We are IntechOpen, the world’s leading publisher of Open Access books
Built by scientists, for scientists

3,700 Open access books available
108,500 International authors and editors
1.7 M Downloads

154 Countries delivered to
TOP 1% Our authors are among the most cited scientists
12.2% Contributors from top 500 universities

WEB OF SCIENCE™
Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com
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 automata 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 the 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 is 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 \( H_i \) of equal partial task \( p_i \) of a certain range of production. Operations of technological processing of the \( i \)-th part are denoted by \( \{O_{ij}\}_{j=1}^{H_i} \). Then for \( Z_i \), we can write \( Z_i = (H_i, \{O_{ij}\}_{j=1}^{H_i}, \{\xi_i\}_{j=1}^{H_i}, \{\tau_{ij}\}_{j=1}^{H_i}) \), where \( \xi_i \) 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 set of machines that is assigned to the operation \( O_{ij} \); \( G \) is a set of all groups of machines arising in the matrix \( | \{ Z_i \} |; \xi_i \) is the 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); \( \tau_{ij} \) 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 \( (C_{\text{max}}) \).
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 ($i \in Zd$), in a regular $d$-dimensional uniform lattice, or grid, its neighborhood $G_i$ is defined by

$$G_i = \{i, i + r_1, i + r_2, ..., i + r_n\}$$

where $n$ is a fixed parameter that determines the neighborhood size, and $r_j$ is a fixed vector in the $d$-dimensional space. The local transition rule $f$

$$f : S^n \rightarrow S$$

maps the state $s_i \in S$ of a given cell $i$ into another state from the set $S$, as a function of the states of the cells in the neighborhood $G_i$. 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 $i$, $f_i$. 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 $s_i(t)$ 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

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

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 \( n = \text{Operation}_{i1}[o_{i1}] \), where \( n = \text{Operation}_{i\text{ST}}[o_{i1}] \), which we use to calculate \( D \) where \( D = \text{Operation}_{i\text{RT}}[n] \). 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 \( o_{i1}, o_{i2} \):

\[
\text{Operation}_{i\text{ST}}[o_{i2}] = \text{Operation}_{i\text{ST}}[o_{i2}];
\]

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

Fig. 2. Automata transition tables (allocate machines)

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

<table>
<thead>
<tr>
<th>Op: 0 [0, 0, -1, 0, -1, -1, -1]</th>
<th>Op: 0 [0, 1, 2, 3, 4, 5, 6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op: 0 [1, 0, -1, 0, -1, -1, -1]</td>
<td>Op: 0 [5, 1, 2, 3, 4, 0, 6]</td>
</tr>
<tr>
<td>Op: 0 [1, 0, -1, 0, -1, -1, -1]</td>
<td>Op: 0 [5, 1, 2, 0, 4, 3, 6]</td>
</tr>
<tr>
<td>Op: 0 [1, 0, -1, 1, -1, -1, -1]</td>
<td>Op: 0 [3, 6, 5, 2, 1, 4]</td>
</tr>
<tr>
<td>Op: 0 [1, 0, -1, 2, -1, -1, -1]</td>
<td>Op: 0 [3, 0, 5, 6, 2, 1, 4]</td>
</tr>
<tr>
<td>Op: 1 [0, 0, -1, 0, -1, -1, -1]</td>
<td>Op: 1 [0, 1, 2, 3, 4, 5, 6]</td>
</tr>
<tr>
<td>Op: 1 [1, 0, 0, -1, 0, -1, -1]</td>
<td>Op: 1 [2, 1, 0, 3, 4, 5, 6]</td>
</tr>
<tr>
<td>Op: 1 [1, 0, 0, -1, 0, -1, -1]</td>
<td>Op: 1 [2, 1, 4, 3, 0, 5, 6]</td>
</tr>
<tr>
<td>Op: 1 [0, 0, 0, -1, 0, -1, -1]</td>
<td>Op: 1 [4, 3, 6, 5, 0, 2, 1]</td>
</tr>
<tr>
<td>Op: 1 [1, 0, 0, -1, 0, -1, -1]</td>
<td>Op: 1 [4, 3, 6, 5, 1, 2, 0]</td>
</tr>
<tr>
<td>Op: 2 [-1, -1, -1, 0, 0, 0, 0]</td>
<td>Op: 2 [0, 1, 2, 3, 4, 5, 6]</td>
</tr>
<tr>
<td>Op: 2 [-1, -1, -1, 0, 0, 0, 0]</td>
<td>Op: 2 [6, 1, 2, 3, 4, 5, 0]</td>
</tr>
<tr>
<td>Op: 2 [-1, -1, -1, 0, 0, 0, 0]</td>
<td>Op: 2 [6, 1, 2, 3, 4, 0, 5]</td>
</tr>
<tr>
<td>Op: 2 [-1, -1, -1, 2, 1, 2, 1]</td>
<td>Op: 2 [5, 6, 1, 2, 3, 0, 4]</td>
</tr>
<tr>
<td>Op: 2 [-1, -1, -1, 2, 1, 2, 2]</td>
<td>Op: 2 [5, 6, 1, 2, 0, 3, 4]</td>
</tr>
<tr>
<td>Op: 3 [-1, -1, -1, 0, 0, 0, 0]</td>
<td>Op: 3 [0, 1, 2, 3, 4, 5, 6]</td>
</tr>
<tr>
<td>Op: 3 [-1, -1, -1, 0, 0, 0, 0]</td>
<td>Op: 3 [4, 1, 2, 3, 0, 5, 6]</td>
</tr>
<tr>
<td>Op: 3 [-1, -1, -1, 0, 0, 0, 0]</td>
<td>Op: 3 [4, 0, 2, 3, 1, 5, 6]</td>
</tr>
<tr>
<td>Op: 3 [-1, -1, -1, 1, 1, 1]</td>
<td>Op: 3 [1, 0, 6, 4, 5, 2, 3]</td>
</tr>
<tr>
<td>Op: 3 [-1, -1, -1, 1, 1, 1]</td>
<td>Op: 3 [1, 2, 6, 4, 5, 0, 3]</td>
</tr>
<tr>
<td>Op: 3 [-1, -1, -1, 1, 1, 1]</td>
<td>Op: 3 [-1, -1, -1, -1, 1, 1]</td>
</tr>
</tbody>
</table>

State table: 1

<table>
<thead>
<tr>
<th>Op: 0 [0, 0, -1, 0, -1, -1, -1]</th>
<th>Op: 0 [0, 1, 2, 3, 4, 5, 6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op: 0 [1, 0, -1, 0, -1, -1, -1]</td>
<td>Op: 0 [6, 1, 2, 3, 4, 5, 0]</td>
</tr>
<tr>
<td>Op: 0 [1, 0, -1, 0, -1, -1, -1]</td>
<td>Op: 0 [6, 1, 2, 3, 4, 0, 5]</td>
</tr>
<tr>
<td>Op: 0 [2, 0, -1, 2, -1, -1, -1]</td>
<td>Op: 0 [5, 6, 1, 2, 3, 0, 4]</td>
</tr>
<tr>
<td>Op: 0 [2, 0, -1, 2, -1, -1, -1]</td>
<td>Op: 0 [5, 6, 1, 2, 0, 3, 4]</td>
</tr>
<tr>
<td>Op: 1 [0, 0, -1, 0, -1, -1, -1]</td>
<td>Op: 1 [0, 1, 2, 3, 4, 5, 6]</td>
</tr>
<tr>
<td>Op: 1 [1, 0, 0, -1, 0, -1, -1]</td>
<td>Op: 1 [2, 1, 0, 3, 4, 5, 6]</td>
</tr>
<tr>
<td>Op: 1 [1, 0, 0, -1, 0, -1, -1]</td>
<td>Op: 1 [2, 1, 4, 3, 0, 5, 6]</td>
</tr>
<tr>
<td>Op: 1 [2, 0, 0, -1, 0, -1, -1]</td>
<td>Op: 1 [4, 3, 6, 5, 0, 2, 1]</td>
</tr>
<tr>
<td>Op: 1 [2, 0, 0, -1, 0, -1, -1]</td>
<td>Op: 1 [4, 3, 6, 5, 1, 2, 0]</td>
</tr>
<tr>
<td>Op: 2 [-1, -1, -1, 0, 0, 0, 0]</td>
<td>Op: 2 [0, 1, 2, 3, 4, 5, 6]</td>
</tr>
<tr>
<td>Op: 2 [-1, -1, -1, 0, 0, 0, 0]</td>
<td>Op: 2 [4, 1, 2, 3, 0, 5, 6]</td>
</tr>
<tr>
<td>Op: 2 [-1, -1, -1, 0, 0, 0, 0]</td>
<td>Op: 2 [4, 0, 2, 3, 1, 5, 6]</td>
</tr>
<tr>
<td>Op: 2 [-1, -1, -1, 2, 0, 0, 0]</td>
<td>Op: 2 [1, 0, 6, 4, 5, 2, 3]</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Op: 2 [-1, -1, -1, 2, 0, 0, 0]</td>
<td>Op: 2 [1, 2, 6, 4, 5, 0, 3]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Op: 3 [-1, -1, -1, 0, 0, 0]</th>
<th>Op: 3 [0, 1, 2, 3, 4, 5, 6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op: 3 [-1, -1, -1, 0, 0, 0]</td>
<td>Op: 3 [6, 1, 2, 3, 4, 5, 0]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Op: 3 [-1, -1, -1, -1, 0, 0, 0]</th>
<th>Op: 3 [6, 1, 2, 3, 4, 0, 5]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Op: 3 [-1, -1, -1, -1, 0, 0, 0]</th>
<th>Op: 3 [5, 6, 1, 2, 3, 0, 4]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Op: 3 [-1, -1, -1, -1, 2, 1, 0]</th>
<th>Op: 3 [5, 6, 1, 2, 0, 3, 4]</th>
</tr>
</thead>
</table>

State table: 2

<table>
<thead>
<tr>
<th>Op: 0 [0, 0, -1, 0, -1, -1, -1]</th>
<th>Op: 0 [0, 1, 2, 3, 4, 5, 6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op: 0 [1, 0, -1, 0, -1, -1, -1]</td>
<td>Op: 0 [5, 1, 2, 3, 4, 0, 6]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Op: 0 [1, 0, -1, 0, -1, -1, -1]</th>
<th>Op: 0 [5, 1, 2, 0, 4, 3, 6]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Op: 0 [0, 0, -1, 0, -1, -1, -1]</th>
<th>Op: 0 [3, 6, 5, 0, 2, 1, 4]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Op: 0 [0, 0, -1, 0, -1, -1, -1]</th>
<th>Op: 0 [3, 0, 5, 6, 2, 1, 4]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Op: 2 [-1, -1, -1, 0, 0, 0]</th>
<th>Op: 2 [0, 1, 2, 3, 4, 5, 6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op: 2 [-1, -1, -1, 0, 0, 0]</td>
<td>Op: 2 [0, 1, 2, 3, 4, 5, 6]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Op: 2 [-1, -1, -1, 1, 0, 0]</th>
<th>Op: 2 [0, 1, 2, 3, 4, 5, 6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op: 2 [-1, -1, -1, 1, 0, 0]</td>
<td>Op: 2 [0, 1, 2, 3, 4, 5, 6]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Op: 3 [-1, -1, -1, 0, 0, 0]</th>
<th>Op: 3 [0, 1, 2, 3, 4, 5, 6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op: 3 [-1, -1, -1, 0, 0, 0]</td>
<td>Op: 3 [5, 1, 2, 3, 4, 0, 6]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Op: 3 [-1, -1, -1, 0, 0, 0]</th>
<th>Op: 3 [5, 1, 2, 0, 4, 3, 6]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Op: 3 [-1, -1, -1, 1, 0, 0]</th>
<th>Op: 3 [3, 6, 5, 0, 2, 1, 4]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op: 3 [-1, -1, -1, 1, 0, 0]</td>
<td>Op: 3 [3, 0, 5, 6, 2, 1, 4]</td>
</tr>
</tbody>
</table>

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 makespans 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 makespans for each schedule with population are writing
and compare to aim choice best schedule among current populations. If population no is
latest we realize crossover 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.

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

<table>
<thead>
<tr>
<th>hybridization ratio = 0.9; intensity of hybridization = 0.9.</th>
<th>size of the population = 1000; number of transitions = 10.</th>
</tr>
</thead>
<tbody>
<tr>
<td>iter.</td>
<td>MakeSpan</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>10</td>
<td>66295.6</td>
</tr>
<tr>
<td></td>
<td>66295.6</td>
</tr>
<tr>
<td>100</td>
<td>67436.6</td>
</tr>
<tr>
<td></td>
<td>66520.1</td>
</tr>
<tr>
<td>1000</td>
<td>66305.5</td>
</tr>
<tr>
<td></td>
<td>67530.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>hybridization ratio = 0.9; intensity of hybridization = 0.9.</th>
<th>size of the population = 1000; number of transitions = 100.</th>
</tr>
</thead>
<tbody>
<tr>
<td>iter.</td>
<td>MakeSpan</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>10</td>
<td>72554.0</td>
</tr>
<tr>
<td></td>
<td>71397.2</td>
</tr>
<tr>
<td>100</td>
<td>70852.5</td>
</tr>
<tr>
<td></td>
<td>71786.4</td>
</tr>
<tr>
<td>1000</td>
<td>66239.3</td>
</tr>
<tr>
<td></td>
<td>60118.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>hybridization ratio = 0.9; intensity of hybridization = 0.9.</th>
<th>size of the population = 1000; number of transitions = 1000.</th>
</tr>
</thead>
<tbody>
<tr>
<td>iter.</td>
<td>MakeSpan</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>10</td>
<td>70217.3</td>
</tr>
<tr>
<td></td>
<td>71086.2</td>
</tr>
<tr>
<td>100</td>
<td>67937.3</td>
</tr>
<tr>
<td></td>
<td>69405.4</td>
</tr>
<tr>
<td>1000</td>
<td>64347.2</td>
</tr>
<tr>
<td></td>
<td>67208.2</td>
</tr>
</tbody>
</table>

www.intechopen.com
Figure 5 summarizes the results for the test problems that were run with the evolving cellular automata algorithm for the 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 $C_{\text{max}}$ criterion) - very good (large population size), average (medium population size) and poor (small population size). Moreover an increase of the IT value influences the $C_{\text{max}}$ more than the RT value, although there are a number of exceptions.

The best results of $C_{\text{max}}$ are always obtained at SP(D) regardless of the IT or RT values. Eg. at SP(D) value the best $C_{\text{max}}$ 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 $C_{\text{max}}$ 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 $C_{\text{max}}$ 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 500 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 10 to 100 gives an increase of $C_{\max}$ from 75000 to 70000 min., i.e. by ca. 5000 min.; moreover an increase of the IT value from 100 to 1000 gives an insignificant decrease of $C_{\max}$ while (HR, IH) = (0,5,0,5) and a decrease of $C_{\max}$ while (HR, IH) = (0,1,0,1) 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 90000 to 67000 min., i.e. by ca. 2000 min. An increase of IT from 100 to 1000 decreases the $C_{\max}$ value from 77000 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(S)-RT(S). Moreover the $C_{\max}$ value increases in the following order: from IT(D)-SP(D)-RT(D) 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 $C_{\text{max}}$ than IT(M)-SP(M)-RT(M) while (HR, IH) = (0, 1; 0, 1) and (HR, IH) = (0, 9; 0, 9) and a better $C_{\text{max}}$ 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 $C_{\text{max}}$ than for IT(D)-SP(D)-RT(D). Similarly combination IT(S)-SP(D)-RT(M) gives a better $C_{\text{max}}$ than IT(S)-SP(D)-RT(S) and IT(M)-SP(D)-RT(M) gives a better $C_{\text{max}}$ than IT(M)-SP(D)-RT(S).

In the group of poorest $C_{\text{max}}$ values (with SP(M) value) we can observe that the best $C_{\text{max}}$ 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 $C_{\text{max}}$ 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 the pair (HR, IH) = (0, 9; 0, 9) - twice.

In the group of average $C_{\text{max}}$ values (with P(S) value) we can observe that the best $C_{\text{max}}$ 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 $C_{\text{max}}$ is achieved while the pair is (0, 1; 0, 1) - 4 times, while (0, 5; 0, 5) - once and at pair (0, 9; 0, 9) - 3 times; moreover the worst $C_{\text{max}}$ (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 $C_{\text{max}}$ 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 $C_{\text{max}}$ 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 $C_{\text{max}}$ 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 $C_{\text{max}}$ is achieved from 69980 to 67351 compared to IT(M)-SP(D)-RT(D), and for combination IT(S)-SP(V)-RT(D) a decrease of $C_{\text{max}}$ was achieved from 68057 to 63840 min. compared to IT(S)-SP(D)-RT(D). For combination IT(M)-SP(V)-RT(D) a decrease of $C_{\text{max}}$ 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 $C_{\text{max}}$ - from 68057 to 63840 min. compared to IT(S)-SP(D)-RT(D).

When analyzing the influence of IT(V) on $C_{\text{max}}$ we can note that almost in all the analyzed cases the an increase from IT(D) to IT(V) gives an insignificant decrease of the $C_{\text{max}}$ value eg. for combination IT(V)-SP(D)-RT(D) compared to IT(D)-SP(D)-RT(D), 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 73647-72115 = 1432 min.

When analyzing the influence of RT(V) eg. in combinations IT(M)-SP(M)-RT(D) at RT(D) we can observe both increases and decreases of the $C_{\text{max}}$ value. For combinations with the SP(D) value (fig. 5) the CA algorithm has a better $C_{\text{max}}$ in all cases as compared to the combinations with the SP(M) value and has a better $C_{\text{max}}$ 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 $C_{\text{max}}$ in almost all cases as compared to the combinations with the SP(M) value and has a worse $C_{\text{max}}$ 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 $C_{\text{max}}$ in almost all cases as compared to the combinations with the SP(S) value and has a worse $C_{\text{max}}$ 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 the CA algorithm, and Table 2 shows some of the results for the GA algorithm (serial route).

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

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

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.

<table>
<thead>
<tr>
<th>Iter.</th>
<th>Min.</th>
<th>Average</th>
<th>Max.</th>
<th>Avg. time [sec.]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>37543.2</td>
<td>38625.2</td>
<td>36427.7</td>
<td>35869.5</td>
</tr>
<tr>
<td>100</td>
<td>37661.2</td>
<td>36350.0</td>
<td>37456.5</td>
<td>38124.6</td>
</tr>
<tr>
<td>1000</td>
<td>36300.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>PM 1/1000</th>
<th>PC 1/1000</th>
<th>Value of $C_{max}$ for different parameters of the GA algorithm [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>37545</td>
</tr>
<tr>
<td>1000</td>
<td>450</td>
<td>38926</td>
</tr>
<tr>
<td>192</td>
<td>256</td>
<td>37368</td>
</tr>
<tr>
<td>96</td>
<td>353</td>
<td>37729</td>
</tr>
<tr>
<td>256</td>
<td>450</td>
<td>40018</td>
</tr>
<tr>
<td>64</td>
<td>256</td>
<td>40359</td>
</tr>
<tr>
<td>128</td>
<td>64</td>
<td>39775</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>41088</td>
</tr>
<tr>
<td>8</td>
<td>128</td>
<td>40200</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>40942</td>
</tr>
<tr>
<td>8</td>
<td>450</td>
<td>36686</td>
</tr>
<tr>
<td>32</td>
<td>450</td>
<td>37158</td>
</tr>
<tr>
<td>32</td>
<td>256</td>
<td>39961</td>
</tr>
<tr>
<td>32</td>
<td>128</td>
<td>40567</td>
</tr>
<tr>
<td>64</td>
<td>128</td>
<td>42941</td>
</tr>
</tbody>
</table>

www.intechopen.com
<table>
<thead>
<tr>
<th>64</th>
<th>256</th>
<th>38145</th>
<th>38874</th>
<th>36901</th>
<th>39209</th>
<th>38915</th>
<th>40416</th>
<th>39649</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>450</td>
<td>39013</td>
<td>39672</td>
<td>39344</td>
<td>40236</td>
<td>39826</td>
<td>39775</td>
<td>41053</td>
</tr>
<tr>
<td>128</td>
<td>353</td>
<td>38671</td>
<td>38903</td>
<td>39186</td>
<td>39422</td>
<td>37540</td>
<td>36506</td>
<td>38223</td>
</tr>
<tr>
<td>512</td>
<td>353</td>
<td>40968</td>
<td>39619</td>
<td>40180</td>
<td>39583</td>
<td>38870</td>
<td>40559</td>
<td>36429</td>
</tr>
<tr>
<td>192</td>
<td>256</td>
<td>37466</td>
<td>39257</td>
<td>38295</td>
<td>38501</td>
<td>38132</td>
<td>40304</td>
<td>39477</td>
</tr>
<tr>
<td>96</td>
<td>256</td>
<td>39444</td>
<td>38595</td>
<td>38905</td>
<td>38402</td>
<td>38774</td>
<td>38134</td>
<td>39817</td>
</tr>
<tr>
<td>64</td>
<td>353</td>
<td>37770</td>
<td>38677</td>
<td>39667</td>
<td>40263</td>
<td>39317</td>
<td>37676</td>
<td>39445</td>
</tr>
<tr>
<td>16</td>
<td>353</td>
<td>39930</td>
<td>39980</td>
<td>41221</td>
<td>38075</td>
<td>38681</td>
<td>39236</td>
<td>40329</td>
</tr>
<tr>
<td>280</td>
<td>130</td>
<td>40586</td>
<td>39648</td>
<td>36805</td>
<td>38176</td>
<td>38239</td>
<td>36077</td>
<td>39783</td>
</tr>
<tr>
<td>280</td>
<td>370</td>
<td>39253</td>
<td>38892</td>
<td>38222</td>
<td>38726</td>
<td>38130</td>
<td>40813</td>
<td>39168</td>
</tr>
<tr>
<td>840</td>
<td>130</td>
<td>41824</td>
<td>37050</td>
<td>33949</td>
<td>37923</td>
<td>37435</td>
<td>43783</td>
<td>39065</td>
</tr>
<tr>
<td>910</td>
<td>370</td>
<td>37426</td>
<td>37692</td>
<td>38223</td>
<td>38467</td>
<td>37562</td>
<td>40862</td>
<td>39039</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>38436</td>
<td>41069</td>
<td>37987</td>
<td>39287</td>
<td>38690</td>
<td>39993</td>
<td>39042</td>
</tr>
<tr>
<td>16</td>
<td>64</td>
<td>37505</td>
<td>38351</td>
<td>36647</td>
<td>39846</td>
<td>34738</td>
<td>40428</td>
<td>39065</td>
</tr>
<tr>
<td>32</td>
<td>64</td>
<td>38554</td>
<td>39284</td>
<td>38144</td>
<td>38171</td>
<td>34093</td>
<td>39474</td>
<td>38144</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td>40415</td>
<td>39862</td>
<td>36583</td>
<td>38454</td>
<td>36264</td>
<td>39039</td>
<td>40426</td>
</tr>
<tr>
<td>96</td>
<td>64</td>
<td>40342</td>
<td>39791</td>
<td>39055</td>
<td>39394</td>
<td>39872</td>
<td>38728</td>
<td>37385</td>
</tr>
<tr>
<td>192</td>
<td>64</td>
<td>40652</td>
<td>36435</td>
<td>38656</td>
<td>40164</td>
<td>38549</td>
<td>37849</td>
<td>37345</td>
</tr>
<tr>
<td>256</td>
<td>64</td>
<td>40814</td>
<td>38851</td>
<td>40186</td>
<td>40448</td>
<td>39435</td>
<td>41031</td>
<td>37419</td>
</tr>
<tr>
<td>280</td>
<td>64</td>
<td>38278</td>
<td>38972</td>
<td>38653</td>
<td>38102</td>
<td>37330</td>
<td>38661</td>
<td>39112</td>
</tr>
<tr>
<td>512</td>
<td>64</td>
<td>38788</td>
<td>37774</td>
<td>39794</td>
<td>38820</td>
<td>40603</td>
<td>39224</td>
<td>39700</td>
</tr>
<tr>
<td>840</td>
<td>64</td>
<td>40521</td>
<td>38146</td>
<td>37848</td>
<td>38323</td>
<td>38501</td>
<td>38126</td>
<td>37391</td>
</tr>
<tr>
<td>910</td>
<td>64</td>
<td>39983</td>
<td>39467</td>
<td>38609</td>
<td>35735</td>
<td>38880</td>
<td>37447</td>
<td>41173</td>
</tr>
<tr>
<td>1000</td>
<td>64</td>
<td>38465</td>
<td>40961</td>
<td>36692</td>
<td>38726</td>
<td>39597</td>
<td>38493</td>
<td>37194</td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td>38645</td>
<td>38577</td>
<td>39785</td>
<td>39314</td>
<td>39181</td>
<td>39658</td>
<td>37034</td>
</tr>
<tr>
<td>96</td>
<td>128</td>
<td>40260</td>
<td>39932</td>
<td>39688</td>
<td>39593</td>
<td>39786</td>
<td>38340</td>
<td>38967</td>
</tr>
<tr>
<td>128</td>
<td>128</td>
<td>39873</td>
<td>39150</td>
<td>37624</td>
<td>39528</td>
<td>40273</td>
<td>38549</td>
<td>40675</td>
</tr>
<tr>
<td>192</td>
<td>128</td>
<td>38559</td>
<td>39003</td>
<td>36695</td>
<td>40691</td>
<td>38635</td>
<td>39235</td>
<td>39041</td>
</tr>
<tr>
<td>840</td>
<td>128</td>
<td>38322</td>
<td>39987</td>
<td>37723</td>
<td>40669</td>
<td>39489</td>
<td>38108</td>
<td>39049</td>
</tr>
<tr>
<td>910</td>
<td>128</td>
<td>37698</td>
<td>39438</td>
<td>37920</td>
<td>35857</td>
<td>40082</td>
<td>39579</td>
<td>34667</td>
</tr>
<tr>
<td>1000</td>
<td>128</td>
<td>38599</td>
<td>39670</td>
<td>39822</td>
<td>37052</td>
<td>38917</td>
<td>38312</td>
<td>40619</td>
</tr>
<tr>
<td>8</td>
<td>130</td>
<td>38254</td>
<td>39256</td>
<td>37131</td>
<td>37520</td>
<td>40000</td>
<td>39848</td>
<td>39644</td>
</tr>
</tbody>
</table>

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

www.intechopen.com
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. Moreover, 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


www.intechopen.com


Cellular automata make up a class of completely discrete dynamical systems, which have become 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.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:
