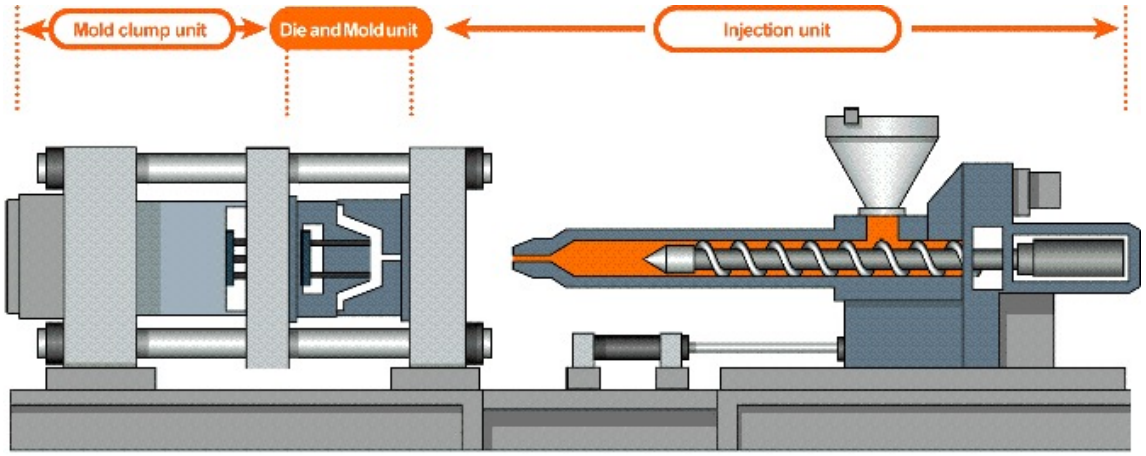
The Mold Shop Problem

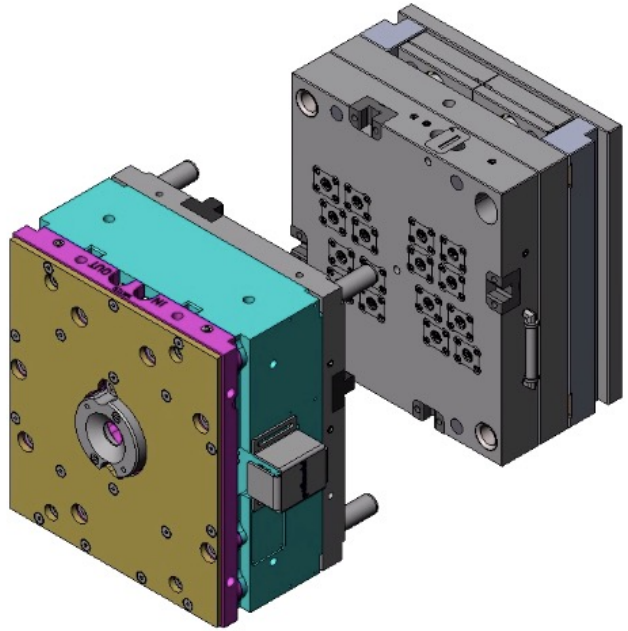
Automating, Optimizing, and Reacting to the Domain-Specific Challenges of Scheduling Injection Molding Jobs

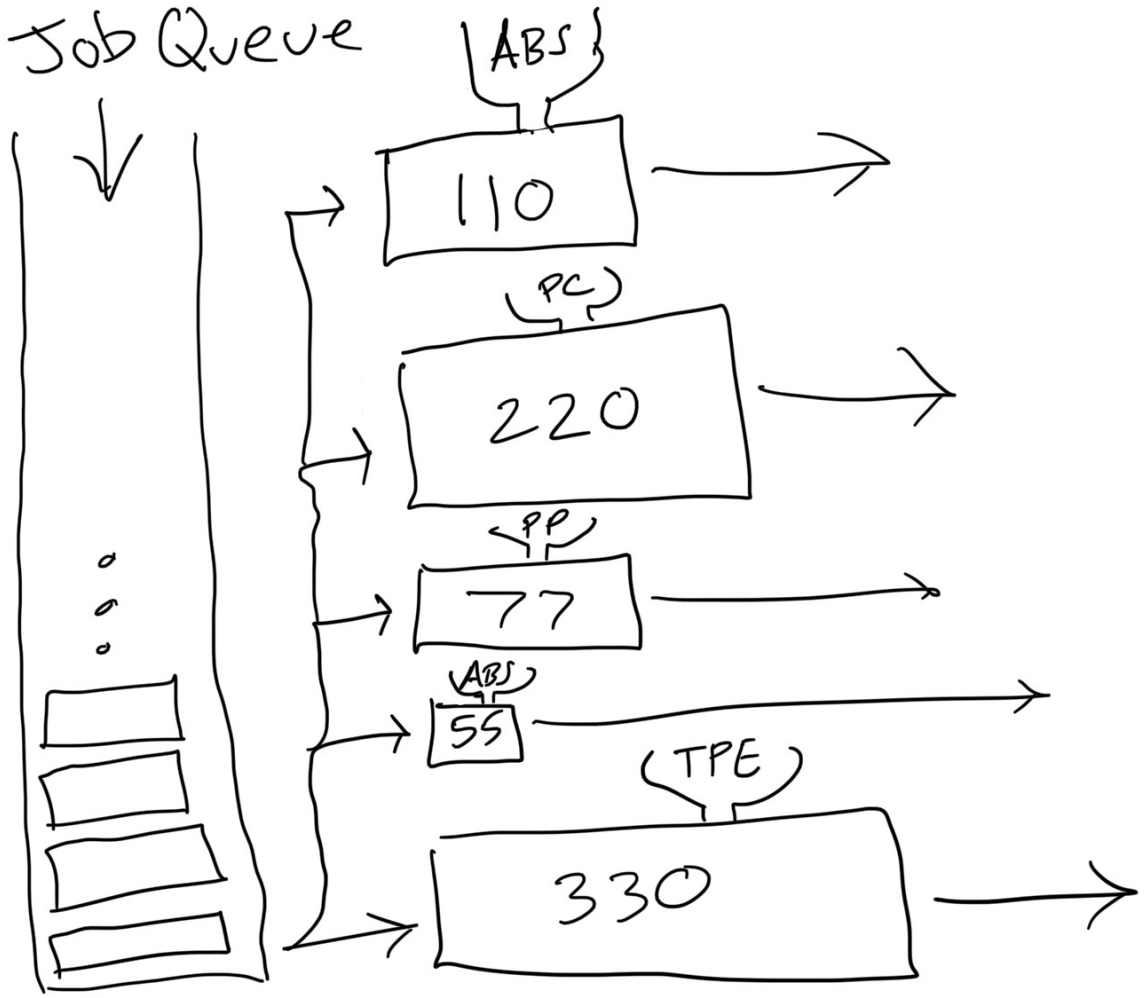
MSCS
Ryan Palo
4/28/2021
Lewis University
A Master's Thesis Presentation

How It Works

- Specific part shape, requires a custom-made mold to create.
- Different size/style molds take different amounts of time to set up and tear down
- # **cavities** = # pieces per cycle
- Mold design also affects **cycle time**
    - how long to complete one part
    - typically < 1 minute






Job Queue
ABS
110
PC
220
PP
77
ABS
55
TPE
330

Defining a Job

- Which mold tool?
- Which machines can it run on?
- Cycle time
- How many cavities?
- A specific material grade
- Tool setup and teardown time
- Quantity needed
- Due date

The General Problem: The Job Shop Problem

N jobs M machines O operations

- Goal: Optimize some fitness function
  - Minimize makespan (total time to finish all jobs)
  - Minimize lateness (completion of jobs after their due dates)
  - Maximize throughput (Flow Shop)
  - Minimize changeover
  - Minimize downtime (keep machines busy all the time)
  - Minimize earliness (lean manufacturing, no stock held in inventory)

Getting Domain-Specific: The Mold Shop Problem

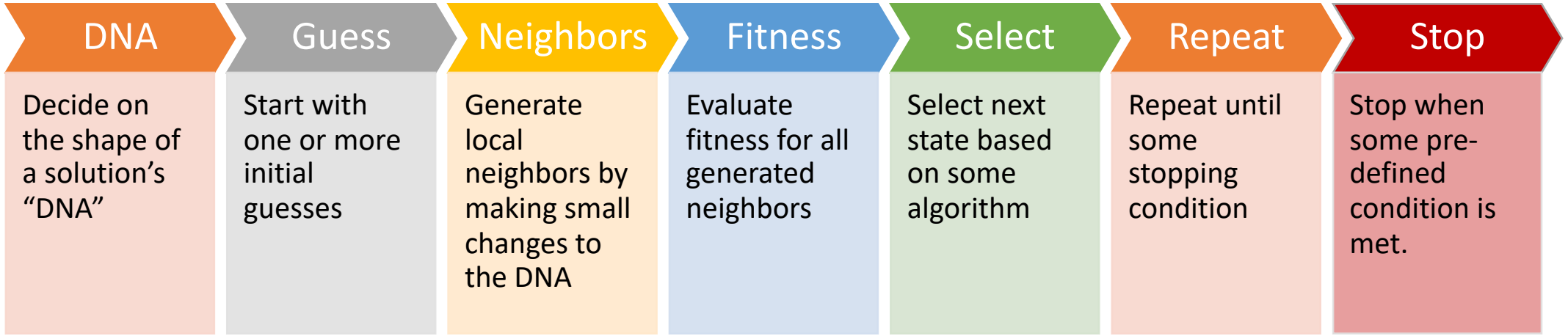
N jobs M similar machines 1 operation

- Goal: Optimize fitness function
  - Minimize lateness (each day late on each job summed up)
  - Earliness not considered a problem
  - Minimize mold changeover: changing molds incurs teardown/setup time
  - Minimize material changeover: changing materials incurs *h* hours of delay purging the machine and loading new material
- Introduction of the “Flexible JSP”: multiple machines could possibly run a particular job
  - Actually a *second searching problem!*

Managing the Giant State Space

- Enumerating all solutions is unfeasible
- Problem is NP-Hard
- Solution:
  - Domain-specific relaxations
  - Genetic-type algorithm
  - Local search
  - (Machine learning)

Local Search: Overview



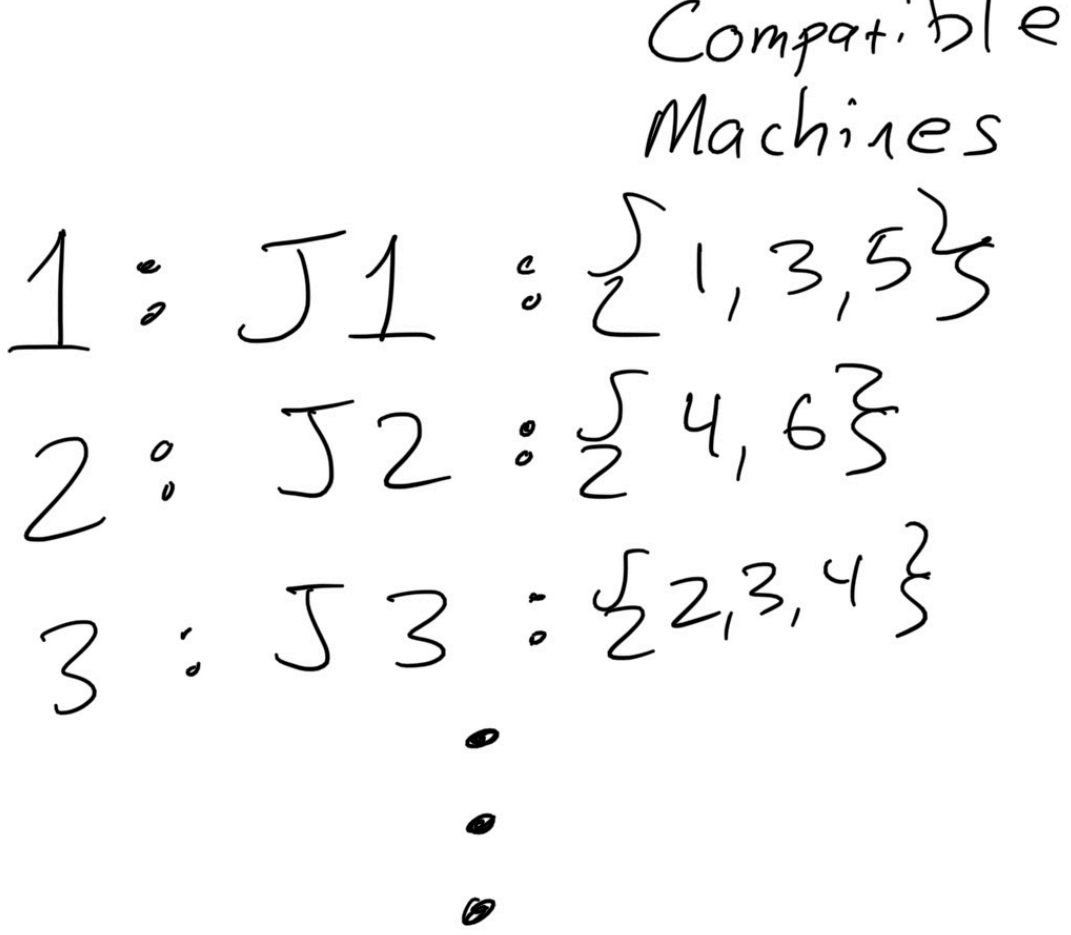
A Rich Ecosystem of Algorithms

- **Stochastic Descent:** Simplest. Randomly generate one neighbor. Choose if fitness is better.
- **Simulated Annealing:** Randomly generate one neighbor. If fitness is better, choose. Otherwise, choose anyway with the probability related to a continuously decreasing (cooling) “temperature” factor.
- **Tabu Search:** Generate a bunch of neighbors and select the best one if it’s better than the current solution. Keep a list of forbidden states based on previously visited states or other rules.
- **Particle Swarm Optimization:** Have many concurrent greedy searches all driven by both a local inertia as well as a pull towards the current global best (“hive mind”)
- **Branch and Bound:** Generate a small amount of neighbors. If stuck, generate more neighbors. Sort of a locally bounded exhaustive search.

My Solution

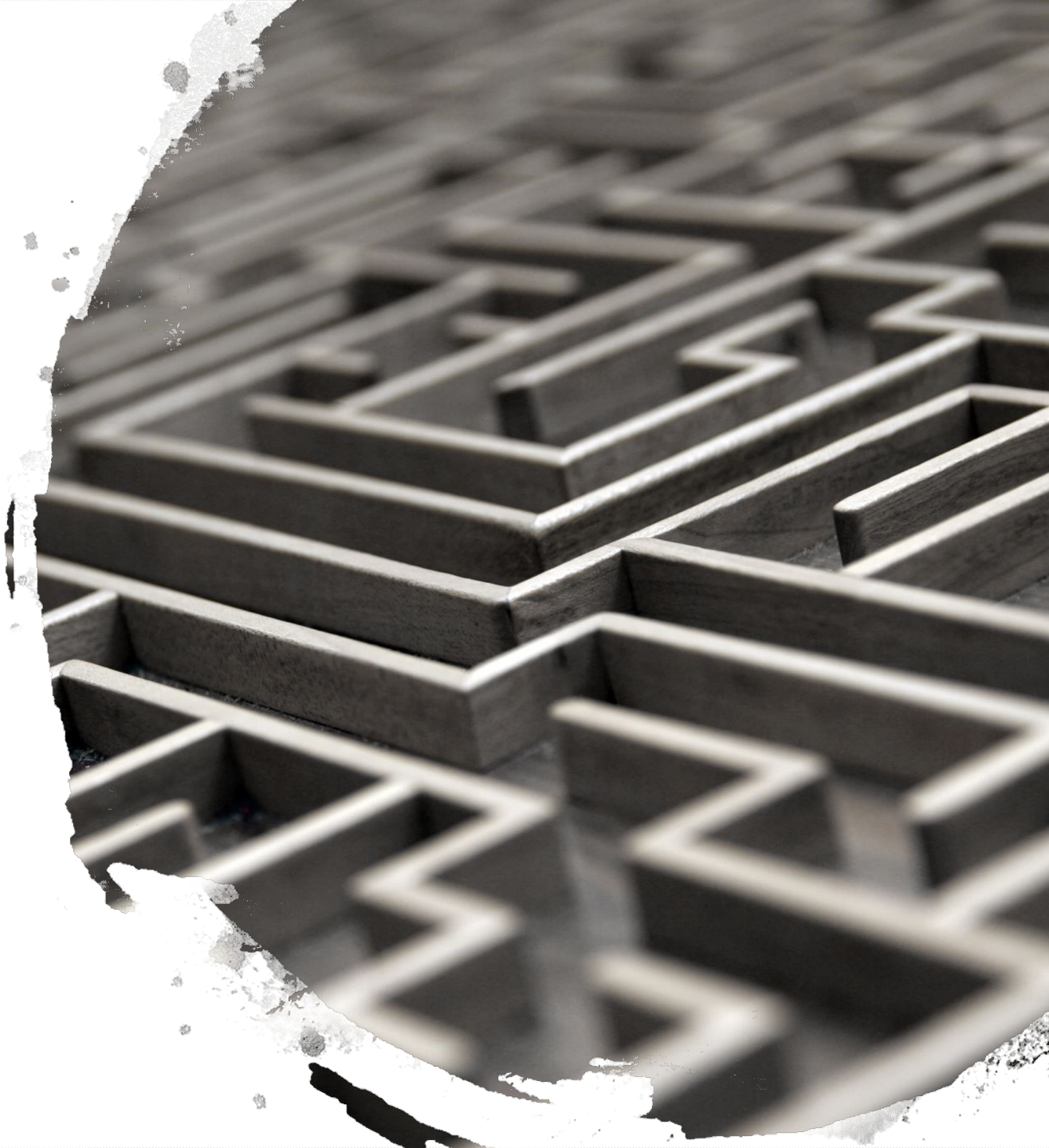
Solution Model

- A simple list of jobs in the order that they should be assigned



Overall Shape of Approach

1. Generate an initial solution.
2. Iterate some local search algorithm on a DNA strand solution.
3. At every iteration, perform a greedy list-to-machine assignment schedule in order to evaluate fitness.
4. Use this fitness to select the next neighbor state in the local search.



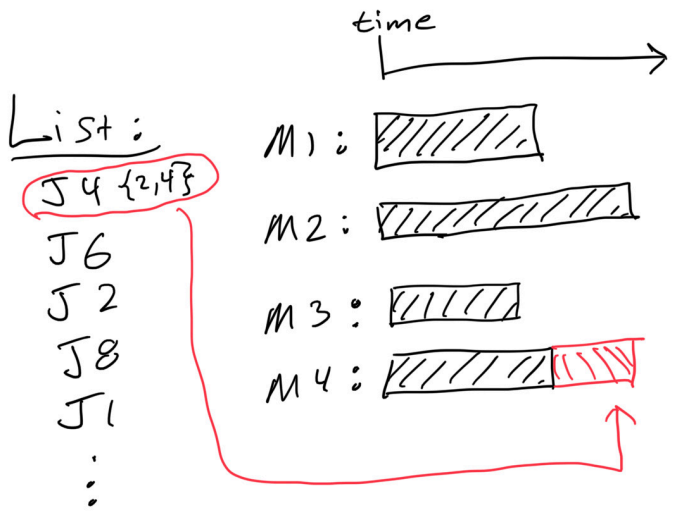
Generate Initial Solution

Generate Randomly

Generate in sorted order

Sort by due date (sooner first) and then by duration (shorter first)

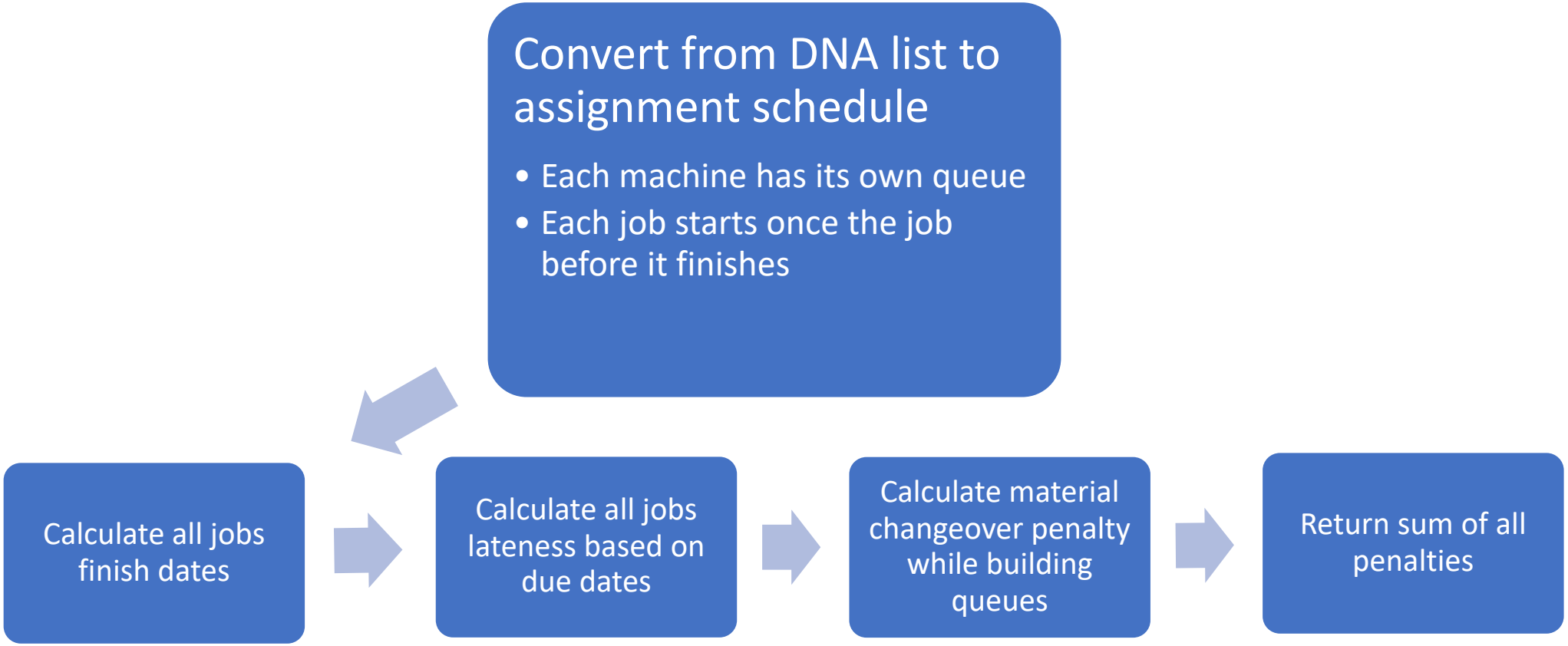
Likely not far from optimal already



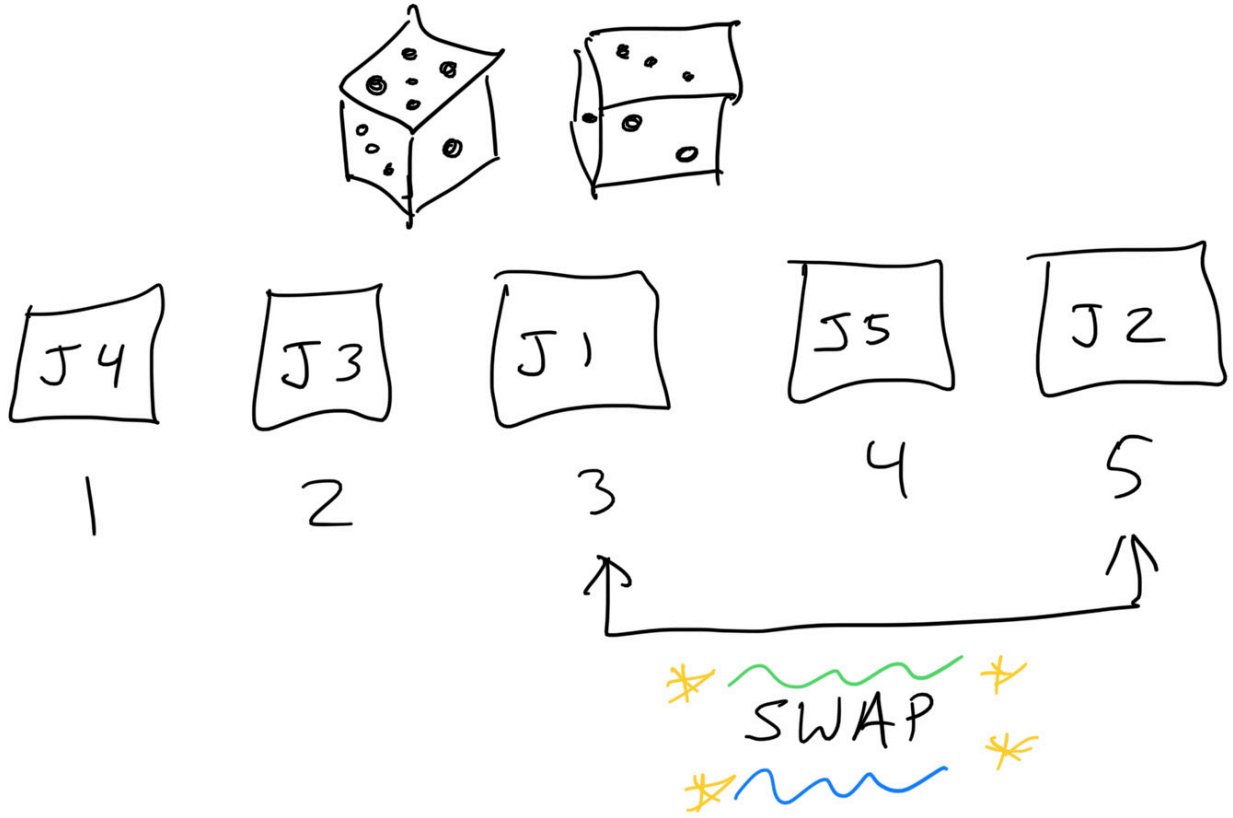
Conversion from List to Schedule

- Simple, greedy algorithm
- Traverse list linearly, assigning each job to the first available, compatible machine.

Calculating Fitness



Generating Neighbor States



Swap two random jobs in the list.

Selecting the Next State

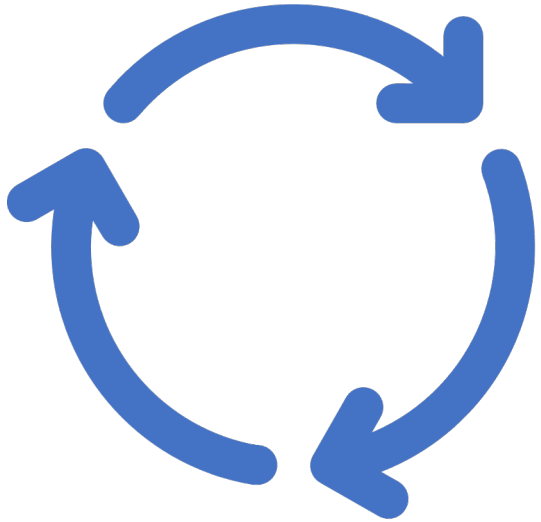
Stochastic Descent

Only select if neighbor is better

Simulated Annealing

Select if neighbor is better or with temperature-driven probability otherwise

Lots of other algorithms to try.
PSO is interesting because it has more than one ”searcher.”
Branch & bound is interesting because it generates more than one neighbor at a time.



Stopping Condition

- Fixed # of iterations
- ~~Once fitness stops changing~~
- ~~Fancier algorithm~~

Testing: Four Test Scenarios

56 unique jobs, with various materials, molds, quantities, and due dates. Derived from a real mold shop schedule from early 2021.

Baseline	• Satisfying all deadlines tight but possible
Heavy	• All deadlines tightened up from baseline. Some jobs will be late.
Late	• Deadlines tightened up even more, some jobs already late as of Day 0.
Light	• All deadlines loosened from baseline. Easy to finish all jobs on time. Goal is to optimize changeovers.

Results

Method	Metric	Baseline Demand	Heavy Demand	Light Demand	Late Demand
Manual	Fitness	11.5	55.49	8	270.71
	Duration (s)	2692.18	1352.78	1085.91	1217.26
Stochastic Descent	Fitness	11.54	19.96	11.5	160.81
	Duration	1.297	1.366	1.382	1.281
Simulated Annealing	Fitness	11.27	19.03	9.5	159.92
	Duration	1.304	1.257	1.404	1.190

Success! Comparable performance with 1000x time reduction!

What's Next?

- More accurate model
- Trying more different local search algorithms
  - Each with special parameters
- UI
- Online adjustments: modifying an existing schedule to react to problems
- Machine learning: risk, setup/teardown from mold to mold

References

Caballero-Villalobos, J. P.-D.-C. (2013). Scheduling of complex manufacturing systems with Petri nets and genetic algorithms: a case on plastic injection moulds. *International Journal of Advanced Manufacturing Technology, 69*(9-12), 2773 - 2786.

Gökan Maya*, B. S. (2015). Multi-objective genetic algorithm for energy-efficient job shop scheduling. *International Journal of Production Research, 53*(23), 7071–708.

Ghiath Al Aqel, X. L. (2019, 3 12). A Modifed Iterated Greedy Algorithm for Flexible Job Shop Scheduling Problem. *Chinese Journal of Mechanical Engineering, 32*(1), 21-32.

Hernández-Ramírez, L. a.-V.-B.-V.-R. (2019, 1). A Hybrid Simulated Annealing for Job Shop Scheduling Problem. *International Journal of Combinatorial Optimization Problems and Informatics, 10*(1), 6-15.

Hoos, H. H. (2005). *Stochastic Local Search : Foundations & Applications.* San Fransisco, CA, USA: Morgan Kaufmann.

Kim, M.-W. P.-D. (1998). A systematic procedure for setting parameters in simulated annealing algorithms. *Computers Ops Res.*, 207-217.

Klement, N. a. (2021). Lot-Sizing and Scheduling for the Plastic Injection Molding Industry—A Hybrid Optimization Approach. *Applied Sciences, 11*(3).

Ladhari, M. H. (2003). A Branch-and-Bound-Based Local Search Method for the Flow Shop Problem. *The Journal of the Operational Research Society, 43*(10), 1076-1084.

Li, Y. W. (2020). An improved simulated annealing algorithm based on residual network for permutation flow shop scheduling. *Complex Intell. Syst*, 1-11.

Lorenz, R. H. (2016). RNA folding with hard and soft constraints. *Algorithms for Molecular Biology, 11*(8), 1-13.

Meeran, S. a. (2012, 8). A hybrid genetic tabu search algorithm for solving job shop scheduling problems: a case study. *Journal of Intelligent Manufacturing, 23*(4), 1063-1078.

References (cont'd)

Nießе, A. S. (2016). Local Soft Constraints in Distributed Energy Scheduling. *Proceedings of the Federated Conference on Computer Science and Information Systems, 8*, 1517–1525.

Rahmati, S. a. (2012). A new biogeography-based optimization (BBO) algorithm for the flexible job shop scheduling problem. *International Journal of Advanced Manufacturing Technology, 58*(9-12), 1115-1129.

Stawowy, A. a. (2017). Coordinated Production Planning and Scheduling Problem in a Foundry. *Archives of Foundry Engineering, 17*(3), 133-138.

Trilling, L. A. (2012). Anesthesiology Nurse Scheduling using Particle Swarm Optimization. *International Journal of Computational Intelligence Systems, 5*(1), 111-125.

Vázquez-rodríguez, J. A. (2010, 12). A new dispatching rule based genetic algorithm for the multi-objective job shop problem. *Journal of Heuristics, 16*(6), 771-793.

Wirsansky, E. (2020). *Hands-On Genetic Algorithms with Python.* Birmingham : Packt Publishing.

Wongwiwat, A. a. (2013). Production scheduling for injection molding manufacture using Petri Net model. *Assembly Automation, 33*(3), 282-293.

Xingjuan Cai, Z. C. (2009). Individual Parameter Selection Strategy for Particle Swarm Optimization. In A. L. (Ed.), *Particle Swarm Optimization* (pp. 89-112). Rijeka: InTech.

Yang, X.-S. (2014). *Nature-Inspired Optimization Algorithms.* London: Elsevier.

Questions?