Oriented Evolutionary Simulation for the Parallel Machine Scheduling Problem with Setup Times

Prof. Csaba FABIAN PhD., Asist. Iulian ÎNTORSUREANU
Academy of Economic Studies Bucharest, Romania

Abstract: *The Parallel Scheduling Problem (PSP) with setup times is of significant importance in praxis, but it is NP-complete, and thus exact solving methods are not appropriate. The Local Search (LS) process can lead to local optima. In order to avoid this with greater probability, an oriented simulation process is proposed, using evolutionary algorithms with specific mutation and crossover operators.*
Keywords*: evolutionary algorithms, mutation, crossover, Parallel Scheduling Problem.*

Parallel Machine Scheduling Problem with Setup Times

The Parallel Machine Scheduling Problem consists of determining a schedule for the processing of *n* jobs on *m* parallel, identical machines, so that the total processing time is minimal. Setup times are time intervals between two consecutive jobs on a machine, required for the reconfiguration of the machine.

A PSP with setup times is therefore noted with P / s(i,j) / C_{max}, where:

- *m* – number of machines
- *n* – number of jobs (orders)
- *s(i,j* – setup times (non-symmetrical matrix)
- *d(j)* – duration of job *j*
- *C(j* – processing time of job *j*
- *MC= max C(j)* – cycle time, measures the solution performance
- A(i,j) – distribution of jobs on machines
- q(i) – number of jobs allocated on machine i
- *MC(i)* – total processing time for machine *i*

$$MC(i) = \sum_{j=1}^{q(i)} d(A(i, j)) + \sum_{j=1}^{q(i)-1} s(A(i, j), A(i, j+1))$$

A possible solution is noted as the matrix *A(i,j),* where elements of a line *i* represent the jobs allocated to the *i* machine.
The quality of a solution is given by its cycle time CM, that we are trying to minimize.

Evolutionary Algorithms

Evolutionary algorithms are a generalization of genetic algorithms. The chromosomes are not vectors of {0,1} values, but problem-dependent, more general vectors.

The representation of a PSP solution:
The chromosome is obtained from the sequence of the lines of matrix A, delimited with zeroes.
P = (A(1,1), A(1,2), … , A(1, q(1)), Ø, … ,
Ø, A(i,1), A(i,2), … , A(i, q(i)), Ø, … ,
Ø, A(m,1), A(m,2), … , A(m, q(m)))

A general chromosome will be noted with:
P = (p(1), p(2), ... , p(i), ... , p(r))
the total length being

$$r = n + m - 1$$

(number of jobs plus number of separators).

Example:
Chromosome: (1, 2, 5, 6, Ø, 3, 7, 9, 10, Ø, 4, 8, 11, 12) is obtained from solution

$$A = \begin{pmatrix} 1 & 2 & 5 & 6 \\ 3 & 7 & 9 & 10 \\ 4 & 8 & 11 & 12 \end{pmatrix}$$

of a PSP with n = 12, m=3.

Evolutionary algorithms start with an initial population of chromosomes (individuals). Then, chromosomes of this population are selected (parents), and are subject to simple mutations and/or crossover mutations. The resulting chromosomes (offsprings) are evaluated and the population modified by keeping the valuable individuals (better solutions) and rejecting the weaker ones, thus obtaining a new generation. The process continues for a given number of generations, or until the improvements of the population stops.

Oriented Simulation

A deterministic mode of selecting jobs would be:

Greedy mode (GR)

Let $\delta_j = s_{p(j),p(j+1)}$, j=1..n-1 and $\delta_n = s_{p(n),p(1)}$

$\Delta\delta_j = \delta_j - \delta_n$, j=1..n-1

k = { j | max ($\Delta\delta_j$) , $\Delta\delta_j > 0$ }

Note: machines or parents can also be selected this way:

k = { i | max (MC(i)), i=1..m }

k = { j | max (C_{max}(j)), j=1..p }, p = population size

Oriented Simulation

Let $D = \sum_{j=1}^{n} d_j$ and $h_k = \sum_{j=1}^{k} \frac{d_j}{D}$,

k = 1..n and $\eta_0 = 0$

We define

k = { j | $\eta_{j-1} \le \xi \le \eta_j$, j = 1..n), where $\xi \in [0,1]$.

A. Swap Based Mutation

For a given chromosome

P = (p(1), p(2), ... , p(i), ... , p(r))

the SBM on position k consists of swapping p(k-1) ↔ p(k) or p(k) ↔ p(k+1). This is called 1SBM.

It can be generalized, e.g. 2SBM, ... tSBM:

2SBM: p(k-2) ↔ p(k) or p(k) ↔ p(k+2)

*t*SBM: p(k-t) ↔ p(k) or p(k) ↔ p(k+t)

For the PSP problem, the mutation position k can be chosen at:

– a machine at which C_{max} is reached

– all machines, with regard to the separators

B. Order Based Mutation

For a given chromosome

P = (p(1), p(2), ... , p(i), ... , p(r))

the OBM consists of swapping two sequences in the chromosome; given the positions i,j and lengths u,v:

p(i), p(i+1),...,p(i+u)↔p(j), p(j+1),..., p(j+v)

This is generally noted with *u*OBM*v*.

For the PSP problem, the mutation positions i,j can be chosen at:

– Both i and j from a machine at which C_{max} is reached

– i from a machine at which C_{max} is reached and j through oriented simulation

– lengths u,v should not get past the machine separator

C. Position Based Mutation

For a given chromosome

P = (p(1), p(2), ... , p(i), ... , p(r))

the PBM*u* consists of bringing a sequence starting at *i* and with a length of *u* on the position *j*:

p(j-1), p(j), p(j+1),..., p(i), p(i+1),...,p(i+u), ... becomes

p(j-1), p(i), p(i+1), ... ,p(i+u), p(j), p(j+1)

For the PSP problem, the mutation positions *i,j* and the length *u* can be chosen at:

– *i* at a machine at which C_{max} is reached, with u not exceeding the machine sequence; *j* at a machine with a low proceesing time MC(i)

– *i,j* in the same machine sequence, *u* with regard to the separator

D. Partially Mapped Crossover (PMX)
Given a position k, the gene sequence p(1)..p(k-1) is taken from one parent and the rest p(k) .. p(r) from the other parent.

? ' = (p'(1), p'(2), … , p'(k), … , p'(r))
? ''= (p''(1), p''(2),… , p''(k), … , p''(r))

? = (p'(k), p'(k+1), … , p'(r), p''(1), p''(2), … , p''(k-1))

For the PSP problem, chromosomes entering the crossover can be combined taking k at a separator position.

E. Generalized Ordered Crossover (GOX)
$GOX_{i,u,j}$
The offspring is obtained by overwriting a sequence p(i) .. p(i+u) from one parent with the corresponding sequence from the other parent.

? ' = (p'(1),…, p'(i),… , p'(i+u), … , p(r))
? '' = (p''(1), … , p''(j), … , p(r))

? = (p''(1), p''(2), … , p''(j-1), p'(i), p'(i+1), … , p'(i+u), p''(j-1+i+u), … , p''(r))
Note:
Particular case 1: j=1
Particular case 2: i+u=r

For the PSP problem, for entering the GOX crossover can be chosen:
– One (duplicated) chromosome, combining the genes corresponding to two different machines
– Two chromosomes, combining the genes corresponding to the same machine or two different machines

Conclusions
The most appropriate evolutionary operators seem to be the mutations, especially the OBM, because they provide meaningful changes of a solution (changes in the order of jobs on the same machine, changes of jobs or job sequences between machines). The chromosomes are essentially - disregarding separators - (1,n) permutations of jobs, and mutations preserve this condition.
On the other hand, crossover operators tend to lead to offsprings which are not feasible solutions, because of duplication of certain jobs and consequent loss of other jobs. This makes further corrections of the offspring chromosomes necessary, affecting the execution time of the algorithm.

Literature

[1] Mihalewicz, Z. *"Genetic Algorithms + Data Structures = Evolution Programs"* 3rd edition, Springer-Verlag, Berlin, 1996
[2] Mattfeld, D.C. *"Evolutionary Search and the Job Shop - Investigations on Genetic Algorithms for Production Scheduling"*
Physica Verlag, 1996
[3] Fabian, Cs., Mihalca, R., Întorsureanu, I., Uta, A.
"*Modeling and Simulation of Production Processes Using Neural Networks and Genetic Algorithms*", Research project A5 , Ministry of Research and Technology 1996-1999.
[4] Fabian, Cs., Simion, F. *"Experiences with Evolutive Mutations for Job Shop Problems*"
Information Technology, the Proceedings of the Fourth International Symposium, 1999
INFOREC Printing House, Bucharest, Romania 1999