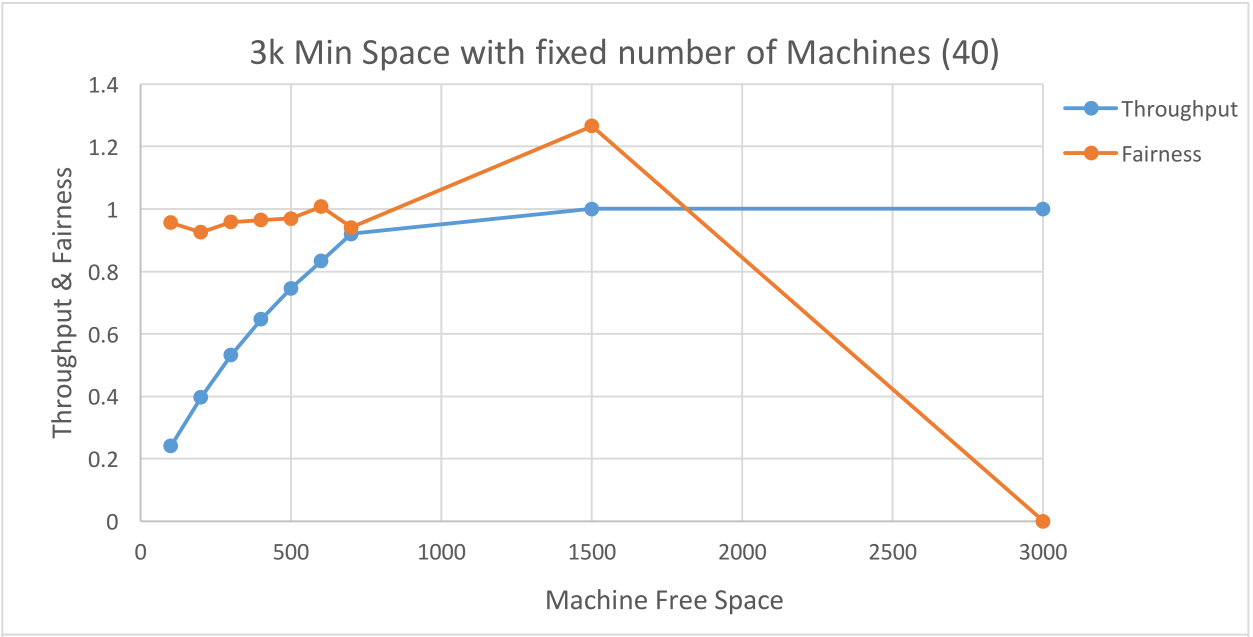
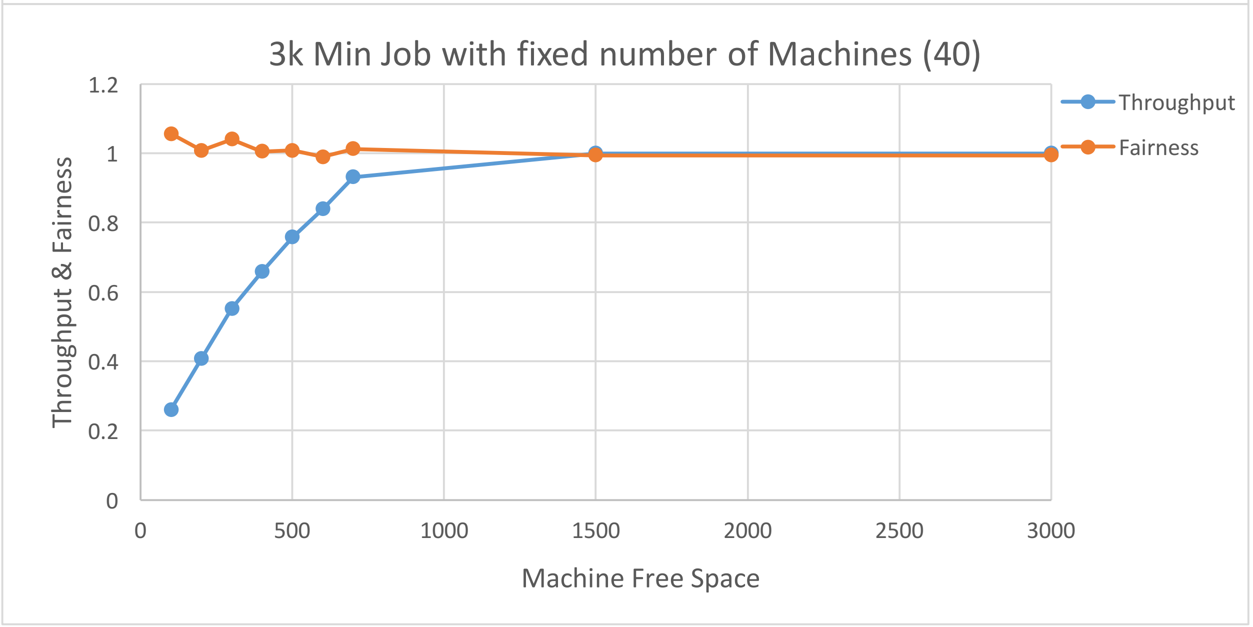
CS251 FALL 2016 Project 4

Seungho Yang
PSO 10

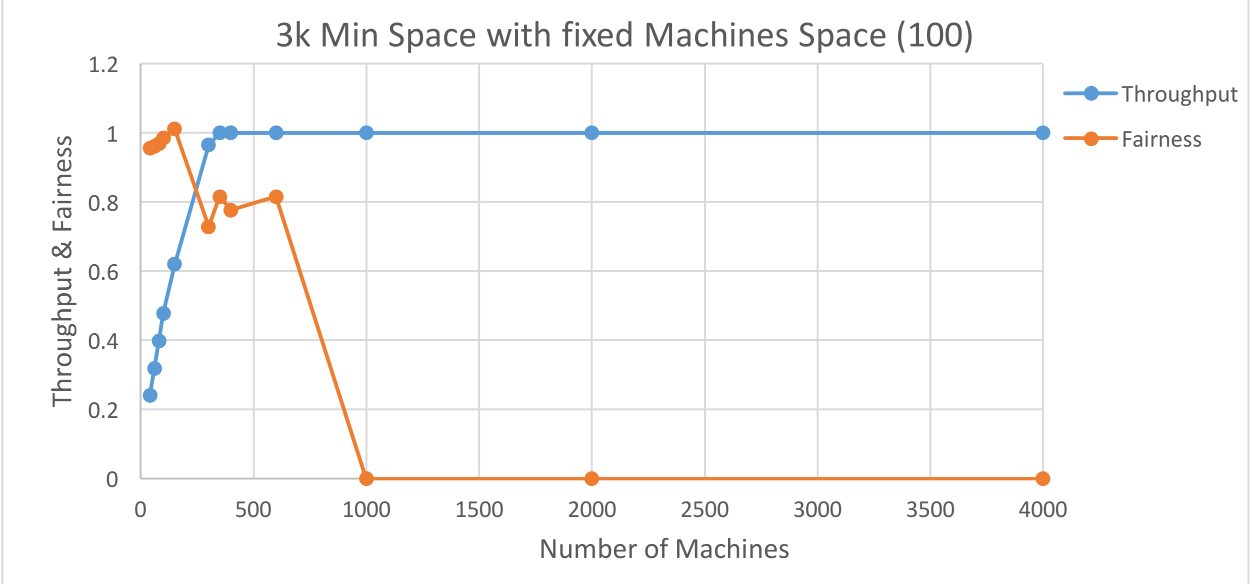
Experimental Analysis (M was inserted at the end of the file for each text files)

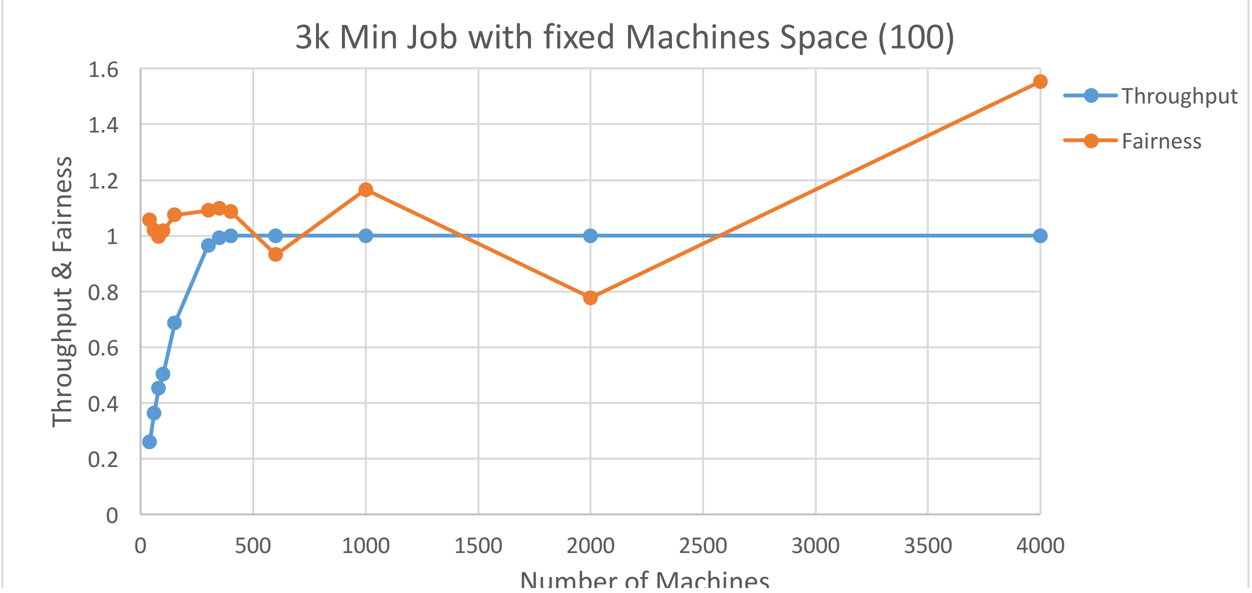
Text File: 3k.txt		**Min Space**		**Min Job**	
Number of Machines	Machine Free Space	Throughput	Fairness	Throughput	Fairness
40	100	0.241	0.956023	0.260333	1.056338
40	200	0.396667	0.92511	0.407667	1.008584
40	300	0.532	0.958466	0.552333	1.040551
40	400	0.646667	0.96401	0.658667	1.005657
40	500	0.746667	0.970027	0.757667	1.008043
40	600	0.833667	1.007678	0.839667	0.989534
40	700	0.920333	0.941378	0.931333	1.013086
40	1500	1	1.266019	1	0.994175
40	3000	1	0	1	0.994175



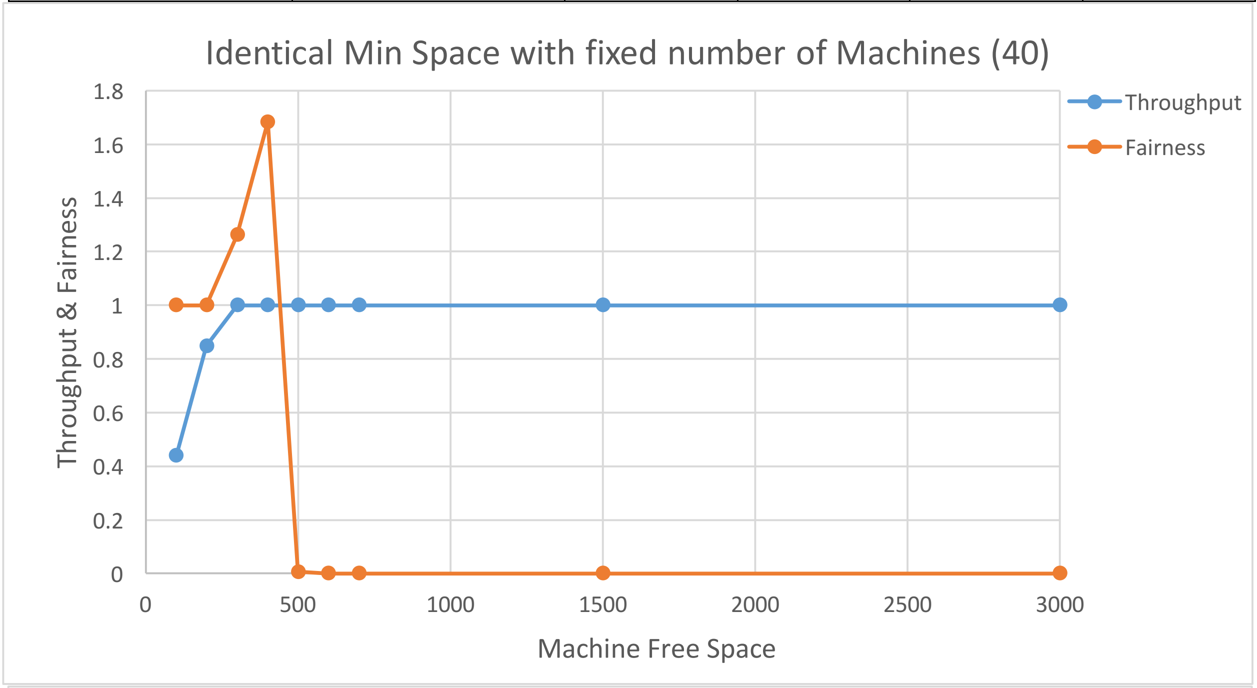


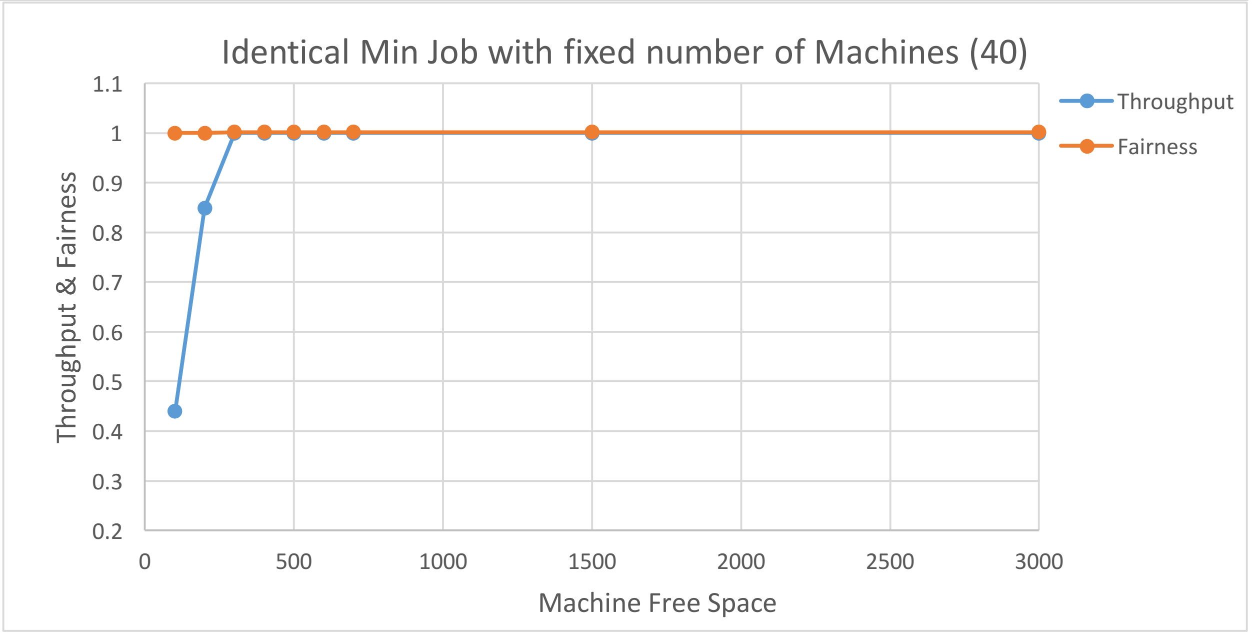
Text File: 3k.txt		**Min Space**		**Min Job**	
Number of Machines	Machine Free Space	Throughput	Fairness	Throughput	Fairness
40	100	0.241	0.956023	0.260333	1.056338
60	100	0.318667	0.961003	0.364333	1.020656
80	100	0.397667	0.969163	0.452	0.996169
100	100	0.478667	0.985663	0.503	1.016949
150	100	0.62	1.011463	0.687	1.074627
300	100	0.965	0.728745	0.965667	1.090909
350	100	1	0.815534	0.992333	1.097179
400	100	1	0.776699	1	1.087379
600	100	1	0.815534	1	0.932039
1000	100	1	0	1	1.165049
2000	100	1	0	1	0.776699
4000	100	1	0	1	1.553398



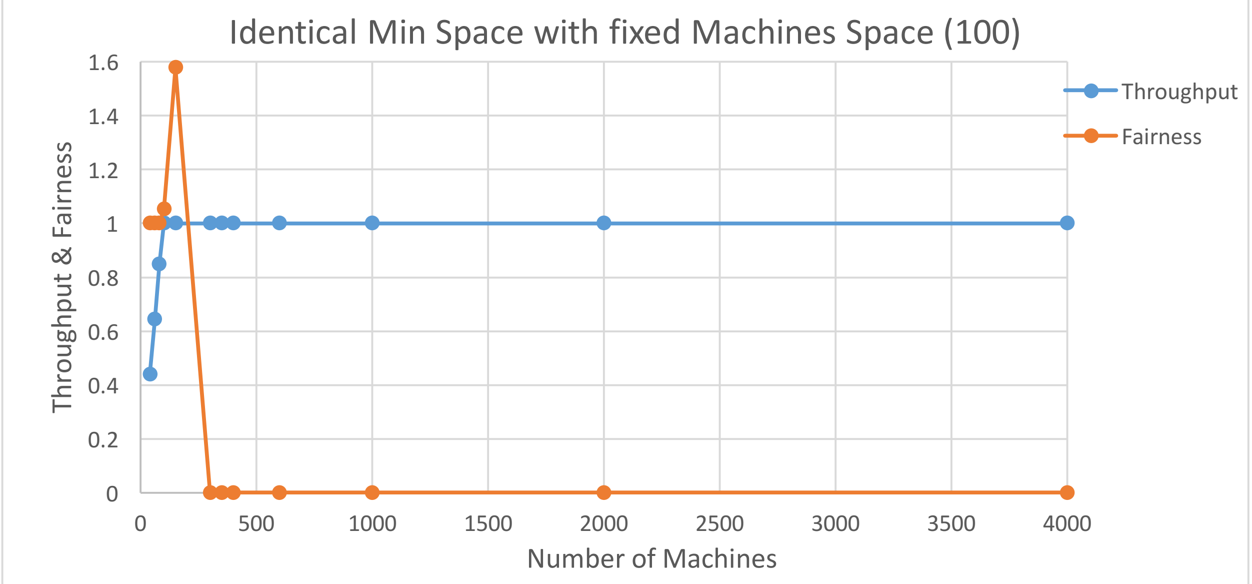


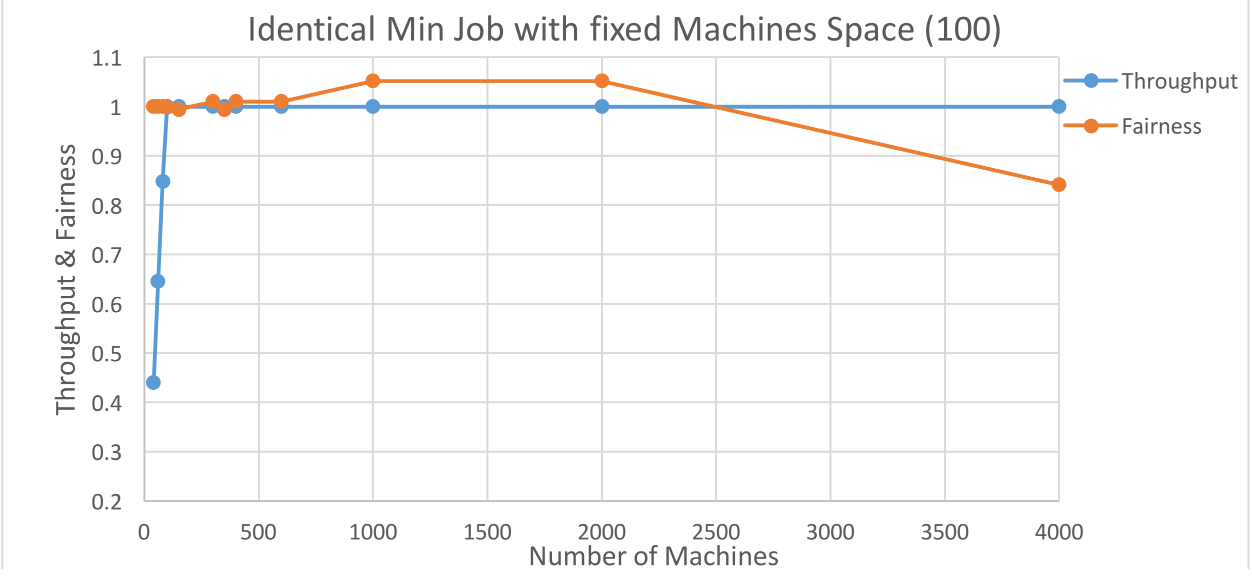
Text File: Identical.txt		**Min Space**		**Min Job**	
Number of Machines	Machine Free Space	Throughput	Fairness	Throughput	Fairness
40	100	0.4399	1	0.4399	1
40	200	0.8488	1	0.8488	1
40	300	1	1.262759	1	1.001789
40	400	1	1.683679	1	1.001789
40	500	1	0.006314	1	1.001789
40	600	1	0	1	1.001789
40	700	1	0	1	1.001789
40	1500	1	0	1	1.001789
40	3000	1	0	1	1.001789





Text File: Identical.txt		**Min Space**		**Min Job**	
Number of Machines	Machine Free Space	Throughput	Fairness	Throughput	Fairness
40	100	0.4399	1	0.4399	1
60	100	0.6452	1	0.6452	1
80	100	0.8488	1	0.8488	1
100	100	1	1.052299	1	0.999684
150	100	1	1.578449	1	0.994423
300	100	1	0	1	1.010207
350	100	1	0	1	0.994423
400	100	1	0	1	1.010207
600	100	1	0	1	1.010207
1000	100	1	0	1	1.052299
2000	100	1	0	1	1.052299
4000	100	1	0	1	0.841839





In general, min-job strategy had slightly better throughput values. It may have assigned couple of more jobs by checking number of jobs in the machines and then by breaking the ties using free size. In terms of fairness, min-job strategy had consistently better values and was very close to 1 even when number of machines or memory size has increased. This is because the implementation tries to evenly distribute the jobs instead of trying to fill one machine until it becomes full or can't take sizes over certain value. As number of machines or memory size increased, fairness values for min-space strategy dropped and reached 0. Min job strategy assigned slightly more jobs than the min space strategy but utilized the machines in balanced order. Min space strategy relied on fewer machines as machine free space of number of machines increased which reflects negative slope in the charts.

Analytical Questions

1.

Count – Starting from the root, if the root's free size is greater than or equal to the required free space, the method accounts for all the nodes in the right subtree. This saves the trouble of checking all the values in right subtree as all the nodes with higher free space are in the right. Then, the implementation runs count on the left subtree and repeats the process. When the method reaches to the left most node, it would stop as all the leaves are null. If root's free space is less than the required free space, method can ignore left subtree and continue it's search along the right subtree of the root using recursion. Asymptotic worst case approach for this case is O(log N) since we don't need to go back up the tree to restart search on unexplored nodes. Using the node's attributes and properties of Red Black binary tree, the method can easily count number of nodes that meet required free space while only visiting about log N nodes.

scheduleMinSpace – This strategy is like searching for a node that meets the requirement in binary search trees with a small difference. Starting from the root, if it satisfies the required job size, the method traverses to the left subtree and compares it's free space to current best node. If the root or current node does not satisfy, we move to the right subtree and continue the search. The implementation does not search all the free values in each of the nodes. Instead it uses the property of binary search trees again to narrow down to the best node with minimum space for the job in O(log N) time. It doesn't need to to go back up and restart the search again.

scheduleMinJob – This approach is similar with min space and does not need to check all of the nodes in the tree. From the node, check if it's free value can satisfy the required space for the job. If it satisfies, check the minJobsNode from the right subtree with it and find the better node. Then compare with the current best node to find the best one. After that run the method recursively using the left subtree until it reaches the base case where the input node is null. If the root doesn't satisfy, move to the right subtree and repeat the loop until current node satisfies the required free space. If the required free space is larger than the initial free space of the machines, method will run until it reaches right-most node and return -1 since there are no nodes that meet the requirements. When the root satisfies, it can run the method recursively until it uses a null node as input. This method only needs to access log N nodes to find the best node from N nodes.

2. Suppose there is a set of 10 jobs with the following memory sizes: {1, 2, 3, 4, 5, 7, 7, 8, 9, 10} and 4 machines with M of 14. If they are inserted in the following order, min-space strategy will discard 2 jobs: [3, 4, 2, 1, 10, 9, 8, 5, 7, 7]. First 4 jobs with memory 4, 3, 2, and 1 will be assigned to 1st machine. 5th job of memory 10 will be assigned to machine 2. 6th jobs of size 9 will be assigned to machine 3, and 7th job will be assigned to machine 4. 8th job with memory 5 will be assigned to machine 3 and it will have no free space. Last two jobs will be discarded because there are no machines with enough free space.

Machine 1	Machine 2	Machine 3	Machine 4
4	10	9	8
3		5	
2			
1			

My assignment strategy places jobs in machines if free – job size > $\frac{M}{2}$ or if free – job size == 0 when numJobs > 0. So first job with memory 3 will be assigned to Machine 1. Second job will go to machine 2 as 3 + 4 = 7. Next two jobs of memory will go to machine 1 as 14 – (3+2+1) = 8 > 7. 5th job of memory 10 will go to machine 2 since the sum will equal 0. Next job will go to machine 3 as it can't go to machine 1 and 2. Job with size 8 will go to first machine as the sum equals 0. 8th job will go to machine 3 and it's free size will be 0. Last two jobs will go to machine 4 and all jobs will be assigned.

Machine 1	Machine 2	Machine 3	Machine 4
3	4	9	7
2	10	5	7
1			
8			

3. Suppose there is a set of 10 jobs with the following memory sizes: {1, 2, 3, 4, 5, 7, 7, 8, 9, 10} and 4 machines with M of 14. If they are inserted in the following order, min-job strategy will discard 2 jobs: [1, 2, 3, 4, 5, 7, 7, 8, 9, 10]. First 4 jobs will be assigned to each of the machines. 5th job of memory 5 will be assigned to machine 4. 6th jobs of size 7 will be assigned to machine 3, and 7th job will be assigned to machine 2. 8th job with memory 8 will be assigned to machine 1. Last two jobs will be discarded because there are no machines with enough free space.

Machine 1	Machine 2	Machine 3	Machine 4
1	2	3	4
8	7	7	5

My assignment strategy from question 2 can work on this input sequence as well. Place jobs in machines if free – job size > $\frac{M}{2}$ or if free – job size == 0 when numJobs > 0. First 3 jobs go to machine 1. Next three jobs of size 4, 5, and 7 go to machine 2, 3, and 4 respectively. Last four jobs will be assigned respectively to each machine where each of them will have no free memory.

Machine 1	Machine 2	Machine 3	Machine 4
1	4	5	7
2	10	9	7
3			
8			

4. If all 20 jobs are scheduled, that means that they are assigned to the 20 machines. To have a fairness greater than 1.8, we need at least 2 more extra jobs. 11 machines will have 2 jobs assigned to each of them while the remaining 9 machines will have none. Median of the machines will be 2 based on the calculation $\frac{2+2}{2} = 2$ since 10th and 11th machines (in sorted order) will have 2 jobs. Ideal equals $\frac{\#\ of\ Jobs}{\#\ of\ Machines} = \frac{22}{20} = 1.1$. Fairness is measured by $\frac{median}{ideal} = \frac{2}{1.1} = 1.8182$

5. The best approach to find the jobs (Id and size) assigned to a node is to iterate through jobs hash map. Hashmaps can be iterated by using their size() method which returns the size of the hash map. If we iterate from 0 to n, we can find the jobs associated with a specific node and its job id and size. Time complexity of this approach would take O(n) as we need to go through all n jobs to identify associated jobs to a node.