cycle of each table -right column) for one population are shown in Fig.4.

State table: 0

Op: 0 [0, 0, -1, 0, -1, -1, -1]	**Op: 0 [0, 1, 2, 3, 4, 5, 6]**
Op: 0 [1, 0, -1, 0, -1, -1, -1]	Op: 0 [5, 1, 2, 3, 4, 0, 6]
Op: 0 [1, 0, -1, 0, -1, -1, -1]	Op: 0 [5, 1, 2, 0, 4, 3, 6]
.....	.....
Op: 0 [1, 0, -1, 1, -1, -1, -1]	Op: 0 [3, 6, 5, 0, 2, 1, 4]
Op: 0 [1, 0, -1, 2, -1, -1, -1]	Op: 0 [3, 0, 5, 6, 2, 1, 4]
Op: 1 [0, 0, 0, -1, 0, -1, -1]	**Op: 1 [0, 1, 2, 3, 4, 5, 6]**
Op: 1 [1, 0, 0, -1, 0, -1, -1]	O p: 1 [2, 1, 0, 3, 4, 5, 6]
Op: 1 [1, 0, 0, -1, 0, -1, -1]	Op: 1 [2, 1, 4, 3, 0, 5, 6]
.....	.....
Op: 1 [0, 0, 0, -1, 0, -1, -1]	Op: 1 [4, 3, 6, 5, 0, 2, 1]
Op: 1 [1, 0, 0, -1, 0, -1, -1]	Op: 1 [4, 3, 6, 5, 1, 2, 0]
Op: 2 [-1, -1, -1, 0, 0, 0, 0]	**Op: 2 [0, 1, 2, 3, 4, 5, 6]**
Op: 2 [-1, -1, -1, 0, 0, 0, 0]	Op: 2 [6, 1, 2, 3, 4, 5, 0]
Op: 2 [-1, -1, -1, 0, 0, 0, 0]	Op: 2 [6, 1, 2, 3, 4, 0, 5]
.....	.....
Op: 2 [-1, -1, -1, 2, 1, 2, 1]	Op: 2 [5, 6, 1, 2, 3, 0, 4]
Op: 2 [-1, -1, -1, 2, 1, 2, 2]	Op: 2 [5, 6, 1, 2, 0, 3, 4]
Op: 3 [-1, -1, -1, -1, 0, 0, 0]	**Op: 3 [0, 1, 2, 3, 4, 5, 6]**
Op: 3 [-1, -1, -1, -1, 0, 0, 0]	Op: 3 [4, 1, 2, 3, 0, 5, 6]
Op: 3 [-1, -1, -1, -1, 0, 0, 0]	Op: 3 [4, 0, 2, 3, 1, 5, 6]
.....	.....
Op: 3 [-1, -1, -1, -1, 1, 1, 1]	Op: 3 [1, 0, 6, 4, 5, 2, 3]
Op: 3 [-1, -1, -1, -1, 1, 1, 1]	Op: 3 [1, 2, 6, 4, 5, 0, 3]

State table: 1

Op: 0 [0, 0, -1, 0, -1, -1, -1]	**Op: 0 [0, 1, 2, 3, 4, 5, 6]**
Op: 0 [1, 0, -1, 0, -1, -1, -1]	Op: 0 [6, 1, 2, 3, 4, 5, 0]
Op: 0 [1, 0, -1, 0, -1, -1, -1]	Op: 0 [6, 1, 2, 3, 4, 0, 5]
.....	.....
Op: 0 [2, 0, -1, 2, -1, -1, -1]	Op: 0 [5, 6, 1, 2, 3, 0, 4]
Op: 0 [2, 0, -1, 2, -1, -1, -1]	Op: 0 [5, 6, 1, 2, 0, 3, 4]
Op: 1 [0, 0, 0, -1, 0, -1, -1]	**Op: 1 [0, 1, 2, 3, 4, 5, 6]**
Op: 1 [1, 0, 0, -1, 0, -1, -1]	Op: 1 [2, 1, 0, 3, 4, 5, 6]
Op: 1 [1, 0, 0, -1, 0, -1, -1]	Op: 1 [2, 1, 4, 3, 0, 5, 6]
.....	.....
Op: 1 [2, 0, 0, -1, 0, -1, -1]	Op: 1 [4, 3, 6, 5, 0, 2, 1]
Op: 1 [2, 0, 0, -1, 0, -1, -1]	Op: 1 [4, 3, 6, 5, 1, 2, 0]
Op: 2 [-1, -1, -1, 0, 0, 0, 0]	**Op: 2 [0, 1, 2, 3, 4, 5, 6]**
Op: 2 [-1, -1, -1, 0, 0, 0, 0]	Op: 2 [4, 1, 2, 3, 0, 5, 6]
Op: 2 [-1, -1, -1, 0, 0, 0, 0]	Op: 2 [4, 0, 2, 3, 1, 5, 6]

.....	.....
Op: 2 [-1, -1, -1, 2, 0, 0, 0]	Op: 2 [1, 0, 6, 4, 5, 2, 3]
Op: 2 [-1, -1, -1, 2, 0, 0, 0]	Op: 2 [1, 2, 6, 4, 5, 0, 3]
Op: 3 [-1, -1, -1, -1, 0, 0, 0]	**Op: 3 [0, 1, 2, 3, 4, 5, 6]**
Op: 3 [-1, -1, -1, -1, 0, 0, 0]	Op: 3 [6, 1, 2, 3, 4, 5, 0]
Op: 3 [-1, -1, -1, -1, 0, 0, 0]	Op: 3 [6, 1, 2, 3, 4, 0, 5]
.....	.....
Op: 3 [-1, -1, -1, -1, 1, 1, 0]	Op: 3 [5, 6, 1, 2, 3, 0, 4]
Op: 3 [-1, -1, -1, -1, 2, 1, 0]	Op: 3 [5, 6, 1, 2, 0, 3, 4]

State table: 2

Op: 0 [0, 0, -1, 0, -1, -1, -1]	**Op: 0 [0, 1, 2, 3, 4, 5, 6]**
Op: 0 [1, 0, -1, 0, -1, -1, -1]	Op: 0 [5, 1, 2, 3, 4, 0, 6]
Op: 0 [1, 0, -1, 0, -1, -1, -1]	Op: 0 [5, 1, 2, 0, 4, 3, 6]
.....	.....
Op: 0 [0, 0, -1, 0, -1, -1, -1]	Op: 0 [3, 6, 5, 0, 2, 1, 4]
Op: 0 [0, 0, -1, 0, -1, -1, -1]	Op: 0 [3, 0, 5, 6, 2, 1, 4]
Op: 1 [0, 0, 0, -1, 0, -1, -1]	**Op: 1 [0, 1, 2, 3, 4, 5, 6]**
Op: 1 [1, 0, 0, -1, 0, -1, -1]	Op: 1 [4, 1, 2, 3, 0, 5, 6]
Op: 1 [1, 0, 0, -1, 0, -1, -1]	Op: 1 [4, 0, 2, 3, 1, 5, 6]
.....	.....
Op: 1 [0, 0, 0, -1, 0, -1, -1]	Op: 1 [1, 0, 6, 4, 5, 2, 3]
Op: 1 [1, 0, 0, -1, 0, -1, -1]	Op: 1 [1, 2, 6, 4, 5, 0, 3]
Op: 2 [-1, -1, -1, 0, 0, 0, 0]	**Op: 2 [0, 1, 2, 3, 4, 5, 6]**
Op: 2 [-1, -1, -1, 0, 0, 0, 0]	Op: 2 [0, 1, 2, 3, 4, 5, 6]
Op: 2 [-1, -1, -1, 0, 0, 0, 0]	Op: 2 [0, 1, 2, 3, 4, 5, 6]
.....	.....
Op: 2 [-1, -1, -1, 1, 0, 0, 2]	Op: 2 [0, 1, 2, 3, 4, 5, 6]
Op: 2 [-1, -1, -1, 1, 0, 0, 0]	Op: 2 [0, 1, 2, 3, 4, 5, 6]
Op: 3 [-1, -1, -1, -1, 0, 0, 0]	**Op: 3 [0, 1, 2, 3, 4, 5, 6]**
Op: 3 [-1, -1, -1, -1, 0, 0, 0]	Op: 3 [5, 1, 2, 3, 4, 0, 6]
Op: 3 [-1, -1, -1, -1, 0, 0, 0]	Op: 3 [5, 1, 2, 0, 4, 3, 6]
.....	.....
Op: 3 [-1, -1, -1, -1, 1, 0, 0]	Op: 3 [3, 6, 5, 0, 2, 1, 4]
Op: 3 [-1, -1, -1, -1, 2, 0, 0]	Op: 3 [3, 0, 5, 6, 2, 1, 4]

Fig. 4. CA changes for the 10 consecutive states

The numbers in the left column of the tables stand for the number of the machine in a group, and their indexes (i.e. allocation in the table) are the numbers of the parts. The numbers in the right column of the tables stand for the sequence of individual parts in a given operation and their indexes (i.e. allocation in the table) are the numbers of the parts. Value (-1) in the left column of the tables stands for lack of machine participation in a given operation with a given part. Value (-1) in the right column would stand for lack of processing of a part in a given operation. All the (-1) values are ignored in the state change

procedure of the CA, and does not participate in the machine allocation procedure. For each iteration makespan is determined for the generated schedule, on the basis of the final states of both automata.

All the makespan**es** for each schedule from a population are recorded and compared in order to select the best schedule from the current population. If it is not the final population then the best rule tables are crossed over in order to generate the best schedules from the current and previous population. In each iteration summary time realize of all operations (makespan) for generate schedule on basis final state two cellular automata is determinated. All makespan**es** for each schedule with population are writing and compare to aim choice best schedule among current populations. If population no is latest we realize crossower operation best rule table which lead for generated best schedules with current and previous populations. Half given number is taken with current population, and second half with previous population. Depending to generated value and given intensity follows determine values with current table to previous table or vice versa (is possible lack operation).

5. Computational results

5.1 Comparative study of cellular automata for FJSP

Two types of routing were considered: a serial and a parallel one. In a serial route an entire batch of parts is processed on one machine and only when all of the products in the batch have been processed are they sent to the next machine. In a parallel route individual items of the batch are sent to the next machines as soon as they have been processed on the previous machine.

The research was carried out on a computer with an Intel Core2 2.4 GHz processor and 2047 MB of RAM for the following settings of the CA algorithm: size of population = 1000; number of iterations = 100; number of transitions = 1000; hybridization ratio = 0.9; and intensity of hybridization = 0.9.

For solution of FJSP problem special software to realize the CA algorithm have been created. Computer experiments were carried out for data presented in (Witkowski, 2005) – where the number of operations is 160, and the number of machines 26.

The experiments have been carried out for the hybridization ratio: 0,1; 0,5; 0,9 and the intensity of hybridization equal to 0,1; 0,5; 0,9. The simulation of each test problem was run with the SP population size equal to 10, 100, 1000, the RT transition rate was equal to 10, 100, 1000, and the IN iteration number was equal to 10, 100, 1000. Besides, in some cases the values of SP, RT and IT reached 10000.

The following symbols for signed algorithm parameters: SP - size of population; IT - number of iterations; RT - number of transitions; HR - hybridization ratio; and IH - intensity of hybridization have been used. Individual SP, IT and RT parameters assume one of the values from the set {10, 100, 1000}; moreover HR and IH from the sets{(0,1; 0,1); (0,5; 0,5) and (0,9; 0,9)} respectively.

For IT, SP and RT values we assume the following linguistic variables : N – low value, S – medium value, D – high value (V – very high value – in some combinations of parameters). In this way 27 combinations with parameters of the algorithm were created (fig. 5) For example. one of such combinations is IT(M)-SP(S)-RT(D), etc..

Table 1 shows some of the results (with the SP (D) value) for the test parameters of the CA algorithm.

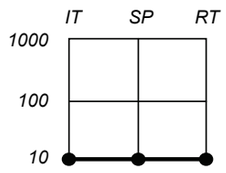
hybridization ratio = 0.9; intensity of hybridization = 0.9. size of the population = 1000; number of transitions = 10.									
iter.	*Makespan*					*min.*	*average*	*max.*	*avg. time [sec.]*
10	66295.6	71146.1	71852.0	69426.5	69729.6	66295.6	69282.3	71852.0	10
	69158.2	68452.7	69949.2	66517.3	70295.7				
100	67436.6	63232.1	65860.1	69623.8	63542.7	63232.1	66403.6	69623.8	102
	66320.0	68723.9	65948.9	66832.0	66516.2				
1000	66305.5	66918.2	64587.4	65644.4	63057.3	63057.3	66176.2	68121.8	1020
	67373.0	68121.8	66553.8	67074.6	66126.4				

hybridization ratio = 0.9; intensity of hybridization = 0.9. size of the population = 1000; number of transitions = 100.									
iter.	*Makespan*					*min.*	*average*	*max.*	*avg. time [sec.]*
10	72254,0	68692,3	72917,7	69880,7	69518,9	66240,2	69449,3	72917,7	10
	71397,2	66240,2	67804,5	68055,5	67732,4				
100	70885,2	68269,0	63363,8	65451,6	70035,0	63363,8	67352,8	70885,2	106
	67585,8	67766,4	66859,0	68564,2	64747,8				
1000	66239,3	68417,1	69373,5	68334,9	62826,0	62826,0	67515,4	69373,5	1068
	69118,0	69067,4	68188,7	67135,2	66453,5				

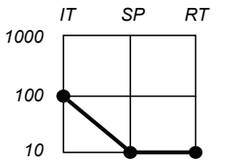
hybridization ratio = 0.9; intensity of hybridization = 0.9. size of the population = 1000; number of transitions = 1000.									
iter.	*Makespan*					*min.*	*average*	*max.*	*avg. time [sec.]*
10	70217.3	66442.4	68170.6	69813.4	65720.8	63161.2	68811.1	72895.0	15
	63161.2	71086.2	72199.9	68404.2	72895.0				
100	67537.3	62753.4	71358.4	67292.3	67810.3	62753.4	67242.0	71358.4	156
	69405.4	66784.8	65753.1	68183.8	65540.8				
1000	64347.2	63539.3	66144.4	66560.0	69469.2	63539.3	67083.4	70579.7	1564
	67208.2	70579.7	66920.0	67386.3	68679.3				

Table 1. Some of the results (with the SP (D) value) for the test parameters of the CA algorithm (serial route)

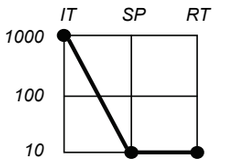
Figure 5 summarizes the results for the test problems that were run with the evolving cellular automata algorithm for the serial route.



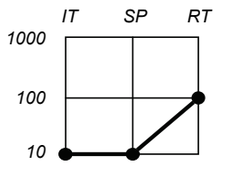
0.1; 0.1	81662,7
0.5; 0.5	81044,9
0.9; 0.9	78554,2



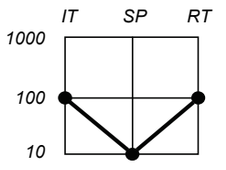
0.1; 0.1	77798,7
0.5; 0.5	76117,5
0.9; 0.9	78357,8



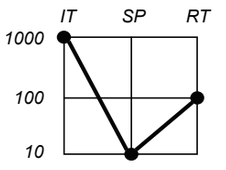
0.1; 0.1	76975,0
0.5; 0.5	79498,8
0.9; 0.9	77680,8



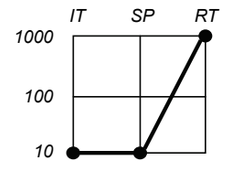
0.1; 0.1	83589,0
0.5; 0.5	83247,5
0.9; 0.9	81261,1



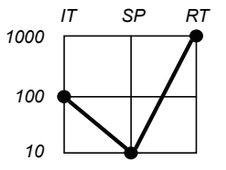
0.1; 0.1	79370,1
0.5; 0.5	81223,4
0.9; 0.9	80910,9



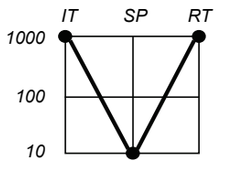
0.1; 0.1	80219,5
0.5; 0.5	83431,1
0.9; 0.9	77384,0



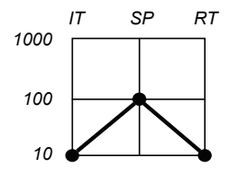
0.1; 0.1	82118,8
0.5; 0.5	79335,4
0.9; 0.9	83613,9



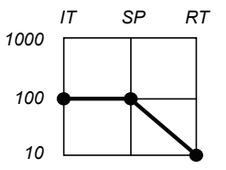
0.1; 0.1	81935,2
0.5; 0.5	78669,3
0.9; 0.9	78325,0



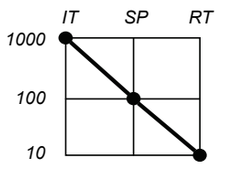
0.1; 0.1	78687,1
0.5; 0.5	82720,1
0.9; 0.9	80292,7



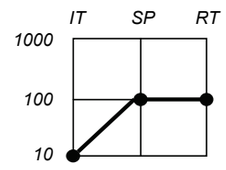
0.1; 0.1	76069,6
0.5; 0.5	75395,4
0.9; 0.9	74073,0



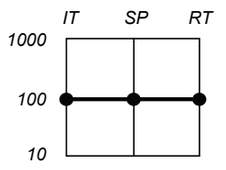
0.1; 0.1	69121,4
0.5; 0.5	70828,1
0.9; 0.9	70462,8



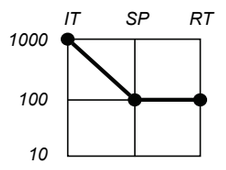
0.1; 0.1	71081,5
0.5; 0.5	70800,4
0.9; 0.9	70602,6



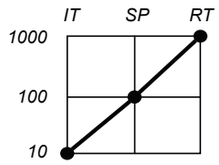
0.1; 0.1	74129,5
0.5; 0.5	76410,9
0.9; 0.9	74623,5



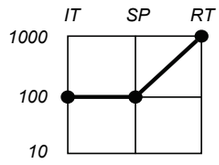
0.1; 0.1	70526,7
0.5; 0.5	73550,7
0.9; 0.9	73124,9



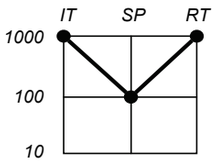
0.1; 0.1	72133,1
0.5; 0.5	72126,3
0.9; 0.9	73429,8



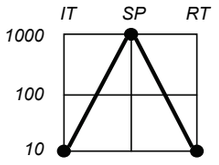
0.1; 0.1	75249,7
0.5; 0.5	73438,5
0.9; 0.9	75201,3



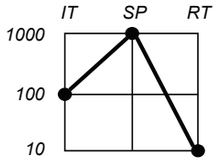
0.1; 0.1	72750,8
0.5; 0.5	72288,4
0.9; 0.9	71287,6



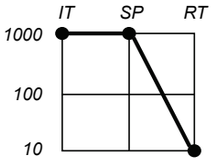
0.1; 0.1	73102,2
0.5; 0.5	73647,3
0.9; 0.9	72543,6



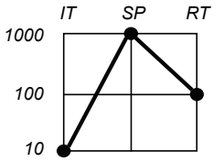
0.1; 0.1	69027,6
0.5; 0.5	69151,0
0.9; 0.9	69282,3



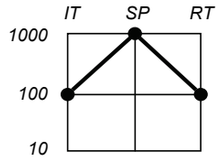
0.1; 0.1	67350,0
0.5; 0.5	66859,7
0.9; 0.9	66403,6



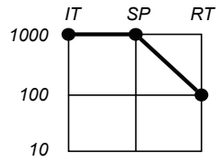
0.1; 0.1	66282,5
0.5; 0.5	66526,8
0.9; 0.9	66176,2



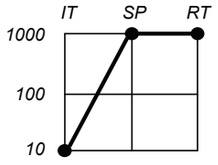
0.1; 0.1	69748,7
0.5; 0.5	69836,3
0.9; 0.9	69449,3



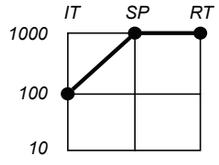
0.1; 0.1	68200,7
0.5; 0.5	67070,3
0.9; 0.9	67352,8



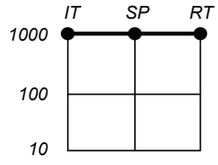
0.1; 0.1	66575,2
0.5; 0.5	66756,9
0.9; 0.9	67515,4



0.1; 0.1	70648,9
0.5; 0.5	69980,9
0.9; 0.9	68811,1



0.1; 0.1	67888,8
0.5; 0.5	68057,5
0.9; 0.9	67242,0



0.1; 0.1	67882,8
0.5; 0.5	67532,3
0.9; 0.9	67083,4

Fig. 5. The results for the test problems that were run with the evolving cellular automata algorithm (serial route)

We can generally see that depending on the PS population size we can single out 3 classes of quality results (with regard to the C_{max} criterion) - very good (large population size), average (medium population size) and poor (small population size). Moreover an increase of the IT value influences the C_{max} more than the RT value, although there are a number of exceptions.
The best results of C_{max} are always obtained at SP(D) regardless of the IT or RT values. Eg. at SP(D) value the best C_{max} is achieved for combination IT(D)-SP(D)-RT(M) rather than for IT(M)-SP(D)-RT(D); at SP(D) value the best C_{max} is achieved for combination IT(S)-SP(D)-RT(M) rather than for IT(M)-SP(D)-RT(S); at SP(D) value the best C_{max} is achieved for combination IT(D)-SP(D)-RT(S) rather than for IT(S)-SP(D)-RT(D). It should be noted that

for combination IT(D)-SP(D)-RT(D) an insignificantly poor C_{max} is achieved than for eg. IT(D)-SP(D)-RT(M).

The worst results of C_{max} are always obtained at SP(M) regardless of the IT or RT values. Eg. at SP(M) value the best C_{max} is achieved for combination IT(D)-SP(M)-RT(M) rather than for IT(M)-SP(M)-RT(D); moreover for combination IT(S)-SP(M)-RT(D) the C_{max} values are better than for IT(D)-SP(M)-RT(S) – while the pair (HR,IH) = (0,5:0,5), and the C_{max} is worse while the pair (HR,IH) = (0,1;0,1). For combination IT(S)-SP(M)-RT(M) the C_{max} values are clearly better than for IT(M)-SP(M)-RT(S). We can also see that for IT(D)-SP(D)-RT(D) an insignificantly poor C_{max} is achieved than eg. for IT(D)-SP(D)-RT(M).

Analizing the influence of SP on the C_{max} we can observe the following behaviour of the CA algorithm. An increase of the SP value from 10 to 100 decreases the average value of C_{max} from ca. 82000 to 74000 min., i.e. by about 8000 min. An increase of the SP value from 100 to 1000 decreases the average C_{max} value from 74000 to 69000 min. - by about 5000 min. Thus an increase of the SP value from 10 to 1000 decreases the average C_{max} value from 82000 to 69000 min. i.e. by about 13000 min. We can see that the increase from 100 to 1000 results in a slower decrease of C_{max} (i.e. by about 5000 min.) than the change of the SP value from 10 to 100 (i.e. about 8000 min.).

Let us consider the influence of IT on the C_{max} value. For combinations with SP(M) and RT(M) an increase of IT from 10 to 100 results in a decrease of C_{max} from 80000 to 77000 min., i.e. by ca. 3000 min. An IT increase from 100 to 1000 results in an insignificant decrease of C_{max} - by about 500 min. - and while the pair (HR,IH) = (0.5;0,5) in an increase of C_{max}. For combination with SP(S) and RT(M) the change of IT from z 10 to 100 gives an decrease of C_{max} from 75000 to 70000 min., i.e by ca. 5000 min.; moreover an increse of the IT value from 100 to 1 000 gives an insignificant decrease of C_{max} while (HR,IH) = (0.5;0,5) and a decrease of C_{max} while (HR,IH) = (0,1;01) and (HR,IH) = (0,9;0,9). At SP(D) and RT(M) values the increase of IT from 10 to 100 gives a decrease of C_{max} from 69000 to 67000 min., ie. by ca. 2000 min. An increase of IT from 100 to 1000 decreases the C_{max} from 67000 to 66500 min. for combinations IT(M)-SP(D)-RT(M), IT(S)-SP(D)-RT(M) and IT(D)-SP(D)-RT(M). A similar situation occurs for combinations IT(M)-SP(D)-RT(D), IT(S)-SP(D)-RT(D) and IT(D)-SP(D)-RT(D).

An increase of the IT value in most cases improves the C_{max} value eg. for combinations at SP(D), but there are also exceptions. For combination IT(S)-SP(D)-RT(S) we have better C_{max} than for IT(M)-SP(D)-RT(S). Moreover the C_{max} value increases in the following order: from IT(D)-SP(D)-RT(M) to IT(S)-SP(D)-RT(M) to IT(M)-SP(D)-RT(M). An increase of the IT value does not always result in a better C_{max}. For example dla C_{max} with average values i.e. with combinations which have the medium parameter SP(S) combination IT(S)-SP(S)-RT(S) gives a better C_{max} than IT(D)-SP(S)-RT(S) and combination IT(S)-SP(S)-RT(D) gives a better C_{max} than IT(D)-SP(S)-RT(D). Moreover combination IT(S)-SP(S)-RT(M) gives a better C_{max} than IT(D)-SP(S)-RT(M) while the pair (HR,IH) = (0,1;0,1) and the pair (HR,IH) = (0,9;0,9).

The increase of the RT value both increases and decreases the C_{max} value. For example combination IT(M)-SP(S)-RT(D) gives a better C_{max} than IT(M)-SP(S)-RT(S) while the pair (HR,IH) = (0,5;0,5) and IT(S)-SP(S)-RT(D) gives a better C_{max} than IT(S)-SP(S)-RT(M) while the pair (HR,IH) = (0,5;0,5) and (HR, IH) = (0,9;0,9). We can also note the following cases: combination IT(D)-SP(S)-RT(D) gives better values of C_{max} than IT(D)-SP(S)-RT(S); IT(D)-SP(M)-RT(S) gives better C_{max} than IT(D)-SP(M)-RT(M) while the pair (HR,IH) = (0,9;0,9); IT(D)-SP(M)-RT(D) gives a better C_{max} than IT(D)-SP(M)-RT(S) while the pair (HR,IH) = (0,5;0,5) and (HR,IH) = (0,9;0,9); IT(S)-SP(M)-RT(D) gives a better C_{max} than IT(S)-SP(M)-

RT(S) while the pair (HR,IH) = (0,5;0,5) and (HR,IH) = (0,9;0,9); IT(M)-SP(M)-RT(D) gives a better C_{max} than IT(M)-SP(M)-RT(M) while (HR, IH) = (0,1;0,1) and (HR, IH) = (0,9;0,9) and a better C_{max} than IT(M)-SP(M)-RT(M) while (HR,IH) = (0,5;0,5).
For IT(D)-SP(D)-RT(S) the CA algorithm gives a better C_{max} than for IT(D)-SP(D)-RT(D). Similarly combination IT(S)-SP(D)-RT(M) gives a better C_{max} than IT(S)-SP(D)-RT(S) and IT(M)-SP(D)-RT(M) gives a better C_{max} than IT(M)-SP(D)-RT(S).
In the group of poorest C_{max} values (with SP(M) value) we can observe that the best C_{max} are achieved while the pair (HR,IH) = (0,1;0,1) - 3 times, while the pair (HR,IH) = (0,5;0,5) - twice and while the pair (HR,IH) = (0,9;0,9) - 4 times; moreover the worst C_{max} is achieved while the pair (HR,IH) = (0,1;0,1) - 3 times, while the pair (HR,IH) = (0,5;0,5) - 4 times and while the pair (HR, IH) = (0,9;0,9) - twice.
In the group of average C_{max} values (with P(S) value) we can obeserve that the best C_{max} is achieved while the pair (HR,IH) = (0,1;0,1) - 3 times, while the pair (HR,IH) = (0,5;0,5) – twice and while (0,9;0,9) – 4 times; moreover the worst C_{max} is achieved while the pair (0,1;0,1) - 4 times, the pair is (0,5;0,5) - 4 times and the pair is (0,9;0,9) - once.
In the group of the best C_{max} values (with SP(D)) we can see that the CA algorithm gives the best C_{max} values while the pair is (0,1;0,1) - twice, (0,5;0,5) - once and at pair (0,9;0,9) – 3 times; moreover the worst C_{max} (with SP(M)) is achieved while the pair is (0,1;0,1) - 4 times, while (0,5; 0,5) - 3 times and while (0,9;0,9) - twice.
Below we present some results achieved when applying higher values of SP, IT and RT than in the main experiment - (ie. SP(V), IT(V), RT(V) values equal 10000).
At SP(V) when the SP increase is from 1000 to 10000 the average value of C_{max} decreases significantly from 69151 to 66060 min. for combination IT(M)-SP(V)-RT(M) compared to IT(M)-SP(D)-RT(M) and from 66859 to 63739 min. for IT(D)-SP(V)-RT(M) compared to IT(D)-SP(D)-RT(M) while the pair (HR,IH) = (0,5;0,5). While the pair (HR,IH) = (0,5;0,5) for combination IT(M)-SP(V)-RT(S) a decrease of C_{max} is achieved from 69836 to 67456 compared to IT(M)-SP(D)-RT(S). For combination IT(S)-SP(V)-RT(S) a decrease of C_{max} is achieved from 67070 to 64626 min. compared to IT(S)-SP(D)-RT(S). Similarly for combination IT(M)-SP(V)-RT(D) a decrease of C_{max} is achieved from 69980 to 67351 compared IT(M)-SP(D)-RT(D), and for combination IT(S)-SP(V)-RT(D) a decrease of C_{max} was achieved from 68057 to 63840 min. compared to IT(S)-SP(D)-RT(S). For combination IT(M)-SP(V)-RT(D) a decrease of C_{max} was achieved from 69980 to 67351 min. compared to IT(M)-SP(D)-RT(D), and for combination IT(S)-SP(V)-RT(D) a decrease of C_{max} - from 68057 to 63840 min. compared to IT(S)-SP(D)-RT(S).
When analizing the influnece of IT(V) on C_{max} we can note that almost in all the analized cases the an icrease from IT(D) to IT(V) gives an insignificant decrease of the C_{max} value eg. for combination IT(V)-SP(D)-RT(M) compared to IT(D)-SP(D)-RT(M), this change is equal to 70800-70407= 393 min.; for combination IT(V)-SP(M)-RT(D) compared to IT(D)-SP(M)-RT(D) the change is equal to 83431-82657 = 774 min.; for combination IT(V)-SP(D)-RT(D) compared to IT(D)-SP(D)-RT(D) the change is equal to 72126-71315 = 811 min.; for combination IT(V)-SP(M)-RT(D) compared to IT(D)-SP(M)-RT(D) the change is equal to 82720-77636= 5084 min.; and for combination IT(V)-SP(S)-RT(D) compared to IT(D)-SP(S)-RT(D) the change is equal to 73647-72 115 = 1432 min.
When analizing the influnece of RT (V) eg. in combinations IT(M)-SP(M)-RT(V) at RT(D) we can observe both increases and decreases of the C_{max} value.
For combinations with the SP(D) value (fig. 5) the CA algorithm has a better C_{max} in all cases as compared to the combinations with the SP(M) value and has a better C_{max} in almost all cases as compared to the combinations with the SP(S) value - as it can be seen in fig. 5.

For combinations with the SP(S) value (fig. 5) the CA algorithm has a better C_{max} in almost all cases as compared to the combinations with the SP(M) value and has a worse C_{max} in all cases as compared to the combinations with SP(D).
For combinations with the SP(M) value (fig. 5) the CA algorithm has a worse C_{max} in almost all cases as compared to the combinations with the SP(S) value and has a worse C_{max} in all cases as compared to the combinations with SP(D).
Overall, the CA algorithm for combinations with the SP(D) value produces solutions of better optimality compared to the CA algorithm for combinations with the SP(S) value and significantly better than with SP(M).
For the problem being solved Gantt charts with the one of best makespan value have been constructed: with machines (Fig. 6), and with parts (Fig.7) while the route is serial.

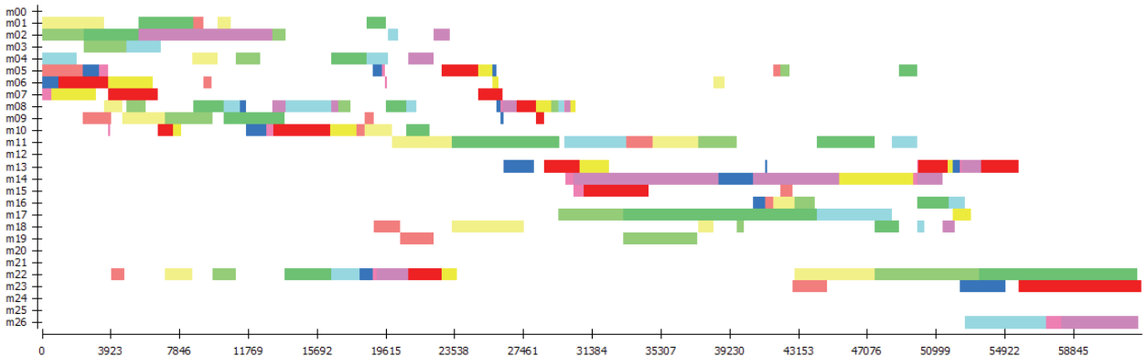


Fig. 6. The Gantt chart for the problem solved for machines (serial route)

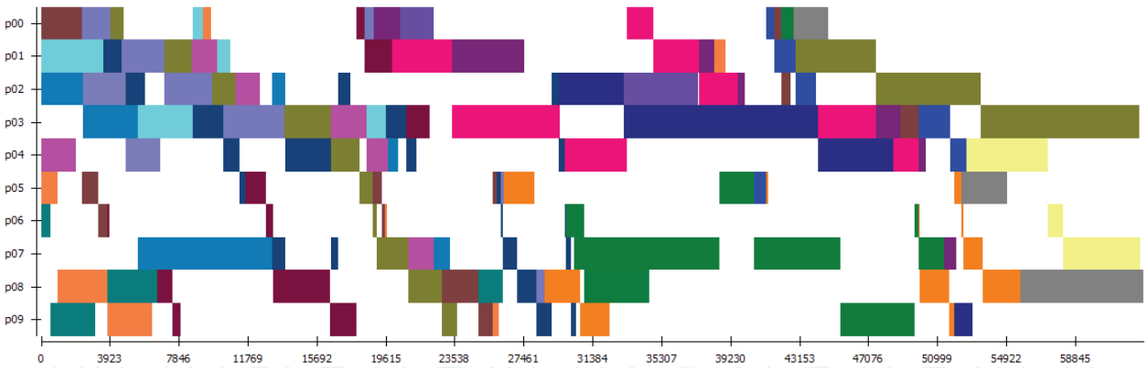


Fig. 7. The Gantt chart for the problem solved for parts (serial routes)

5.2 Comparison of the CA with a genetic algorithm for FJSP

The results obtained with the evolving cellular automata algorithm and genetic algorithm have been compared. A genetic algorithm is characterized by a parallel search of the state space by keeping a set of possible solutions under consideration, called a population. A new generation is obtained from the current population by applying genetic operators such as mutation and crossover to produce new offspring. The application of a GA requires an encoding scheme for a solution, the choice of genetic operators, a selection mechanism and the determination of genetic parameters such as the population size and probabilities of applying the genetic operators.

In our test, we use the genetic algorithm tested in Witkowski et. al (2004, 2007), where there is a more detailed description of the algorithm. Here, we use the recommended parameters, in particular we use a mutation probability of 0.8 and a crossover probability of 0.2.
Figure 8 shows some of the best results for of the CA algorithm, and Table 2 shows some ofthe results for the GA algorithm (serial route).

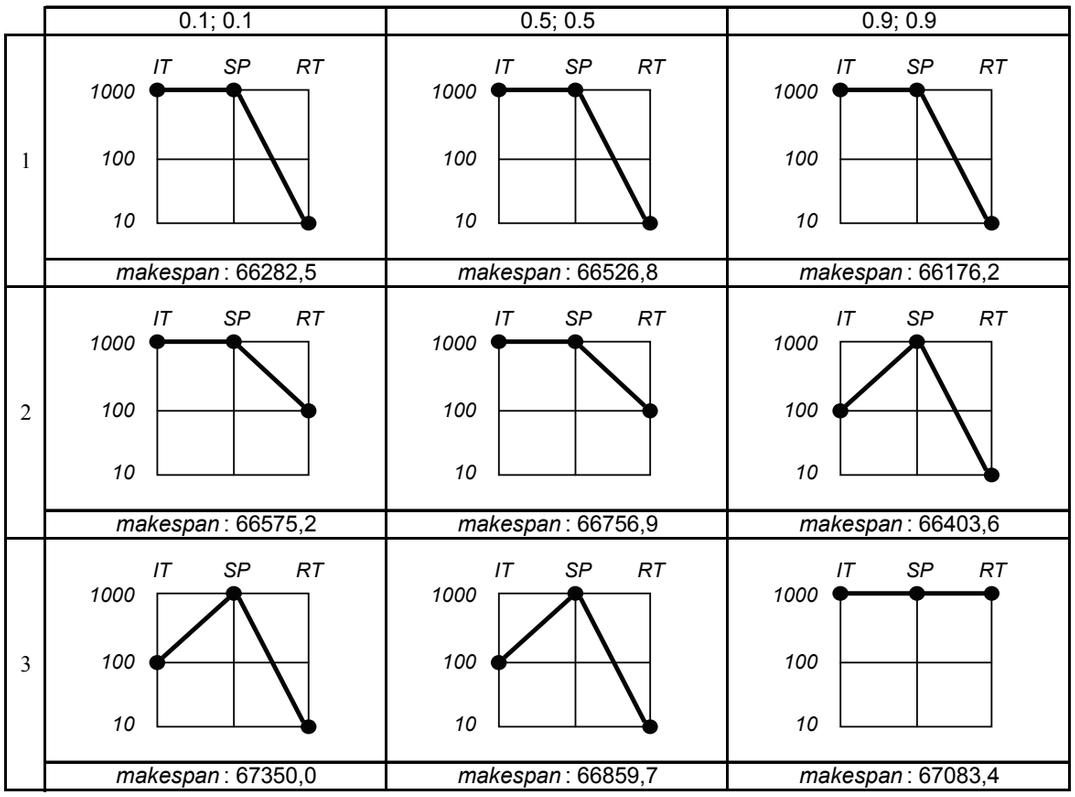


Fig. 8. Some of the best results for the test parameters of CA algorithm (serial route)

	Experiment number	Number of generations	Min imum makespan [min]	Number of schedules	Average makespan [min]
	1	42	59830	595	64380
	2	28	69211	142	74443
	3	44	62664	210	69384
	4	40	67199	120	69230
	5	19	64615	421	69657
	6	6	58734	768	64459
	7	46	63438	330	67457
	8	46	57636	630	61238
	9	33	60236	646	70858
Average		34	62618		67901

Table 2. Some of the results for the GA algorithm (serial route)

In the experiments with the CA algorithm (parallel route) simulations of each test problem were run with the SP population size equal to 10, 100, 1000, the RT transition rate equal to 10, 100, 1000, and the IN iteration number equal to 10, 100, 1000. Each experiment was repeated 10 times.

hybridization ratio = 0.9; intensity of hybridization = 0.9. size of the population = 1000; number of transitions = 10.									
iter.	*Makespan*					*min.*	*average*	*max.*	*avg. time [sec.]*
10	37543.2	38625.2	36427.7	38969.5	37880.1	36427.7	38923.8	41153.0	42
	38976.1	41153.0	39895.2	39895.2	39872.7				
100	37661.2	37281.3	36120.0	37456.5	36597.7	36120.0	37219.7	38143.9	420
	36350.0	37029.1	37433.0	38124.6	38143.9				
1000	36300.6						36330.6		4201

Table 3. Some of the results (with SP (D) value) for the test parameters of the CA algorithm (parallel route)

For the problem being solved Gantt charts with the one of best makespan value have been showed with machines (Fig. 6) while the route is parallel.
In the experiments with the GA algorithm (parallel route) we have used the following: mutation type – single-swap; crossover type - order-based; selection type – roulette. The experiment series was carried out with the following parameters: population size– 1000; generation number - 50. Each experiment was repeated 9 times.
Tabele 4 shows the results for the GA algorithm.

PM 1/ 1000	PC 1/ 1000	Value of C_{max} for different parameters of the GA algorithm [min]						
		1	**2**	**3**	**4**	**5**	**6**	**7**
512	128	37545	36724	38155	37585	35637	39430	38822
1000	450	38926	38543	37084	37065	38862	38588	40597
192	256	37368	40725	41188	38902	40709	39457	38531
96	353	37729	38707	40275	37866	39706	39514	39915
256	450	40018	35124	39795	38777	37631	38515	38777
64	256	40359	38339	36397	37939	38109	38610	39853
128	64	39775	38181	40018	37210	40112	39615	38968
16	256	41088	38055	40519	40566	39857	37301	39629
8	128	40200	41239	39281	40422	39025	40047	41556
8	256	40942	38210	38390	40132	39879	39890	35470
8	450	36868	39628	39560	39932	39959	38842	40078
32	450	37158	40589	40653	37440	37855	38031	39183
32	256	39961	38647	41262	40165	39269	35166	39483
32	128	40567	40193	39226	39579	40423	38843	39825
64	128	42941	38274	39981	39491	40111	39908	37428

64	256	38145	38874	36901	39209	38915	40416	39649
64	450	39013	39672	39344	40236	39826	39775	41053
128	353	38671	38903	39186	39422	37540	36506	38223
512	353	40968	39619	40180	39583	38870	40559	36429
192	256	37466	39257	38295	38501	38132	40304	39477
96	256	39444	38595	38905	38402	38774	38134	39817
64	353	37770	38677	39667	40263	39317	37676	39445
16	353	39930	39980	41221	38075	38681	39236	40329
280	130	40586	39648	36805	38176	38239	36077	39873
280	370	39253	38892	38222	38726	38130	40813	39168
840	130	41824	37050	33949	37923	37435	38013	38712
910	370	37426	37692	38223	38467	37562	40862	39039
8	64	38436	41069	37987	39287	38690	39993	39042
16	64	37505	38351	36647	39846	34738	40428	39065
32	64	38554	39284	38144	38171	41728	39474	38144
64	64	40415	39862	36583	38454	36264	39039	40426
96	64	40342	39791	39055	39394	39872	38728	37385
192	64	40652	36435	38656	40164	38549	37849	37345
256	64	40814	38851	40186	40448	39435	41031	37419
280	64	38278	38972	38653	38102	37330	38661	39112
512	64	38788	37774	39794	38820	40603	39224	39700
840	64	40521	38146	37848	38323	38501	38126	37391
910	64	39983	39467	38609	35735	38880	37447	41173
1000	64	38465	40961	36692	38726	39597	38493	37194
16	128	38645	38577	39785	39314	39181	39658	37034
96	128	40260	39932	39688	39593	39786	38340	38967
128	128	39873	39150	37624	39528	40273	38549	40675
192	128	38559	39003	36695	40691	38635	39235	39041
840	128	38322	39987	37723	40669	39489	38108	39049
910	128	37698	39438	37920	35857	40082	39579	34667
1000	128	38599	39670	39822	37052	38917	38312	40619
8	130	38254	39256	37131	37520	40000	39848	39644
16	130	41453	38340	37480	40308	37198	40346	39298
32	130	38986	39323	39196	39646	40179	38961	38651
64	130	38597	38765	39151	40179	39599	38224	40713
96	130	41377	38974	40071	39615	38247	40970	39434
128	130	40669	40836	39114	40808	39476	37848	38918
192	130	38561	39158	39877	39820	39020	38387	40398
256	130	36108	37828	38378	40828	38998	40361	38449
512	130	38935	39359	37933	33549	37184	40117	39846

910	130	38595	40498	40037	39457	38055	37419	39086
1000	130	38604	37872	39596	39220	39818	38110	39327
128	256	35614	38756	37070	40614	40770	38541	39979
256	256	39891	39413	35647	38149	38485	41064	40019
280	256	38330	36906	37492	39298	39913	39255	39768
512	256	39951	39321	39691	37334	36105	36620	40355
840	256	38195	39439	39945	40077	40545	38156	36714
910	256	39265	40465	37356	40592	39022	31701	38760
1000	256	41797	40467	40199	37534	37890	39089	38813
8	353	38198	38215	39587	40206	37049	37457	37841
32	353	38420	38526	38422	36501	39419	37903	39557
192	353	39418	39418	35429	39043	39223	40586	39113
256	353	39693	38591	39548	37636	39295	39853	38624
280	353	37849	35995	39018	38860	37544	38312	37608
840	353	37159	37159	39290	39942	38015	37159	38966
910	353	41668	34752	39535	37406	39493	37733	38185
1000	353	39868	40506	38434	39869	38237	39632	38582
8	370	40255	38051	39140	38494	40274	38577	40698
16	370	39564	40285	39366	39737	39077	36294	40451
32	370	39253	40837	39576	37373	41460	38928	36018
64	370	38105	38778	39236	39644	36098	38552	40660
96	370	36646	39237	38242	36963	40197	38688	39378
128	370	39074	39184	38297	39650	40936	37823	38687
192	370	39305	36835	40083	41108	39513	38629	38312
256	370	38860	41255	42417	39995	34023	40120	37957
512	370	39695	40118	38358	39824	40379	40349	39603
840	370	40084	38184	37808	38206	37710	40083	38468
1000	370	38875	38661	40703	36498	38493	38535	37560
16	450	38756	39286	38315	39106	38933	40246	35516
96	450	35559	38088	39556	38268	40046	37104	40137
128	450	38532	40352	38448	42062	39222	38763	39441
192	450	39985	41215	40856	36504	40110	35746	39633
280	450	37565	38601	39134	39571	41389	38153	37565
512	450	38871	38547	38282	39381	39058	37768	37832
840	450	39854	38626	39132	33809	38059	40107	32947
910	450	36160	39825	38749	38673	39747	37460	39459

Tabele 4. Results (value C_{max}) for different parameters of the GA algorithm

The experiments showed that for CA algorithm we can achieve results similar to the GA algorithm – both for the serial route and parallel routes.

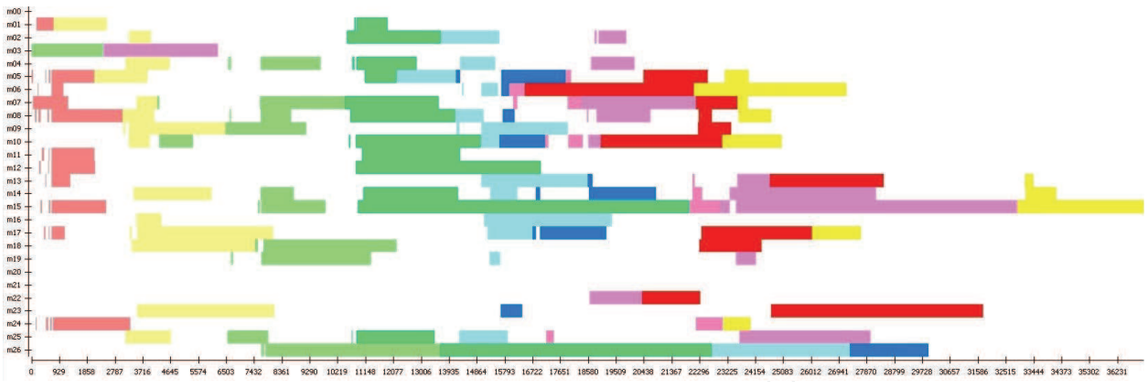


Fig. 9. Gantt chart for the solving problem with parallel route (for machines)

6. Conclusion

The paper presents an algorithm based on evolving cellular automata for solving flexible job shop scheduling problem. The presentation of the algorithm CA and its comparison with the GA algorithm shows positive results. The software of this algorithm allows for analysis of the schedule construction process for many variants reflecting a variety of combinations of other factors. We can generally see that depending on the PS population size we can single out 3 classes of quality results with regard to the C_{max} criterion - very good (large population size), average (medium population size) and poor (small population size). Moreover an increase of the IT value influences the C_{max} more than the RT value, although there are a number of exceptions..

In addition, we observed that for our specialized FJSP problem the trajectory methods (e.g. tabu search, simulated annealing, GRASP) have better efficiency than the CA algorithm, particularly when those algorithms are used in hybrid approaches [Witkowski et al., 2005a, 2005b, 2006]. Experiments for the analyzed FJSP problem indicate that the evolving cellular automata algorithm is comparable with such population-based methods as the genetic algorithm. Morover, the successful use of this approach will also depend on the amount of calculation that can be done and on further improvement of this algorithm for our problem.

7. References

Fattahi,P., Mehrabad, M.S. & Jolai, F. (2007). Matemathical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal Intel. Manufacturing*, Vol. 18, pp. 331-342

Kacem, I., Hammandi, S. & Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job shop scheduling problems, *IEEE T. System Man Cybernetics C.*, Vol. 32, pp.1-13

Liu, H., Abraham, A., Choi, O. & Moon, S.H. (2006). Variable Neighborhood Particle Swarm Optimization for Multi-objective Flexible Job-Shop Scheduling Problems, In T.-D. Wang et al. (eds), SEAL 2006, *LNCS* 4247, pp. 1997-2004.

Ong, Z.X., Tay, J.C. & Kwoh, C.K. (2005). Applying the Clonal Selection Principle to Find Flexible Job Shop Schedules ", ICARIS 2005, LNCS 3627, pp. 442-455

Preissl, R. (2006). A Parallel Approach For Solving The Flexible Job Shop Problem With Priority Rules Developed By Genetic Programming, Master's thesis, J. Kepler University, Linz

Shi-Jin, W., Bing-Hai, Z. & Li-Feng, X. (2008). A filtered-beam-search-based heuristic algorithm for flexible job shop scheduling problem. *International Journal Production Research*, Vol. 46, pp. 3027-3058

Tay, J.C.& Ho, N.B. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job shop problems", *Comput. Ind. Eng.*, Vol. 54, pp. 453-473

Yazdani, M., Gholami, M., Zandieh, M. & Mousakhani, M. (2009). A simulated Annealing Algorithm for Flexible Job Shop Scheduling Problem, *Journal of Applied Sciences*, pp. 1-9

Affenzeller, M. & Wagner, S. (2004). SASEGASA: A New Genetic Parallel Evolutionary Algorithm for Achieving Highest Quality Results. *Journal of Heuristics-Special Issue on New Advances on Parallel MetaHeuristics for Complex Problem,* Vol. 10, pp. 239-2630

Back, D., Fogel, B. & Michalewicz Z. (eds.) (1997). *Handbook of Evolutionary Computation,* Oxford University Press and Institute of Physics Publishing, Bristol-NY

Beham, A., Winkler, S., Wagner, S. & M. Affenzeller, M. (2008). A Genetic Programming Approach to Solve Scheduling Problems with Parallel Simulation. *Parallel and Distributed Processing*, pp. 1-5.

Koza, J.R. (1992). *Genetic Programming: On the Programming of Computers by Natural Selection*, MIT Press, Cambridge

Mitchell, M., & Forrest, S. (2005). *Genetic Algorithms and Artificial Life, Artificial Life. An Overview.* , G. Langton (Ed.), MIT Press, 1995.

Mitchell, M., Crutchfield, J. & R. Das, R. (1996). Evolving Cellular Automata with Genetic Algorithms: A Review of Recent Work. *Proceedings of the First International Conference on Evolutionary Computation and Its Applications* , 1996.

Zomaya, A. Y., Ward, C. & Macey, B. (1999). Genetic Scheduling for Parallel Processor Systems: Comperative Studies and Performance Issues, *IEEE Trans. on Parallel and Distributed Systems,* Vol. 10, N 8, pp. 795-812

Stocher, W., Kabelka, B., & Preissl, R. (2007). Automatically Generating Priority Rules for the Flexible Job Shop Problem with Genetic Programming", *Proc. of Computer Aided Systems Theory*, Euro CAST 2007

Winkler, S., Affenzeller, M. & Wagner, S. (2007). Advanced Genetic Programming Based Machine Learning. *Journal of Mathematical Modelling and Algorithms*, Vol. 6, N 3, pp. 455-480

De Castro, L. N. (2006). *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications*, Chapman&Hall/CRC, NY.

Tomassini, M., Sipper, M. & Perrenoud, M. (2000). On the Generation of High-Quality Random Numbers by Two-Dimensional Cellular Automata, *IEEE Trans. Computers*, Vol. 49, N 10, pp. 1140-1151

Seredyński, F. & Święcicka, A. (2002). Immune - like System Approach to Cellular Automata – based Scheduling, *Parallel Processing and Applied Mathematics*, R. Wyrzykowski et al. (Ed.), LNCS 2328, pp. 626-633, Springer Berlin/ Heidelberg

Seredyński, F. & Zomaya A. Y. (2002). Sequential and Parallel Cellular Automata-Based Scheduling Algorithms. IEEE Trans. Parallel Distributed Systems 13(10), pp. 1009-1023

Back, T., & Breukelaar, R. (2005). Using Genetic Algorithms to Evolve Behavior in Cellular Automata, In: *Lecture Notes in Computer Science*, Springer Berlin/ Heidelberg, vol. 3699, pp. 1-10

Kanoh, H., Wu, Y. (2003). Evolutionary Design of Rule Changing Cellular Automata, In: Palade, V., Howlett, R.J., Jain, L.C (Eds.). Knowlegde-Based Intelligent Information and Engineering Systems, 7th International Conference KES 2003, Oxford, UK, September 3-5, 2003, Lecture Notes in Computer Science, Springer Berlin/Haidelberg, Vol. 2773, pp. 258-264

Martins, C. L. M. & de Oliveira, P.P.B. (2005). Evolving Sequential Combinations of Elementary Cellular Automata Rules, In: Capcarrere, M. et al. (Ed.), LNAI 3630, pp. 461-470

Das, R., Mitchell, M &J. P. Crutchfield, J. P. (1994). A Genetic Algorithm Discovers Particle-Based Computation in Cellular Automata, In: Parallel Problem Solving from Nature – PPSN III, Davidor, Y. et al. (Ed.), LNCS 866, Springer, pp. 344-355

Sipper, M. (1997). *Evolving of Parallel Cellular Machines: The Cellular Progrmming Approach*, Springer-Verlag, Heidelberg

Sipper, M. (1999). The Emergence of Cellular Computing, IEEE Computer, Vol. 32, N 7 , pp. 18-26, July 1999

Subrata, R., & Zomaya, A. Y. (2003). Evolving Cellular Automata for Location Management In Mobile Computing, *IEEE Trans. on Parallel and Distributed Systems*, Vol. 14, N 1, pp. 13-26

Sahoo, G. & Kumar, T. (2007). A Genetically based Evolutionary Computing Technique based on Cellular Automata, *International Journal of Computer Science and Network Security*, Vol. 7, N 11, pp. 26-31

Witkowski, T., Antczak, A. & Antczak, P. (2004). Random and Evolution Algorithms of Tasks Scheduling and the Production Scheduling, *Proceedings of International Join Conference on Fuzzy Systems – IEEE 2004*, Budapest, July 2004 vol. 2, pp. 727-732

Witkowski, T., Antczak, P. & Antczak, A. (2005 a). Tabu Search and GRASP used in hybrid procedure for optimize the flexible job shop problem, In: Fuzzy Logic, Soft Computing and Computational Intelligence - *11th* IFSA World Congress, Liu, Y. et al. (Ed.), Tsinghua University Press and Springer, pp. 1620-1625

Witkowski, T., Antczak P.& Antczak A. (2005 b). Application of GRASP procedure for production scheduling and its comparision with other methods. *Journal of Automation and Information Sciences*, Beggel House Inc., New York, Vol. 37, 6(40), pp. 35-40

Witkowski T., Antczak, P. & Antczak, A. (2006). The application of simulated annealing procedure for the flexible job shop scheduling problem, *Proceedings of 11th International Conference.: Information Processing and Management of Uncertainty in Knowledge-Based Systems (Industrial Track),* IPMU 2006, Paris, 2006, pp. 21-26

Witkowski, T., Elzway, S., Antczak, A. & Antczak, P. (2007). Representation of Solutions and Genetic Operators for Flexible Job Shop Problem, In (eds. D.-S. Huang, L. Heutte, and M. Loog): ICIC 2007, CCIS N2: Advanced Intelligent Computing Theories and Applications. Springer-Verlag, Berlin Heidelberg 2007, pp. 256- 265

Witkowski, T., Antczak, A., Elzway, S., Antczak, P. (2009). Evolving Cellular Automata - based Flexible Job Shop Scheduling, 5th International Conference on Natural Computation ICNC'09, Vol. 2 (eds. H. Wang, K. S. Low, K. Wei, & J. Sun); Tianjian, China, 14-16 August 2009, CPS, Los Alamitos, California, Washington, Tokio, pp. 8-13

Cellular Automata - Simplicity Behind Complexity

Edited by Dr. Alejandro Salcido

ISBN 978-953-307-230-2
Hard cover, 566 pages
Publisher InTech
Published online 11, April, 2011
Published in print edition April, 2011

Cellular automata make up a class of completely discrete dynamical systems, which have became a core subject in the sciences of complexity due to their conceptual simplicity, easiness of implementation for computer simulation, and their ability to exhibit a wide variety of amazingly complex behavior. The feature of simplicity behind complexity of cellular automata has attracted the researchers' attention from a wide range of divergent fields of study of science, which extend from the exact disciplines of mathematical physics up to the social ones, and beyond. Numerous complex systems containing many discrete elements with local interactions have been and are being conveniently modelled as cellular automata. In this book, the versatility of cellular automata as models for a wide diversity of complex systems is underlined through the study of a number of outstanding problems using these innovative techniques for modelling and simulation.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen