# GA scheduling

From – Practical Issues and Recent Advances in Job- and Open-Shop Scheduling; Corne D., Ross P.



## What are scheduling problems

- Allocation of tasks to a limited set of resources such that the measure of schedule quality is optimized.
- Examples,
  - Time-tabling of {ambulance crews, lectures, etc}
  - Batch manufacturing of discrete products {consumer products, ship-building, cars, etc}

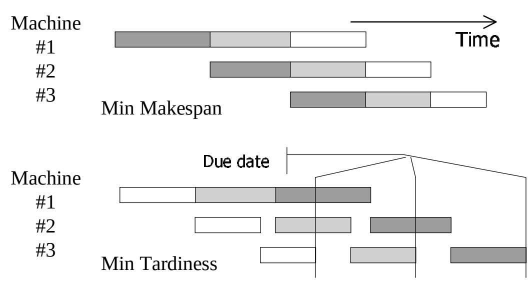


## Manufacturing case

- Receive orders in real-time or for the current production period
- Decompose the Bill of materials for each order.
- Defines minimum temporal requirements and quantities of raw materials.
- Objective
  - Minimal makespan; waste; tardiness.



## Two views of scheduling requirements



## Classical technique – backtracking

- Search algorithm suggests the ***sequence*** of variables to compose a solution.
- Each new variable adds a new search state, corresponding to a new more complete partial solution.
- Process repeats until a solution or dead-end state is encountered
  - DEAD-END states imply a state from which a solution cannot be reached without violating a constraint
  - BACK-TRACKING necessary to find an alternative allocation satisfying all the constraints



## Classical Solutions to Job-shop benchmarks – Makespan minimization

| Research | Algorithm | 10×10 | 20×5 |
|---|---|---|---|
| Balas (69) | B & B | 1177 | 1231 |
| McMahon (75) | B & B | 972 | 1165 |
| Baker (85) | B & B | 960 | 1303 |
| ABZ (88) | Bottleneck | **930** | 1178 |
| Carlier (89) | B & B | **930** | **1165** |



## GA Scheduling

- Problem definition
  - '*j*' jobs, '*m*' machines, where each job consists of an ordered sequence of tasks, '*t*'.
- Objective – minimize makespan to manufacture the jobs whilst satisfying delivery deadlines.
  - (Fig 1)



## Example Job-shop task definition

| Job | (m, t) | (m, t) | (m, t) | (m, t) | (m, t) | (m, t) |
|---|---|---|---|---|---|---|
| 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 |

## Direct Encoding

- Individual represents the schedule

  Job#1 Job#2 Job#3

  – {{20, 35, 59}, {60, 90, 172},{2, 75, 81},…}

  Task start times

  – Machine allocation performed by task matching
  – Very difficult to design crossover and mutation operators capable of preserving a feasible schedule.



## Permutation Based Encoding (PBE)

- Consider a 3 machine, 2 job problem – each job having 3 tasks
  – $J1_1$, $J1_2$, $J1_3$, $J2_1$, $J2_2$, $J2_3$
- Contents of an individual now simplifies to,
  – J1, J2, J2, J1, J1, J2
- Schedule builder interprets this as,
  – $J1_1$, $J2_1$, $J2_2$, $J1_2$, $J1_3$, $J2_3$



## Mutation for PBE

- Two possibilities
  – Random flipping of a job-task to a different job-task + swapping the partnering job-task to maintain a valid schedule.
  – Ignore the problem and deal with it at the schedule loading stage.



## Crossover and PBE – 1

| M | Parent #1 | | | | | Parent #2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| #1 | J1 | J2 | J4 | J3 | J5 | J3 | J5 | J4 | J2 | J1 |
| #2 | J3 | J1 | J2 | J5 | J4 | J4 | J3 | J5 | J1 | J2 |
| #3 | J5 | J4 | J1 | J2 | J3 | J1 | J2 | J3 | J5 | J4 |

Initial candidate parents

## Crossover and PBE – 2

| M | Child #1 | | | | | Child #2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| #1 | J1 | | J4 | | | | | J4 | | J1 |
| #2 | | J1 | | | J4 | J4 | | | J1 | |
| #3 | | J4 | J1 | | | J1 | | | | J4 |

Randomly select ‘m’ jobs which will remain fixed – J1, J4



## Crossover and PBE – 3

| M | Child #1 | | | | | Child #2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| #1 | J1 | J3 | J4 | J5 | J2 | J2 | J3 | J4 | J5 | J1 |
| #2 | J3 | J1 | J5 | J2 | J4 | J4 | J3 | J2 | J1 | J5 |
| #3 | J2 | J4 | J1 | J3 | J5 | J1 | J5 | J2 | J3 | J4 |

Fill in the remaining jobs in Child #1 (#2) using the “free jobs” from the rows of parent #2 (#1)



## Indirect Encoding

- Consider an encoding of the form,
  - a b c d e f … (where this is a sequence of integers)
- Interpreted as,
  - schedule the next task of the $a^{th}$ unfinished job
  - schedule the next task of the $b^{th}$ unfinished job
  - Etc
- Unfinished jobs are summarized in a cyclic list,
  - E.g. $8^{th}$ unfinished job in the list J1, J5, J9 is J5



## GA solutions to Job-shop benchmarks – Makespan minimization

| Research | 10×10 | 20×5 |
|---|---|---|
| Nakano (91) | 965 | 1215 |
| Yamada (92) | 930 | 1184 |
| Croce (92) | 946 | 1178 |
| Dorndorf (93) | 938 | 1178 |
| Atlan (93) | 943 | 1182 |
| Fang (94) | 939 | **1165** |
| Mattfeld (94) | **930** | **1165** |
| Ono (96) | **930** | **1165** |

## Indirect Encoding on Benchmark Problems – Average Tardiness

| Problem | EMOD | SPT | EOD | MST | GA |
|---|---|---|---|---|---|
| #1 | 16 | 16 | 16 | 20 | 13** |
| #2 | 26 | 28 | 26 | 29 | 24** |
| #7 | 18 | 19 | 22 | 16* | 17 |
| #10 | 25 | 30 | 31 | 46 | 24** |
| #12 | 25 | 31 | 32 | 35 | 25* |



## Summary

- By concentrating on the representation issues, the operation of the search operators is simplified.
- In comparison with deterministic methods, GA method is both computationally faster and capable of finding equally 'optimal' solutions.



## References

– Corne D., Ross P. "Practical Issues and Recent Advances in Job- and Open-Shop Scheduling," in Evolutionary Algorithms in Engineering Applications – Dasgupta D., Michalewicz Z. (eds), Springer-Verlag, 1997.
– Fang H.-L., Ross P. , Corne D., "A promising Hybrid GA/ Heuristic Approach for Open Shop Scheduling Problems," ICAI'94, pp 590-594.