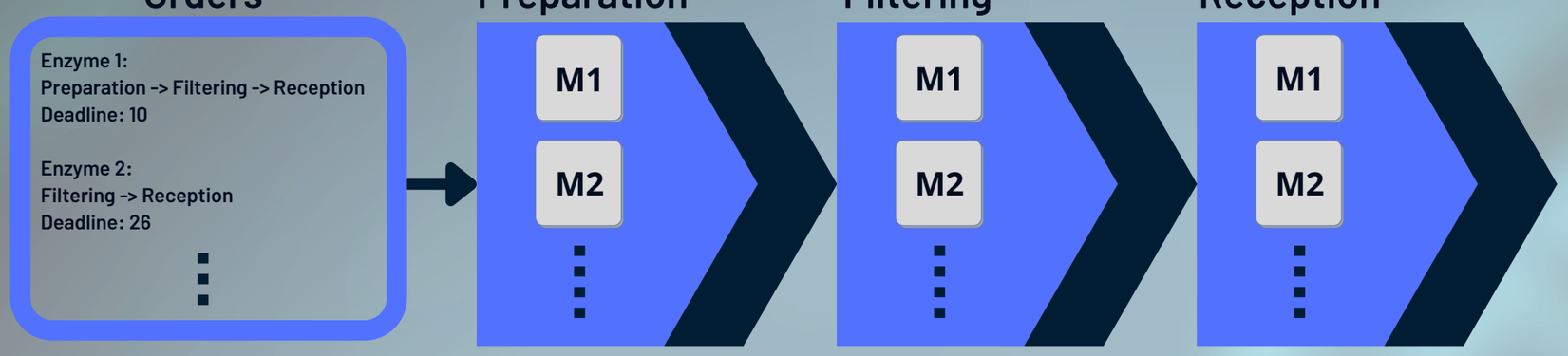
Suitability of Genetic Algorithms for solving Flexible Job Shop Problems

Author: Marko Ivanov ‖ Supervisors: Dr. Mathijs de Weerdt, Kim van den Houten

TU Delft

1. Problem background



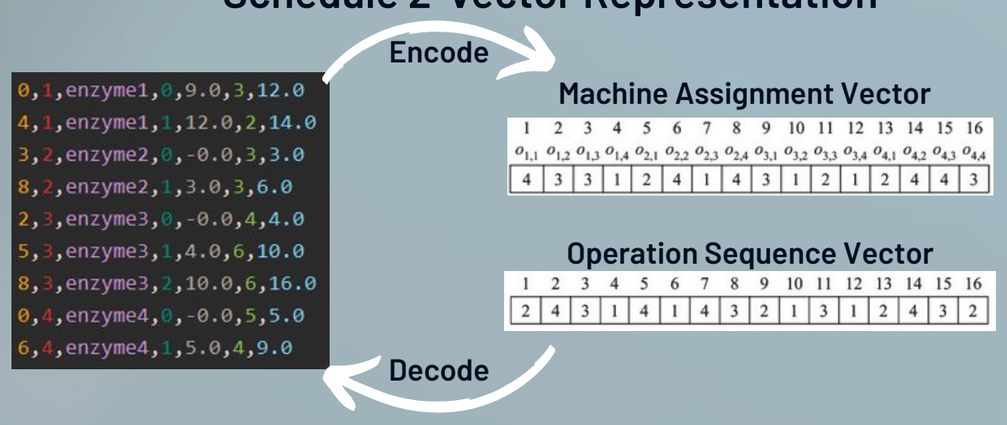
This can be modelled as a Flexible Job Shop Problem!
How do we assign the operations to the available machines?

Main research question

Is a Genetic Algorithm suited to find satisfactory solutions to the given Flexible Job Shop Problem (FJSP) instances in a reasonable amount of time compared to the provided (Mixed Integer Linear Programming) MILP implementation?

2. Methodology

Schedule 2-vector Representation



Single- and Multi-objective Genetic Algorithm

Parameters: Population Size, Generation count, Fitness Function, Mutation coefficient

Generate a starting population of schedule representations

For each generation:

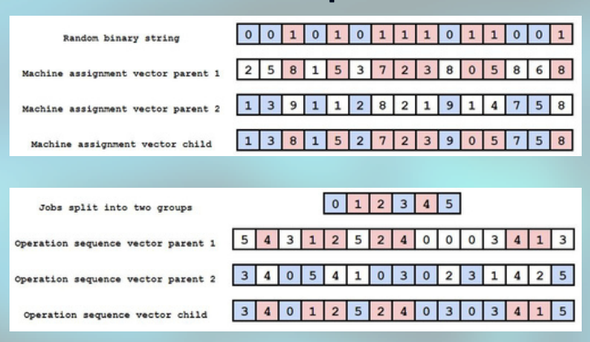
- Optional: Select the fittest individuals to keep as elites. Elites are kept unchanged in the next generation.
- Use selection to choose parents
- Split the parents into pairs and apply crossover
- Apply mutation to children and non-elite members of the population
- Add elite members to the next generation's population
- Fill the rest of the population size with the fittest individuals from the mutated non-elites and children

Repeat either until we have reached the maximum number of generations or a set time limit

Mutation Operators

- For the machine assignment vector swap a single machine assignment with another valid machine for the operation
- For the operation sequences - chose an operation and insert it at another random place in the operation sequence vector

Crossover Operators



Selection Operators

Single objective GA

Roulette wheel selection is used.

- Rank individuals according to makespan.
- Construct probability vector.
- Draw random individual from the population according to the probability distribution described by the vector until as many parents as the population size are chosen.
- Split parents into pairs and apply crossover.

To determine which individuals are chosen for the next generation, the elites, non-elites and children are ranked based on their makespan. The top |*population_size*| schedules are chosen for the next generation.

Multi-objective GA

Tournament selection is used.

- Draw ten individuals from the population at random.
- Chose one with the best fitness to be a parent.
- Repeated until we have chosen as many parents as the population size.
- Split parents into pairs and apply crossover.

After the children are generated, non-dominated sorting is applied to a joint population of the elites, non-elites and children. As with the single objective case, the top |*population_size*| schedules are chosen for the next generation.

Note on non-dominated sorting

Domination condition

$$\forall i : O_i^A \geq O_i^B$$

$$\exists i : O_i^A > O_i^B$$

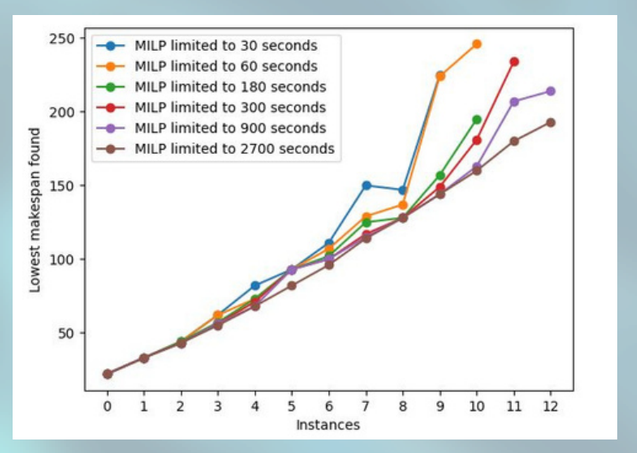
IFF these conditions are met, scedhule A dominates schedule B

Two objectives are used: makespan and lateness.
Makespan is the latest operation completion.
Lateness is the total delay after the deadline across all jobs.
Schedules in the population are split into frontiers. The optimal frontier contains the schedules which are not dominated by any other schedule. All other frontiers contain contain schedules which are dominated only by those in previous frontiers. The algorithm by which this ranking is performed is presented in [5] and its runtime is O(mn^2).

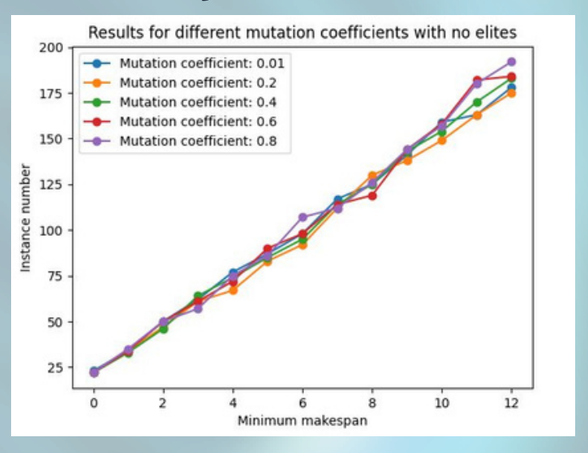
3. Results

The performance of the algorithms is measured on the 13 available problem instances. The instances increase in complexity.

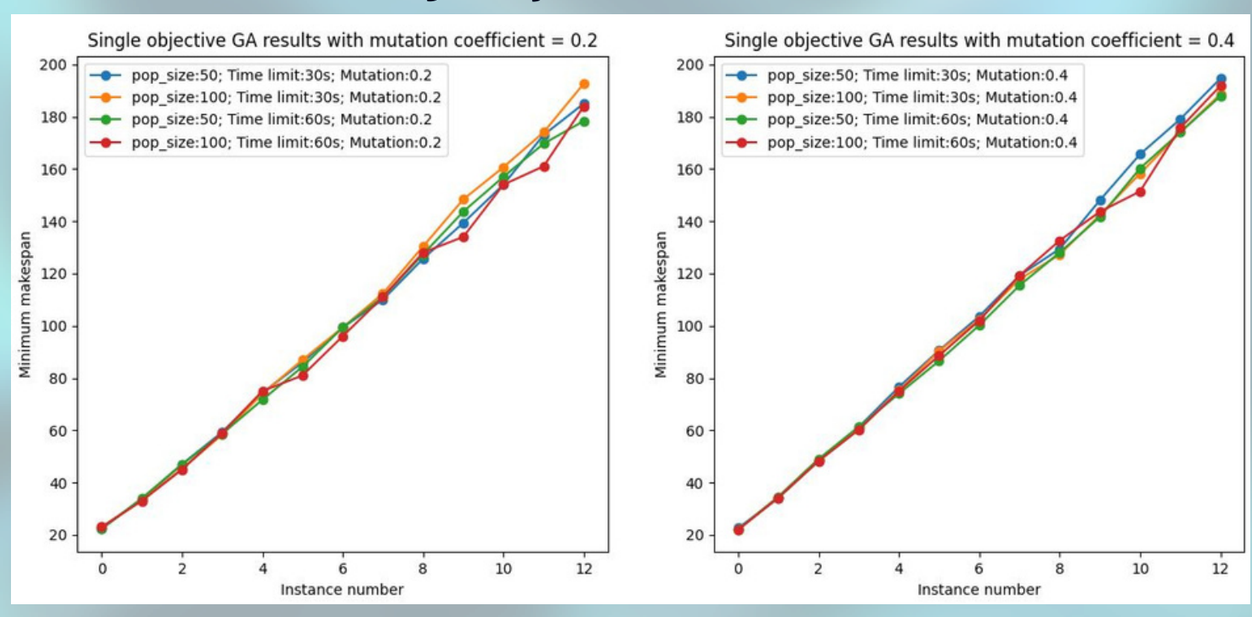
MILP results



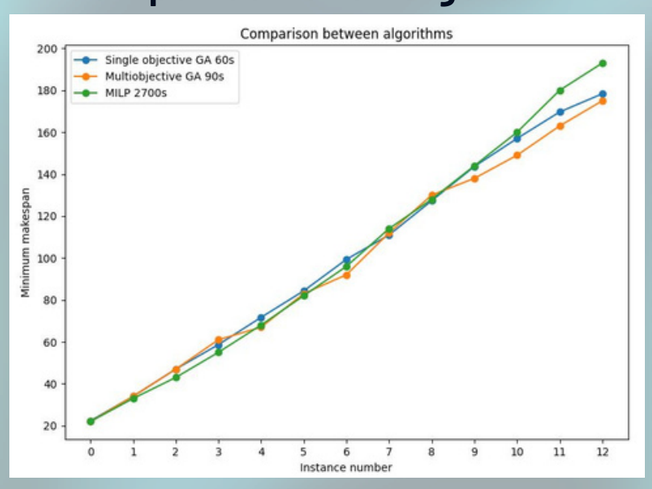
Multi-objective GA results



Single objective GA results



Comparison of all algorithms



4. Conclusions

Both genetic algorithms can outperform the MILP implementation on larger problem instances. If given enough time and computing power a MILP implementation will usually reach a solution close the optimal. However, when it comes to practical applications, usually a solution which is good enough and is found in a reasonable amount of time is sufficient. The two presented genetic algorithms showcase that in just a few minutes, they can produce better results for large problem instances.

Strengths of Genetic Algorithm approach:

- Produce multiple feasible solutions.
- Incorporate multiple objective functions.
- Reach good solutions faster than MILP implementation for larger instances.

Weaknesses of Genetic Algorithm approach:

- Difficulty in setting parameters.
- Rarely reaches the optimal solutions.
- Premature convergence.
- Randomness in results.

5. Future work

- Test the algorithms on problem instances which are used as benchmarks in the literature.
- Use problem specific traits as objectives. The unique part about the DSM problems is the presence of cleaning times. A possible objective function is the minimisation of total cleaning time or maximum machine cleaning time.
- Test the multi-objective approach on more than two objectives, as well with objectives which might be conflicting with one another. This would really add to the ability to generalise the results for more practical applications, where flexibility is needed.

References:

[1] M. Gen, Y. Tsujimura, and E. Kubota. Solving job-shop scheduling problems by genetic algorithm. 2:1577–1582 vol.2, 1994.
[2] Jie Gao, Linyan Sun, and Mitsuo Gen. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers amp;Operations Research, 35(9):2892–2907, 2008.
[3] Xiaojuan Wang, Liang Gao, Chaoyong Zhang, and Xinyu Shao. A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. The International Journal of Advanced Manufacturing Technology, 51(5-8):757–767, 2010.
[4] I. Kacem, S. Hammadi, and P. Borne. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 32(1):1–13, 2002.
[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii, 2002.