Jop-Shop Scheduling

How to solve a Jop-shop scheduling problem with Genetic Algorithms?

Rick van Aalst, 20236338

Content

- Description
- Problem Definition
- Objective and Conditions
- Operators (GA)
- Algorithm Setup
- Results
- Conclusions and Limitations

What is Jop-Shop Scheduling?

- **Job Shop Scheduling:** In manufacturing or production environments, genetic algorithms can be used to optimize the scheduling of jobs on machines, considering constraints such as processing times, machine availability, and precedence relationships between tasks.
- Genetic algorithms can be applied to find an optimal or near-optimal schedule for more complex instances of the Job Shop Scheduling problem, considering additional factors such as machine constraints, job priorities, and setup times between operations.

Survey on Shop-Scheduling methods

GENETIC ALGORITHMS FOR SHOP SCHEDULING PROBLEMS: A SURVEY

*Frank Werner**
Otto-von-Guericke-Universität, Fakultät für Mathematik, 39106 Magdeburg, Germany

What is Shop Scheduling?

- A set of n jobs (J_1, J_2, J_3, …, J_n) which are processed on
  A set of m machines (M_1, M_2, M_3, …, M_m)
- The processing of a job $\boldsymbol{J_i}$ on a particular machine $\boldsymbol{M_j}$ is denoted as an operation and noted by *(i, j)*
- Each job $\boldsymbol{J_i}$ consists of a number $\boldsymbol{n_i}$ of operations
- For the <u>deterministic</u> scheduling problems, the processing time $\boldsymbol{p_{ij}}$ of each operation *(i, j)* is given in advance

Problem Definition

- For describing these problems, there exists three classifications:
  - **α** (Machine Environment, e.g. Flow Shop, Jop-shop etc.)
  - **β** (Job characteristics, e.g. release dates, due dates, and weights etc.)
  - **γ** (Optimization Criterion, e.g. minimize- total makespan and maximized lateness etc.)
- According to the restrictions on the technological routes of the jobs, we distinguish a **flow shop**, a **job shop** and an **open shop**

Problem Definition (Shops)

- Flow Shop (**α = F**)
  - Each job J_i has exactly *m* operations
  - Same route
  - Assumption: M_1 -> M_2 -> M_3 -> M_m
- Job Shop (**α = J**)
  - A specific route for each job
    - M_{j_1} -> M_{j_2} -> M_{j_3} -> $M_{j_{n_i}}$ for each job J_i (1 < i < n)
  - Number of operations smaller, equal or larger then *m*
    - <u>Different notation</u>: *(i, j, k)* -> k_{th} processing of job J_i on machine M_j
- Open Shop (**α = O**)
  - No routes for jobs
  - Assumed every job has to be processed on every machine
- There exists Generalizations on these shops such as mixed- and general shop
- There exists extensions to shops: **hybrid or flexible shops**
  - Multi-stage and parallel problems

Problem Definition (Constraints)

- For any job $\mathbf{J_i}$ there might be a release date $\mathbf{r_i}$, a due date $\mathbf{d_i}$ and/or a weight $\mathbf{w_i}$
- Other constraints such as no waiting times between operations of a job (**β = no wait**), sequence-(in)dependant set-up times between processing of operations

Problem Definition (Optimality criterion)

- γ indicates optimality criterion
- Minimization of makespan C_{max}
- Minimization of sum of weighted completion time $\sum w_i C_i$
- Minimization of sum of weighted tardiness $\sum w_i T_i$
- Or problems without weights ($w_i = 1$)

Feasible Solutions

- Specify Job orders on the machines
  - Job Sequence
  - Combining routes and job orders into rank matrix *R* where r_{ij} denotes the rank of operation (i, j)
- Example Matrix (*n x m*):

$$R = \begin{pmatrix} 4 & 3 & 2 & 1 \\ 2 & 4 & 1 & 5 \\ 1 & 5 & 3 & 4 \end{pmatrix}$$

$$\begin{array}{lccccc} M_1: & J_3 & \rightarrow & J_2 & \rightarrow & J_1 \\ M_2: & J_1 & \rightarrow & J_2 & \rightarrow & J_3 \\ M_3: & J_2 & \rightarrow & J_1 & \rightarrow & J_3 \\ M_4: & J_1 & \rightarrow & J_3 & \rightarrow & J_2 \end{array}$$

$$\begin{array}{lccccccc} J_1: & M_4 & \rightarrow & M_3 & \rightarrow & M_2 & \rightarrow & M_1 \\ J_2: & M_3 & \rightarrow & M_1 & \rightarrow & M_2 & \rightarrow & M_4 \\ J_3: & M_1 & \rightarrow & M_3 & \rightarrow & M_4 & \rightarrow & M_2 \end{array}$$

Genetic Algorithms

- Representation:
    - Not clear representation, depends on problem
    - Several encoding strategies exists

operation sequence	1,1	2,3	2,1	3,2	1,3	3,1	2,2	3,3	1,2
equivalent representations:									
job repetitions	1	2	2	3	1	3	2	3	1
operation positions	1	9	5	3	7	2	5	4	8
one-dimensional operation sequence	1	6	4	8	3	7	5	9	2

Figure 1. Operation-based representation

Operation-based

- Each job consists ***m*** operations, so each chromosome has ***nm*** length (each gene represents operation *(i, j)*)
- Several variations of representations

operation sequence	1,1	2,3	2,1	3,2	1,3	3,1	2,2	3,3	1,2
equivalent representations:									
job repetitions	1	2	2	3	1	3	2	3	1
operation positions	1	9	5	3	7	2	5	4	8
one-dimensional operation sequence	1	6	4	8	3	7	5	9	2

Figure 1. Operation-based representation

Operation-based

- Job repetitions
  - Each gene contains a job index *i*
  - Repeats m times for a given route

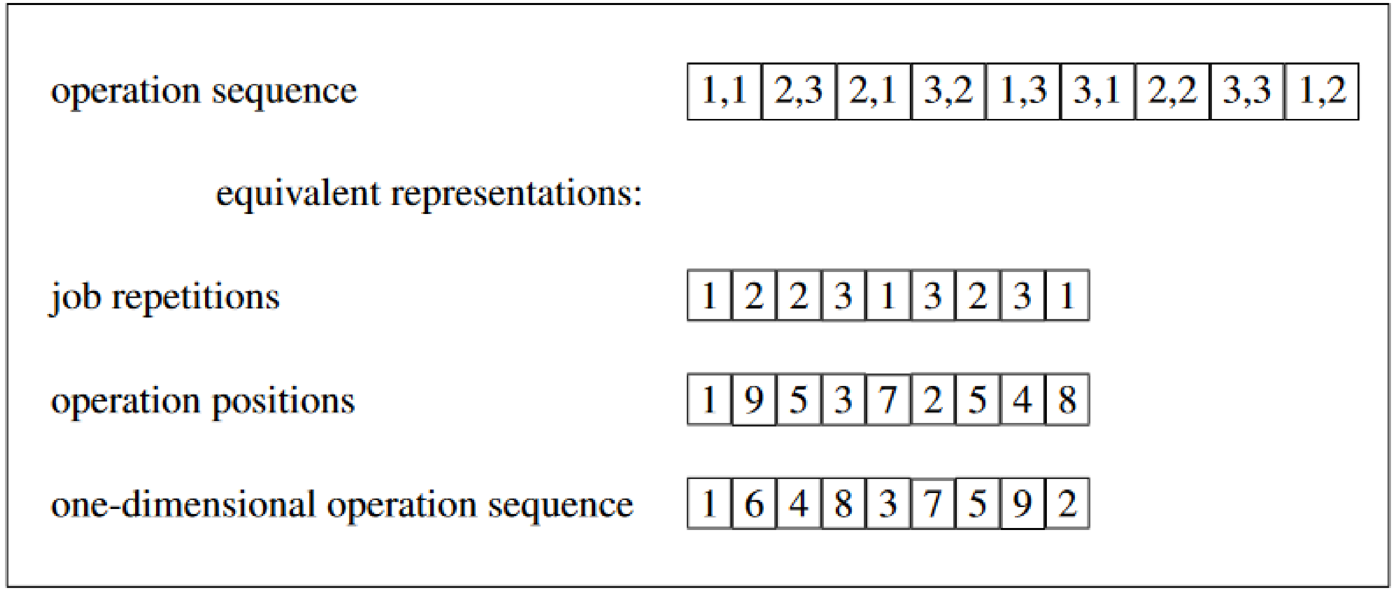

Figure 1. Operation-based representation

Operation-based

- Operation Positions
    - Subsequent numbering
        - (1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)
        - E.g. number 5 at gene 3 represents (1,3) at position 5

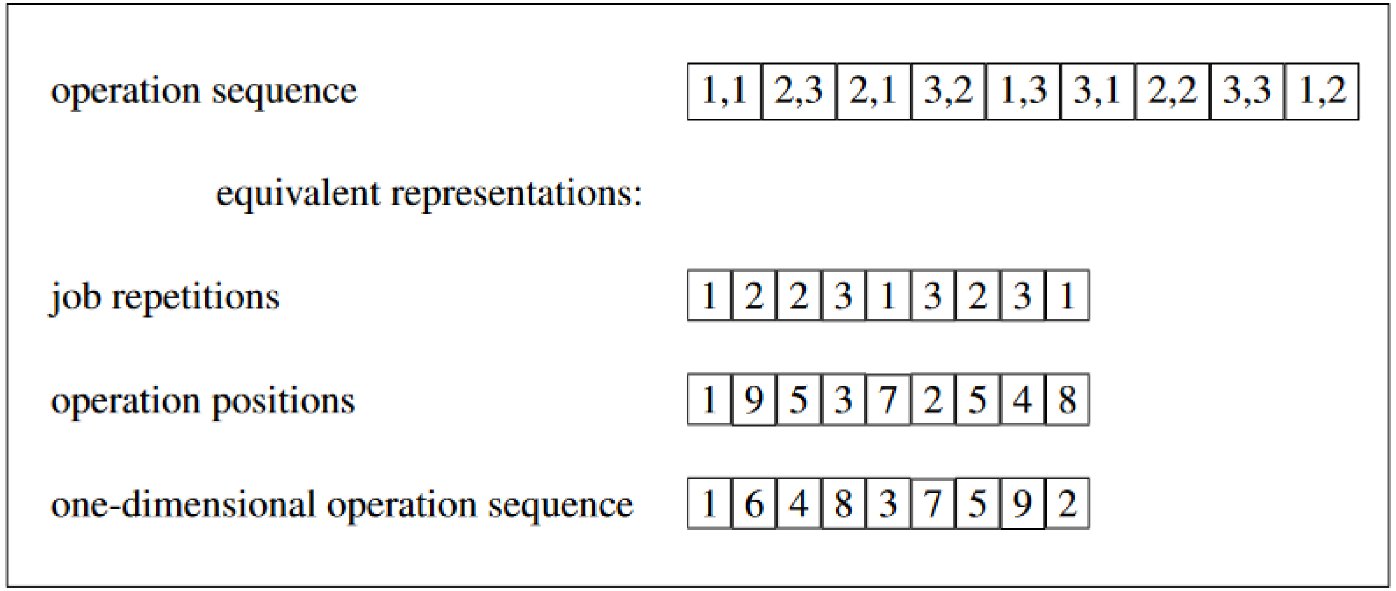

Figure 1. Operation-based representation

Algorithm

A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems

Takeshi Yamada and Ryohei Nakano
NTT Communication Science Laboratories, 2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-02, Japan

- GA-GT Algorithm
- Active schedules (parents) to create new job-schedules (offspring)
- Griffler & Thompson (GT) algorithm + uniform crossover
- Optimal solutions for 10 x 10 (n jobs x m machines)
- Good solutions for 20 x 20

Algorithm

Table 1
A 6×6 job-shop problem

Job	Operation routing (processing time)					
1	3(1)	1(3)	2(6)	4(7)	6(3)	5(6)
2	2(8)	3(5)	5(10)	6(10)	1(10)	4(4)
3	3(5)	4(4)	6(8)	1(9)	2(1)	5(7)
4	2(5)	1(5)	3(5)	4(3)	5(8)	6(9)
5	3(9)	2(3)	5(5)	6(4)	1(3)	4(1)
6	2(3)	4(3)	6(9)	1(10)	5(4)	3(1)

Reprinted from reference [11], pp. 226.

- Minimization Makespan
- The processing of job J_j
  On machine M_r
  Is the operation $O_{j, i, r}$
  Where i Є {1, …, m} is the position of operation in the technological sequence with processing time $p_{j, i, r}$
- The objective is to determing the set of completion times for each operation for each operation $c_{j, i, r}$ which minimizes makespan S_{max}

Algorithm

- Gantt-Chart

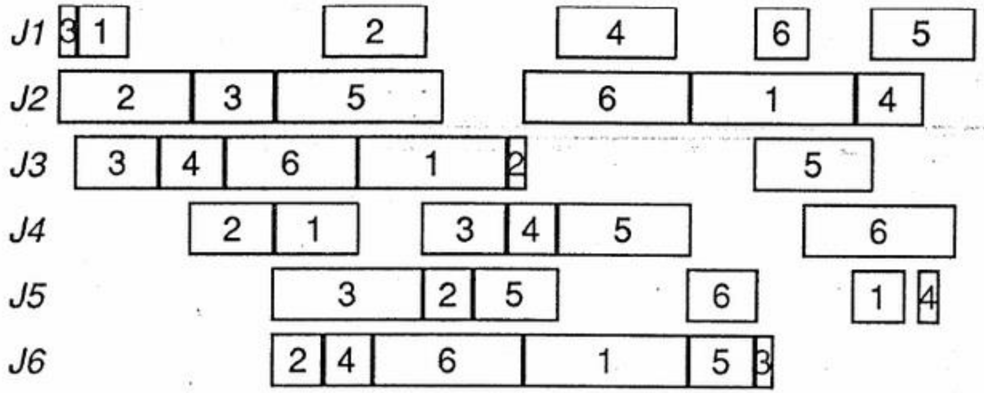

Figure 1. Gantt chart of a solution to the 6×6 job-shop problem

Table 2
Optimal solution to the 6×6 job-shop problem

Job	Completion times					
1	1	4	22	37	45	55
2	8	13	23	38	48	52
3	6	10	18	27	28	49
4	13	18	27	30	38	54
5	22	25	30	42	51	53
6	16	19	28	38	42	43

Algorithm

- Consider Temporal order
- A set *C* of the earliest operations in technological sequence among operations which are not yet scheduled is defined (cut)
- The earliest possible completion time $EC_{j,\ i}$ is calculated for each operation $O_{j,\ i} \in C$

(A1) Find O_{j^*,i^*,r^*} which has a minimum[2] EC in C: $EC_{j^*,i^*} = min\{EC_{j,i} \mid O_{j,i} \in C\}$

Specify G: a set of operations which consists of $O_{j,i,r^*} \in C$ sharing the same machine M_{r^*} with O_{j^*,i^*,r^*} and the processing of O_{j,i,r^*} and O_{j^*,i^*,r^*} overlaps. G is called a *conflict set*.

(A2) Choose one of the operations O_{j_s,i_s} from G, and schedule O_{j_s,i_s} according to EC_{j_s,i_s}.

(A3) Update C and ECs.

Repeat Step (A1) $\sim$ Step (A3), until all operations are scheduled, and then an active schedule is obtained.

In Step (A2), if all possible choices are considered, active schedules are generated for all[3] .

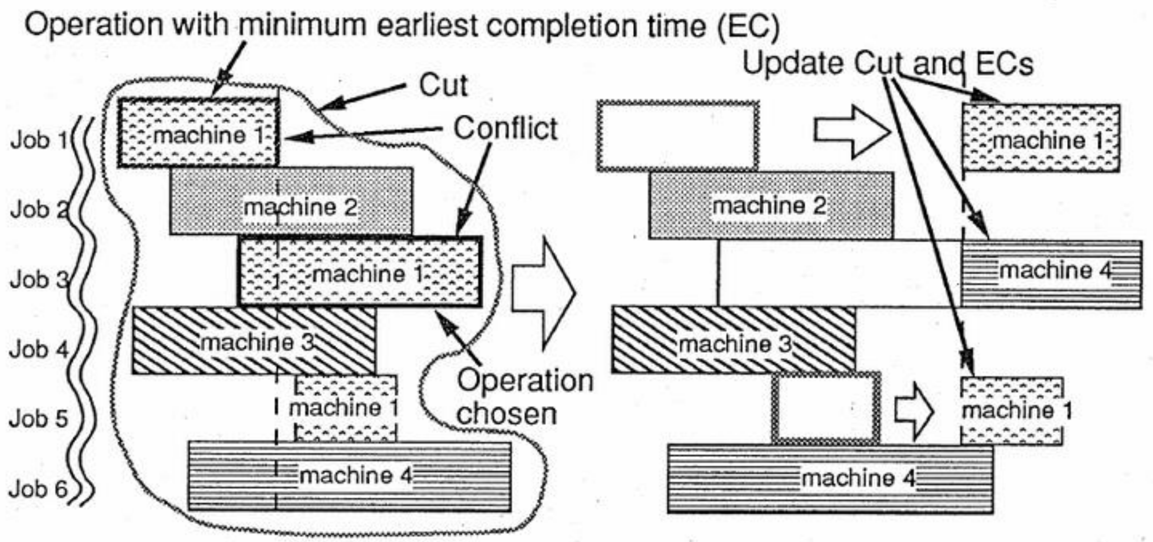

Figure 2. Giffler & Thompson's algorithm

Representation

- Each individual *psn* represents an active schedule directly using elements $\{psn_{j,\,i,\,r}\}$
  - $psn_{j,\,i,\,r} = c_{j,\,i,\,r}$

The makespan S_{psn} of the schedule represented by the individual psn is calculated as follows:

$$S_{psn} = max\{psn_{j,i} \mid 1 \leq j \leq n,\ i = m\} \qquad (1)$$

Crossover

(C1) Do Step (A1) of the GT algorithm, obtain C, ECs and G.

(C2) Choose one of the operations to be scheduled next from G as follows:

(a) generate a random number $\epsilon \in [0, 1)$ and compare it with $R_\mu \in [0, 1)$ which is a predefined constant called the *mutation rate*.
if $(\epsilon < R_\mu)$ ***then*** choose any operation O_{j_s,i_s} from G (mutation occurs).

(b) ***otherwise*** select either *mom* or *dad* with an equal probability 1/2.
Mom is assumed to be selected.
Find an operation O_{j_s,i_s} which was scheduled earliest in *mom* among all the operations in G:

$$mom_{j_s,i_s} = min\{mom_{j,i} \mid O_{j,i} \in G\} \tag{2}$$

(c) schedule O_{j_s,i_s} according to EC_{j_s,i_s}, then set $kid_{j_s,i_s} = EC_{j_s,i_s}$.

(C3) Update C and ECs.

By repeating Step (C1) ~ Step (C3) until all operations are scheduled, the new individual *kid* is obtained.

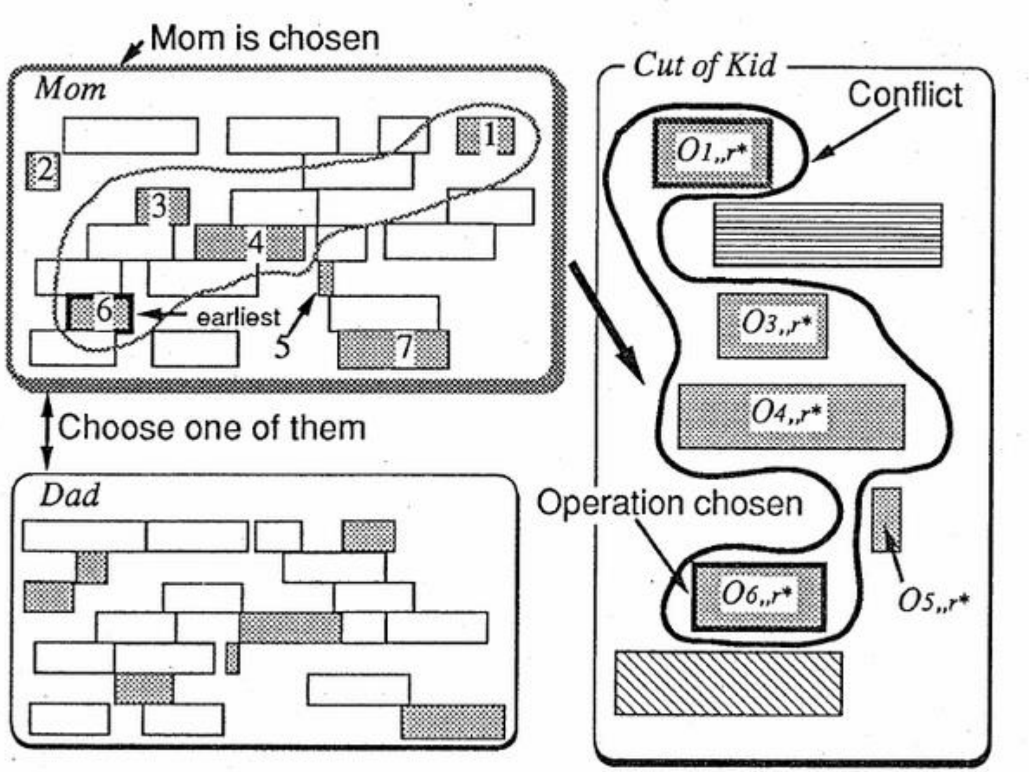

Figure 3. GA/GT crossover

Results

Table 3
Muth & Thompson's benchmark

Papers	6×6	10×10	20×5
Balas (1969)	55	1177	1231
McMahon (1975)	55	972	1165
Barker (1985)	55	960	1303
Adams (1988)	55	930	1178
Carlier (1989)	55	930	1165
Nakano (1991)	55	965	1215
Yamada (1992)	55	930	1184
Optimal	55	930	1165
Lower Bound	52	880	1164

Table 4
GA/GT v.s Random Samplings

20×20 problems	Random		GA/GT	
No.	Average	Best	Average	Best
1	1356.8	1126	979	967
2	1318.6	1104	953	945
3	1313.5	1107	957	951
4	1414.3	1202	1060	1052

Results

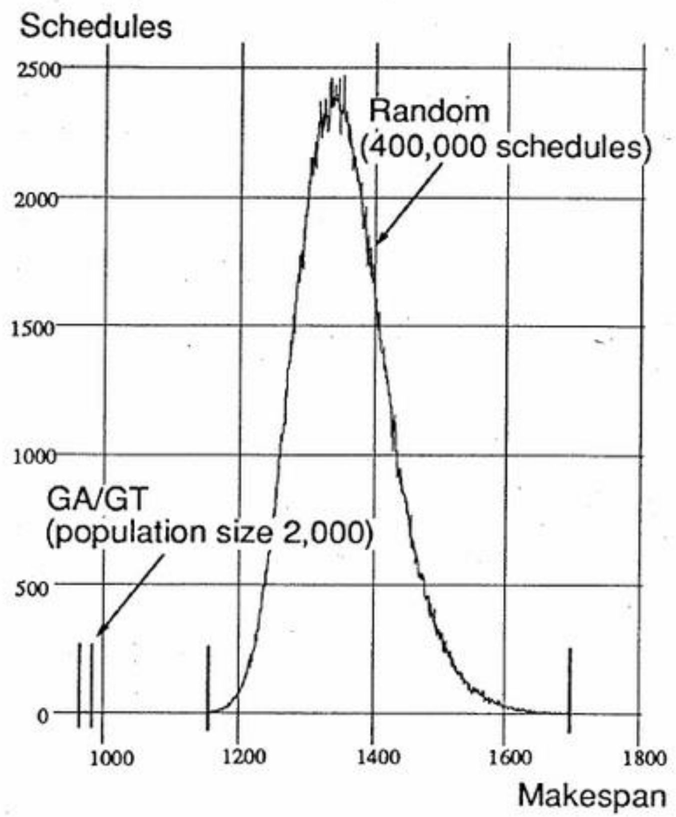

Figure 5. GA/GT v.s. Random Samplings

Limitations

- Missing code (complexity of Job-shop with constraints and feasibility)
  - Tried to compare many representations and algorithms setup by survey but spended too much time on researching it
- In survey, diversity-operators were not considered at all