National Technical University of Athens
School of Electrical and Computer Engineering
MSc Data Science & Machine Learning

Scheduling unrelated machines with job splitting, setup resources and sequence dependency

Apostolos Moustakis (03400182)
Student of DSML

The Main Concept of The Topic

- Assign jobs to multiple machines in order for them to be processed in parallel (Parallel Machine Scheduling - PMS)
- When jobs are processed in parallel in multiple machines the total execution time is reduced
- The goal is the minimization of makespan, which is the total time for all the jobs to finish

Analysis of Each Term

- **Unrelated Machines:** A job can have different processing time in different machines, which is realistic
- **Job Splitting:** Each job can be split in different parts and each part can be processed in different machines in parallel
- **Setup Resources:** Setup is the time needed between the execution of two jobs in the same machine. The number of setups that can be performed at the same time is restricted
- **Sequence Dependency:** The setup time of a job i that follows job j is different from the setup time of the job j that follows job i. The setup times are also different in different machines

An Example of the Studied Problem

Processing times (p_{im})

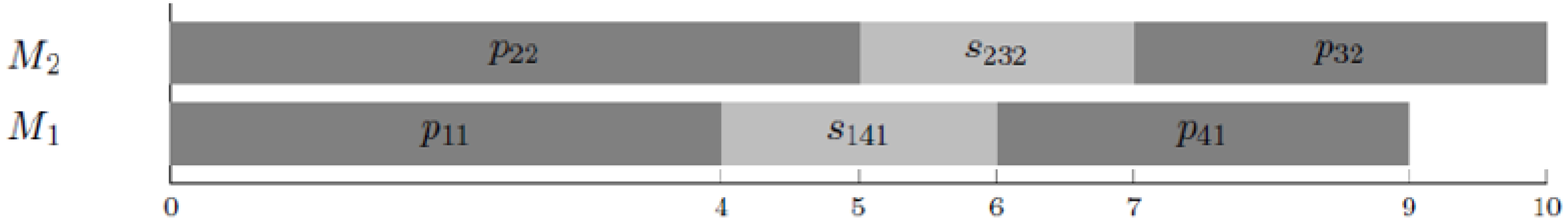
J \ M	1	2
1	4	7
2	5	5
3	5	3
4	3	4

Setup times (s_{ijm})

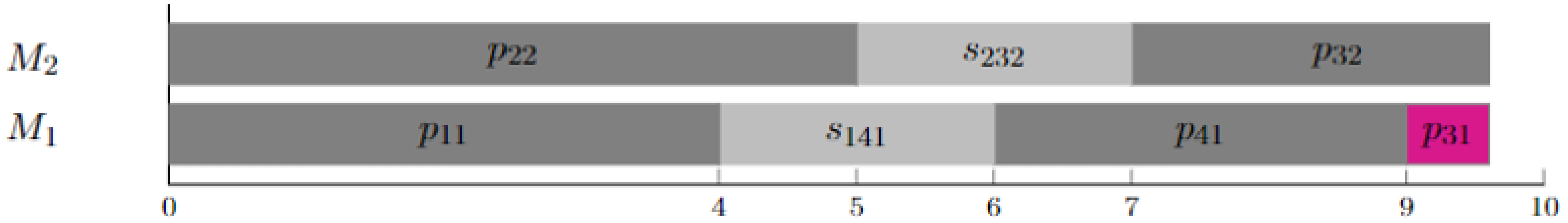
J \ J	1	2	3	4
1	0	1	1	2
2	1	0	2	2
3	2	2	0	1
4	1	1	0	0

J \ J	1	2	3	4
1	0	1	3	3
2	2	0	2	1
3	0	1	0	0
4	0	1	0	0

Gantt chart of optimal solution (**<u>without splitting</u>**)



Gantt chart of optimal solution (**<u>with splitting</u>**)



The Main Idea for Solving the Problem

Heuristic Algorithm

Fast & Efficient

- Warm up Solution
- Tight upper bound

Logic Based Benders Decomposition

Exact Method for Optimality

Model Parameteres and Variables

Model parameters	Description
J	The set of jobs – including a dummy job 0
J^*	The set of jobs – excluding the dummy job 0
M	The set of machines
p_{im}	The processing time of $i \in J$ on $m \in M$
s_{ijm}	The setup time of $j \in J$ succeeding job $i \in J - \{j\}$ on $m \in M$
$s_0 \in \mathbb{R}$	The setup time of the first job, $\forall m \in M$
R	A setup resource constraint to indicate that, at each time interval, at most R machines can be set up in parallel
δ	A positive constant for (MIP 1), with $\delta \to \infty$

Model variables	Description
$z_{i,m}$	1 if $i \in J^*$ is the first job to be assigned on machine $m \in M$, 0 otherwise
$f_{i,m}$	the percentage of the first $i \in J^*$ to be processed on machine $m \in M$
$y_{i,m}$	1 if $i \in J^*$ is assigned on machine $m \in M$, 0 otherwise
x_{ijm}	1 if $j \in J$ succeeds $i \in J^* - \{j\}$ on machine $m \in M$, 0 otherwise
w_{im}	the percentage of job $i \in J^*$ to be processed on machine $m \in M$
$W_{im} \in \mathbb{Z}^+$	the integer percentage of job $i \in J^*$ to be processed on machine $m \in M$
$n_{im} \in \mathbb{Z}^+$	auxiliary integer variables that ensure the feasibility of sequencing
$C_{max} \in \mathbb{R}^+$	the makespan of the schedule
$\mu_{im} \in \mathbb{R}^+$	interval variables that indicate the time interval of setup of job $i \in J^*$ machine $m \in M$

Lower Bounds

MIP1: The objective of MIP1 is C_{max} subject to the constraints:

$$y_{i,m} \geq w_{im} \; \forall i \in J^*, m \in M$$
$$z_{i,m} \geq f_{im} \; \forall i \in J^*, m \in M$$
$$z_{i,m} + y_{i,m} \leq 1 \; \forall i \in J^*, m \in M$$
$$z_{i,m} - f_{i,m} \leq 1 - \delta \; \forall i \in J^*, m \in M$$
$$\sum_{m \in M} z_{i,m} + f_{i,m} = 1 \; \forall i \in J^*$$
$$\sum_{i \in J^*} z_{i,m} \leq 1 \; \forall m \in M$$
$$\sum_{i \in J^*} z_{i,m} \geq \sum_{i \in J} y_{i,m} \; \forall m \in M$$
$$\sum_{j \in J^*} (f_{j,m} \dot{p}_{j,m} + s_0 \dot{z}_{j,m} + w_{j,m} \dot{p}_{j,m} + y_{j,m} min_{i \in J^* - \{j\}} s_{i,j,m} \leq C_{max} \; \forall m \in M$$
$$y_{i,m}, z_{i.m} \in \{0,1\}, f_{i,m}, w_{i.m} \in [0,1], C_{max} \in \mathbb{R}^+, \forall i \in J^*, m \in M$$

MIP2: The objective of MIP2 is minimizing C_{max} subject to the constraints:

$$\sum_{i \in J^*} w_{i,m} \dot{p}_{i,m} \leq C_{max} \; \forall m \in M, \quad \sum_{m \in M} w_{i,m} = 1 \; \forall i \in J^*$$

MIP3: The objective of MIP3 is minimizing C_{max} subject to the constraints:

$$\sum_{j \in J^*} (w_{j,m} \dot{p}_{j,m} + y_{j,m} min_{i \in J^* - \{j\}} s_{i,j,m} \leq C_{max} \; \forall m \in M, \quad \sum_{m \in M} w_{i,m} = 1 \; \forall i \in J^*$$

Greedy Heuristic Algorithm (GHA)

Algorithm 1 GHA: A three-stage greedy heuristic

max_assgn $\leftarrow |J||M|$, $C_{max} \rightarrow \infty$
while max_assgn $\geq |J|$ **do**
  Solve (MIP 1) or (MIP 2) or (MIP 3) and let y be the number of job parts assigned over all machines
  max_assgn $\leftarrow y$
  Run Sequencing Stage
  Run Resource Management Stage
  Let ldm be the load of each machine $m \in M$
  If necessary update C_{max} with $max_{m \in M}\{ldm\}$
  max_assgn $\leftarrow$ max_assgn 1
end while
Return C_{max}

3 Stages: Assignment, Sequencing and Resource Management

Logic Based Benders Decomposition (LBBD)

Benders Decomposition is a primal-dual method that can provide an exact solution to the problem

The main components of the method are:

- The Master Problem M that provides lower bounds
- The Subproblem S that provides upper bounds
- A set of optimality and feasibility cuts that pass from S to M

The idea is to move difficult restrictions to S after solving M easily (a relaxed version of the problem), which is simple. Logic Based refers to integer subproblems

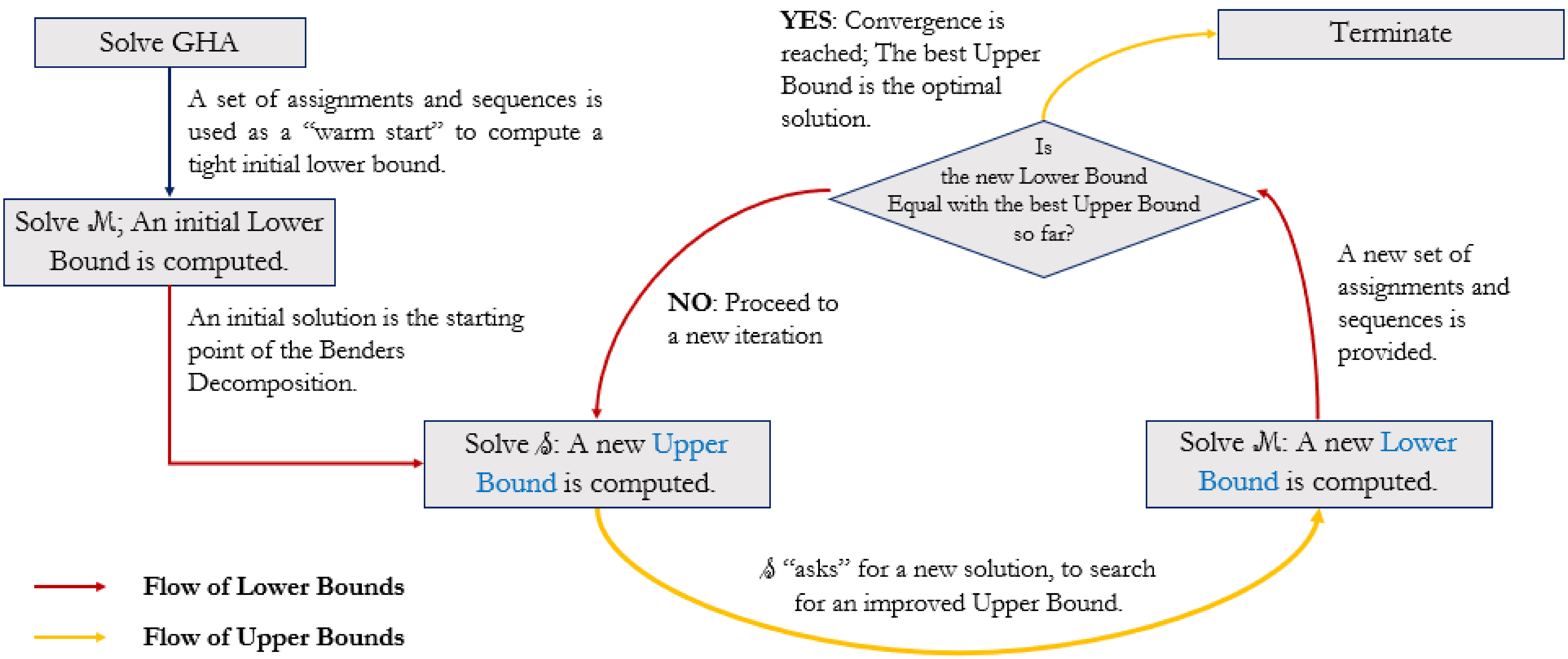
LBBD in this Specific Problem

Master Problem M: Assignment and Sequencing Stage
All jobs are split into parts, which are assigned and sequenced to machines. Assumption that the resources are unlimited. The complexity is significantly reduced

Subproblem S: Resource Management Stage
Considering fixed sequences of parts of jobs to machines, as provi-ded by the solution of M, the subproblem adds intervals of idle time between jobs, so that the available resources are not exceeded

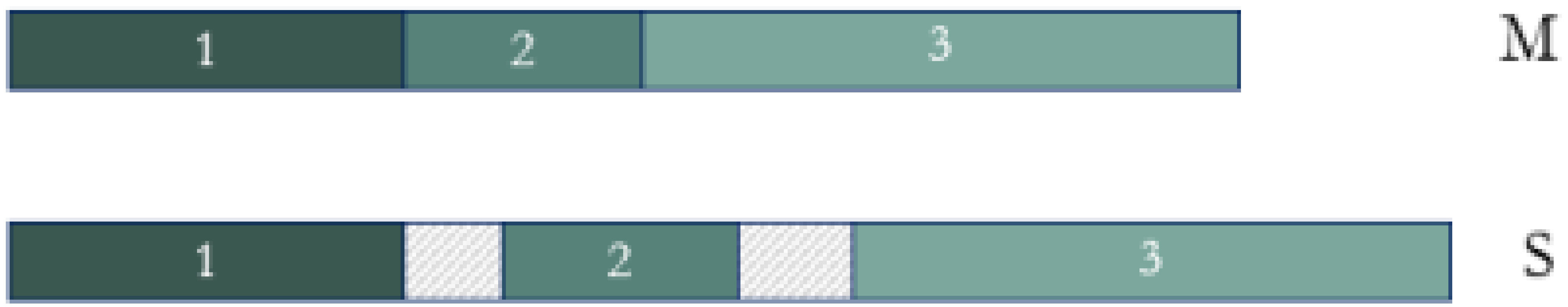
The solution of GHA provides a tight upper bound for M, which is used as a "warm start"

A Flowchart of LBBD

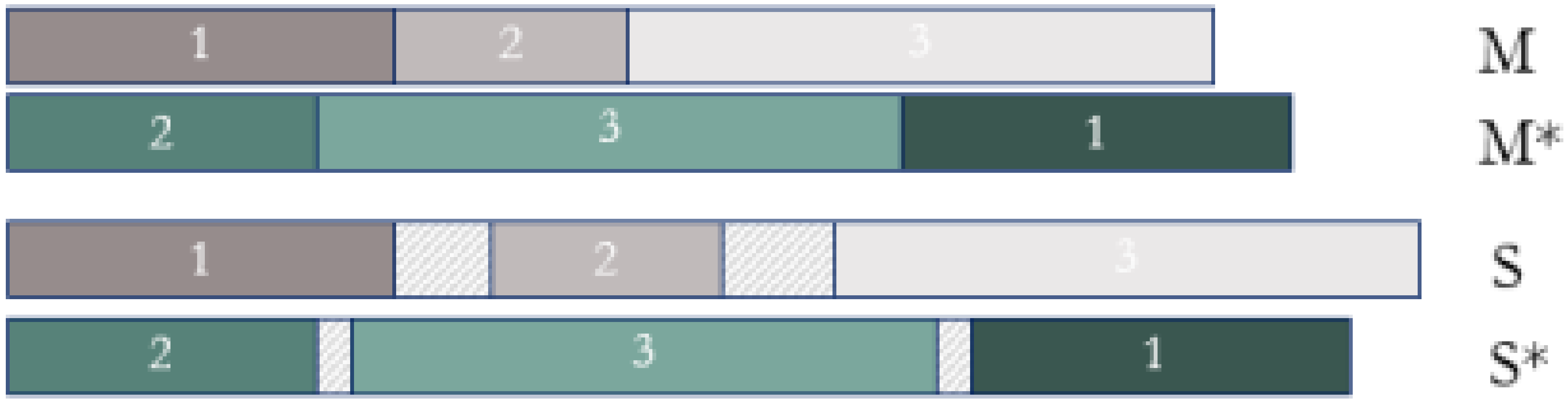


Simple LBBD Example

- Initial Lower Bound of M and Upper Bound of S



- Iterations between M and S



S* is the Optimal Solution

Implementation: Benchmark Instances

$|M| \in \{2, 5, 10, 15, 20\}$

$|J| \in \{10, 20, 30, 40, 50, 80, 100, 200, 300, 400, 500, 700, 1000\}$

$p_{im} = b_i a_{im} + u_{im}$ u.a.r from [1,10]

$s_{ijm} = p_{jm} a_{ijm}$ $a_{ijm} \in \{[0.01, 0.1], [0.1, 0.2], [0.1, 0.5]\}$

$R \in \{1, 3, 5\}$

Total number of experiments: 70 * 3 * 3 = 630

Gap is equal to: $\frac{Solution - Lower\ Bound}{Lower\ Bound}$

GHA Benchmark Results

		Machines											
		2		5		10		15		20		Mean	
	Instance	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
J	10	8.48	< 1	13.26	< 1	12.19	3	18.69	8	15.84	18	13.69	6
	20	7.19	< 1	10.06	1	21.97	52	24	73	33.52	136	19.35	52
	30	6.69	< 1	13.34	2	31.06	150	34.36	137	42.74	347	25.64	127
	40	5.64	< 1	10.8	2	25.13	107	38.23	141	51.56	272	26.27	104
	50	5.98	< 1	10.27	3	22.41	148	41.08	26	57.16	270	27.38	90
	80	6.07	2	7.51	3	23.51	10	41.29	29	59.57	164	27.59	42
	100	6.03	3	7.27	4	19.49	14	40.19	25	55.24	57	25.64	21
	150	5.95	6	7.52	8	17.97	22	33.57	43	51.94	78	23.39	31
	200	5.77	12	7.58	16	16.29	24	34.01	60	50.49	108	22.83	44
	300	5.81	47	8.1	33	14.97	79	30	95	42.57	161	20.29	83
	400	6.03	106	7.49	77	13.63	147	28.91	217	41.75	329	19.56	175
	500	6.1	210	7.54	158	13.38	191	29.25	330	42.86	482	19.83	274
	700	16.28	428	7.58	355	12.74	455	29.04	520	39.79	639	21.09	479
	1000	–	–	7.66	610	13.82	849	29.06	868	39.16	1152	22.43	870
α	[0.01,0.1]	1.05	59	2.24	81	3.62	165	4.39	179	6.83	307	3.63	161
	[0.1,0.2]	8.84	62	10.94	87	20.62	168	36.54	197	48.96	288	25.18	158
	[0.1,0.5]	11.35	67	13.81	105	31.16	150	55.86	175	77.96	307	38.03	161
R	1	7.85	63	13.21	91	38.28	161	72.8	184	99.1	301	46.25	160
	3	6.69	63	6.87	91	8.74	160	13.22	184	21.75	300	11.46	160
	5	6.69	63	6.87	91	8.39	161	10.78	184	12.91	301	9.14	160
	Mean	7.08	63	8.98	91	18.47	161	32.27	184	44.59	301	22.28	160

LBBD Benchmark Results

		Machines																	
		2			5			10			15			20			Mean		
	Instance	Gap	Time	GHA	Gap	Time	GHA	Gap	Time	GHA	Gap	Time	GHA	Gap	Time	GHA	Gap	Time	GHA
J	10	0.89	< 1	4.64	1.23	5	10.65	3.82	1488	11.00	6.30	3507	17.02	10.86	6026	14.99	4.62	2205	11.66
	20	0.51	< 1	2.41	1.98	11	6.57	3.96	5755	18.25	9.87	6940	20.59	10.94	7338	31.48	5.45	4009	15.86
	30	0.39	1	2.20	1.73	83	8.71	5.16	4430	26.25	10.29	7035	30.20	10.63	7781	39.39	5.64	3866	21.35
	40	0.57	< 1	1.04	2.13	260	6.31	5.72	5113	20.62	11.47	6809	32.57	15.42	7065	47.20	7.06	3850	21.55
	50	0.49	9	0.98	1.48	3479	5.84	3.81	8740	17.86	9.47	7537	36.30	19.69	7274	54.54	6.99	5408	22.90
	80	0.29	6	1.26	1.39	232	3.06	2.87	7545	18.06	9.35	8019	35.50	18.98	8221	53.82	6.58	4804	22.34
	100	0.39	11	0.91	1.52	196	3.10	2.64	6879	14.64	7.03	6578	34.33	17.57	7952	48.99	5.83	4323	20.39
	150	0.40	41	0.83	1.27	334	3.09	2.58	7836	13.99	5.93	8541	27.90	13.67	10109	46.17	4.77	5372	18.20
	200	0.46	392	0.58	1.26	2645	2.85	2.33	8679	11.58	5.47	8925	28.64	17.95	7955	44.28	5.49	5719	17.59
	300	0.44	597	0.56	1.18	5312	2.79	7.40	8185	14.97	–	–	–	–	–	–	3.01	4698	6.10
	400	0.43	559	0.66	–	–	–	–	–	–	–	–	–	–	–	–	0.43	559	0.66
	500	0.44	689	0.54	–	–	–	–	–	–	–	–	–	–	–	–	0.44	689	0.54
	700	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	1000	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
α	[0.01, 0.1]	0.88	170	0.33	2.53	3029	2.85	4.19	7944	4.34	6.15	7424	7.16	4.92	5891	6.99	3.56	4758	3.55
	[0.1, 0.2]	0.23	193	1.19	1.59	1279	6.58	4.03	6299	17.98	6.46	6949	28.82	12.32	7068	41.69	4.21	4117	17.68
	[0.1, 0.5]	0.32	214	2.63	1.62	642	11.12	5.90	4532	28.64	14.30	6885	51.26	20.06	6239	57.39	8.55	3913	31.49
R	1	0.58	196	1.81	3.60	2165	20.60	11.99	7324	49.59	29.93	8157	89.69	25.46	5827	67.64	10.97	4740	37.55
	3	0.42	185	1.17	1.32	1142	4.70	3.14	6168	10.09	4.44	7523	20.79	4.35	5945	18.48	2.70	4165	8.79
	5	0.42	196	1.17	1.32	1141	4.35	2.94	5680	8.23	4.27	6531	12.49	5.67	7598	13.70	2.65	3883	6.38
	Mean	0.48	192	1.38	2.46	2267	11.24	6.78	6667	24.51	14.73	7260	44.23	15.08	7747	42.21	5.44	4262	17.57

Next Step: Simulated Annealing

- Simulated Annealing: A metaheuristic optimization algorithm used to find the global optimum of a function. The algorithm's name comes from annealing in metallurgy
- Better solutions always get accepted (Lower Energy)
- Worse solutions are accepted if the probability p is greater than a pseudorandom number x generated from the uniform distribution in the interval [0, 1]

$$p = exp(-\frac{E_{proposed} - E_c}{T_c})$$

Algorithm 2 Pseudocode of Simulated Annealing

Require: define DECREASE(T), define PERTURB(x)

```
Begin
T ← initial_temp
x ← x_c
while T > Final_T do
    Equilibrium ← False
    while Equilibrium == False do
        x_proposed ← PERTURB(x)
        if E_proposed < E_c then
            x ← x_proposed
        else if exp(−(E_proposed − E_c)/T_c) > Random(0, 1) then
            x ← x_proposed
        else if then
            x ← x
        end if
        step ← step + 1
        T ← DECREASE(T)
    end while
end while
End
```

Initial Temperature and Solution

Find a new pertu-rbative solution

By decreasing T the algorithm becomes greedy

Simulated Annealing in this problem

- The goal is to improve the solutions provided by the GHA. The "Energy" is the solution's makespan

- Temperature: Tc, Tcry, q
  - The temperature is decreased regarding the cooling factor q ($Tc = q \cdot Tc$). Many experiments are performed

- Perturbative Solutions
  - Interior Swap: Randomly chooses 1 machine and 2 jobs in that machine and swaps them
  - Exterior Swap: Randomly chooses 2 machines and 1 job in each machine and swaps them

Brief explanation of the code

- main: reads the instances and solutions of the GHA, performs experiments, calls the function simulated_annealing repeatedly, saves the results in a csv file
  - Note: each experiment is performed 5 times

- sol_calc: Each time calculates the solution, returns tables with the solution (jobs as rows, initial machine load, setup, assignment and final machine load as columns) and the makespan

- simulated annealing: performs interior and exterior swaps, tries to find better solutions, returns the solution found: execution time, gap and objective

Results

Machines	Jobs	Workers	Alpha	GHA_Gap	GHA_Obj	SA_Gap	SA_Obj	SA_time
2	10	5	0	15.669	148.38	10.282	141.47	0.599
2	10	5	2	8.149	132.06	8.133	132.04	0.269
5	10	5	0	8.972	37.05	8.913	37.03	0.369
5	10	5	1	10.29	44.85	7.36	43.66	0.494
5	10	5	2	13.506	55.4	11.663	54.5	0.305
2	20	5	0	11.622	320.79	10.060	316.30	1.480
2	20	5	2	7.962	241.11	7.811	240.77	0.450
5	20	5	0	10.758	90.42	9.840	89.67	2.044
2	30	5	0	10.427	552.03	9.531	547.55	0.685
2	30	5	2	7.478	416.06	7.254	415.19	1.987
5	30	5	2	9.64	125.09	9.482	124.91	1.600
5	40	5	0	12.407	183.63	11.899	182.80	0.447
5	40	5	2	7.615	169.26	7.323	168.80	3.419
2	50	5	2	9.356	695.12	9.193	694.08	0.963
5	50	5	0	12.636	210.73	12.150	209.82	3.373
5	50	5	2	8.501	229.15	8.108	228.32	0.375

- SA improved the solution in more than 50% of the tested instances

Next steps - Future Work

Implement different strategies when generating a perturbative solution:

- Find the machine with the maximum load (makespan) and split the last job performed into other machines
- Find the machine with the maximum load (makespan) and the job with the maximum setup time and swap it with the previous job

Add the resource constraints

Apply a genetic algorithm to the problem

National Technical University of Athens
School of Electrical and Computer Engineering
MSc Data Science & Machine Learning

Thank you!

Apostolos Moustakis (03400182)
Student of DSML

Avgerinos, I. et al. (2022) ‘Scheduling unrelated machines with job splitting, setup resources and sequence dependency’, International Journal of Production Research, 61(16), pp. 5502–5524. doi:10.1080/00207543.2022.2102948.7