Simulation of Multi-Agent Based Scheduling Algorithms for Waiting-line Queuing Problems

New Mexico Supercomputing Challenge

Final Report

April 4th, 2012

Team 73

Los Alamos Middle School

Team Members

Steven Chen

Andrew Tang

Teacher

Pauline Stephens

Project Mentor

Hsing-bung (HB) Chen

Table of contents

Executive Summary

In this project, we designed and implemented a multi-agent computer simulation program. We used this simulation software to model a real-life waiting line or queuing problem in variety of business and industrial situations such as supermarket's checkout lines and bank's teller service windows. Through our experiments we addressed the following issues: (1) How to model an independent task scheduling problem (single waiting queue (multiple servers with Multiple service queues) using NetLogo multi-agent simulation system, (2) How to provide an interactive approach to control run-time simulation activities, (3) How to collect performance data and justify implemented scheduling methods, and (4) Is it possible to create an useful Multi-agent education tool to teach scheduling problem. To solve the problems presented above, we would like to apply efficient scheduling solutions. Scheduling is a key concept in computer multitasking, the multiprocessing operating system and real-time operating system designs. Scheduling refers to the way processes assigned to run on available CPUs. This assignment is normally carried out by software known as a task scheduler or a job dispatcher. The "NetLogo" is an agent based modeling software tool that we can use to create and investigate various models for application problems. In reality, there is no universal scheduling algorithm to solve real-life waiting-line or queuing problems. However, using heuristic approaches is the most reasonable way to obtain acceptable solutions. We implemented five scheduling algorithms—round-robin, random selection, early start time first, less workload first, and a mixed selection heuristic that combines the four previously listed methods. The rich property of the random number generator in "NetLogo" is an excellent tool to generate random task behaviors such as task size and task arriving time. We conducted testing cases to cover various task patterns on our NetLogo simulation program. We defined and collected various performance matrices such as waiting time, turnaround time, and queue length. We found the Early Starting Time First algorithm to be the best heuristic in most of the testing cases. For example, it can obtain a shorter waiting time and average queue length and a faster turnaround time. We also demonstrated that our interactive multi-agent simulation program is a good tool to teach multi-processing task scheduling problem.

1. Problem statement

Waiting line queuing problems are commonly seen in everyday life. Some typical examples are:

1. Supermarkets must decide how many cash registers should be opened to reduce customers' waiting time.
2. Gasoline stations must decide how many pumps should be opened and how many attendants should be on duty.
3. Manufacturing plants must determine the optimal number of mechanics to have on duty in each shift to repair machines that break down.
4. Banks must decide how many teller windows to keep open to serve customers during the various hours of the day.
5. Peer-to-Peer, Grid, and Cloud computing need to effectively and quickly manage and schedule distributed resources for solving large-scale problems in science, engineering, and commerce.
6. Modern large scale HPC cluster machines need to schedule millions of processes or threads so it can provide fast turnaround time and better machine utilization.

Whether it is waiting in line at a grocery store to buy deli items (by taking a number), checking out at the cash registers (finding the quickest line), waiting in line at the bank for a teller, or submitting a batch job to available computers, we spend a lot of time waiting. The time you spend waiting in a line depends on a number of factors including the number of people (or in general tasks) served before you, the number of servers working, and the amount of time it takes to serve each individual customer/task.

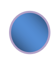
To deal with the problems mentioned above, we must provide effective and reasonable solutions in order to reach goals of minimizing wait time in a queue, minimizing turnaround time, balancing workload among service points, and increasing server utilization. To simplify our project's problem description, we would like to formalize a waiting line and queuing problem as a task scheduling problem. We can treat all objects (clients, customers, mechanics, tellers, messages, jobs, processes, threads) waiting in a service line by putting them into a queue as tasks. The task scheduling problem is very challenging and interesting. This problem is a class of hard problems that cannot be optimally solved in a reasonable amount of computation time.

For this reason, researchers have spent the past several decades developing work heuristic (rule of thumb) methods to try and find a near-optimal solution.

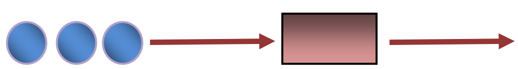
The main goal of our project is to design and implement a multi-agent simulation model for task scheduling problems and provide an interactive software tool to learn distributed task scheduling problems. Now, why are we using simulation? Simulation appears to be the only feasible way to analyze algorithms on large-scale distributed systems using various resources. Unlike using the real system in real time, simulation works well, without making the analysis mechanism unnecessary complex, by avoiding the overhead of co-ordination of real resources. Simulation is also effective in working with very large hypothetical problems that would otherwise require involvement of a large number of active users and resources, which is very hard to coordinate and build at large-scale research environment for an investigation purpose.

2 Multi-agent task scheduling simulation design and implementation

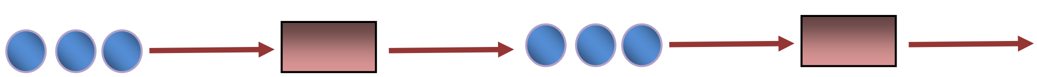
2.1 Simulation queue model

There are several Waiting line 's Queue models:  Task,  processing point

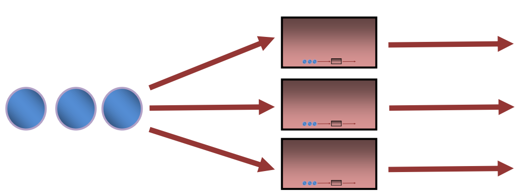
- Single-server, single-phase



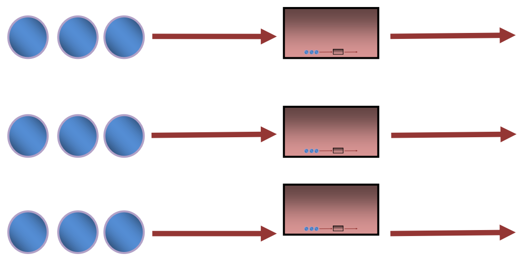
- Single-server , multiphase



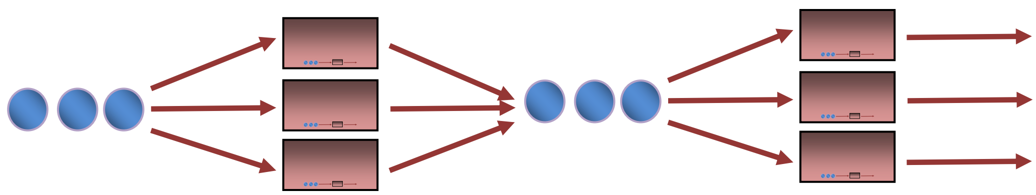
- Multi-server, single-line single-phase: centralized scheduler



- Multi-server, multiline, single-phase



- Multi-server, multiphase



In this project, we implement the multi-server, single-line, and single-phase queue model. This model typically and simply represents an independent task scheduling model on a distributed computing system such as Cluster, GRID or Cloud computing environment. A centralized task scheduler handles many randomly arrived tasks and finds available processing points to execute tasks.

2.2 Multi-agent task scheduling simulation System

Our multi-agent based models are composed of three different types of agents: the schedule agent, machine agent, and task agent.

The agents in a multi-agent system have several important characteristics:

- **Autonomy**: the agents are at least partially autonomous
- **Local views**: no agent has a full global view of the system, or the system is too complex for an agent to make practical use of such knowledge
- **Centralization and decentralization**: there is a designated and centralized scheduling agent and there are number of decentralized task agents randomly ask the scheduler to schedule a created "task" on a selective machine agent.

Figure 1 shows the system diagram of our simulation agents.

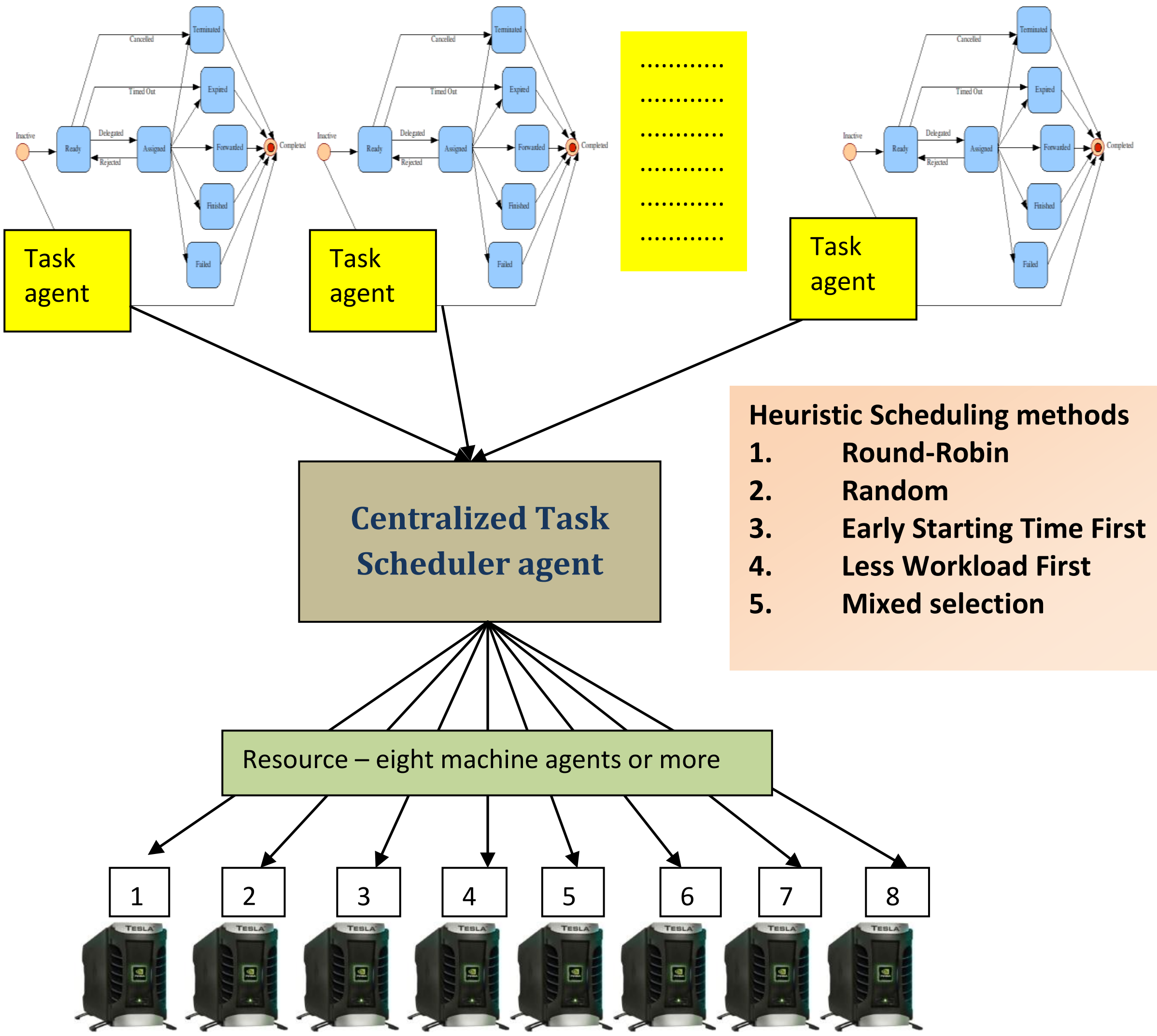


Figure 1: Multi-agent task scheduling simulation

2.3 Agent designs

Scheduler agent:

Only one centralized scheduler agent is defined here. We use the scheduler agent to receive scheduling requests from task agents and find available machine agents to run tasks. The selection of an available machine is based on the selected scheduling heuristic.

Machine agent:

A Machine agent is the main system resource to schedule a task. We have used eight and sixteen machine agents in this simulation. Each machine agent is used to receive a task assignment request from the scheduler agent and update its avail-time, accumulate-task-time, and idle time. We can simulate more machine agents but we have considered to provide and interactive approach and construct an education tool. Enploying eight to sixteen machine agents in our simulation is within a reasonalbe range to view -real-time task schedule activities on a monotoring screen.

Task agent:

We have used up to 99,999 task agents in our similation. Each task agent comes with a different arriving time and task execution length. We used various number of task agents to represent different run-time environments such as light workload (hundred tasks), moderate workload (thousand tasks), heavy workload (multiple ten thousand tasks), lots of small tasks (small execution time), lots of large tasks (very long execution time), or mixed small and large task sizes etc..

Task information

<u>Property of a Task:</u>

- Each scheduling task is an independent task. There is no dependency relation or related execution order between tasks
- Each task has been assign a random execution time, i.e. the length of a task
- Each task has been assigned a random arriving time
- Task arrival times are not known a priori. Every task has the attributes arrival time, worst case computation time, and deadline. The ready time of a task is equal to its arrival time. Task's arriving time is generated by a selected random number generator.
- Tasks are non-preemptive; each of them is independent.

We used two different random number generator provided by ~~the~~ NetLogo— random and Poisson-random. We used a random number generator to generate a task's arriving time and a task's execution time.

Task selection discipline :

The selection of a task is based on the First Come First Serve FCFS) order. The NetLogo system decides the task order in a queue when there are multiple tasks arrive at the same time tick.

2.4 Heuristic scheduling methods

We implemented five different decision-making heuristics. Various heuristic scheduling methods represent the intelligence and capabilities of each method. The heuristic is how we select a machine to execute an arriving task.

2.4.1 Round-robin method

The scheduler agent uses a round-robin order to select each machine agent and assigns an arriving task to it. Each machine agent takes an equal share of responsibility to run a task in turn.

2.4.2 Random Selection

The scheduler agent randomly selects an available machine agent and assigns ~~an~~ arriving tasks to it. It randomly selects a machine to run an incoming task.

2.4.3 Less Workload First

The scheduler agent selects an available machine agent with the smallest accumulated task workload and assigns an arriving task to it.

2.4.4 Early Starting Time First

The scheduler agent selects an available machine agent with the early task starting time to run a task and assigns an arriving task to it.

2.4.5 A Mixed Selection of the Above Four Heuristic Methods

A mixed selection of the above scheduling methods can be called a heuristic of heuristics. For each arriving task, the scheduler agent randomly picks one of the ~~above~~ four heuristic methods mentioned above and applies this selected method to find an available machine agent, and then assigns an arriving task to it.

2.5 Time step simulation and Task interactive sequence

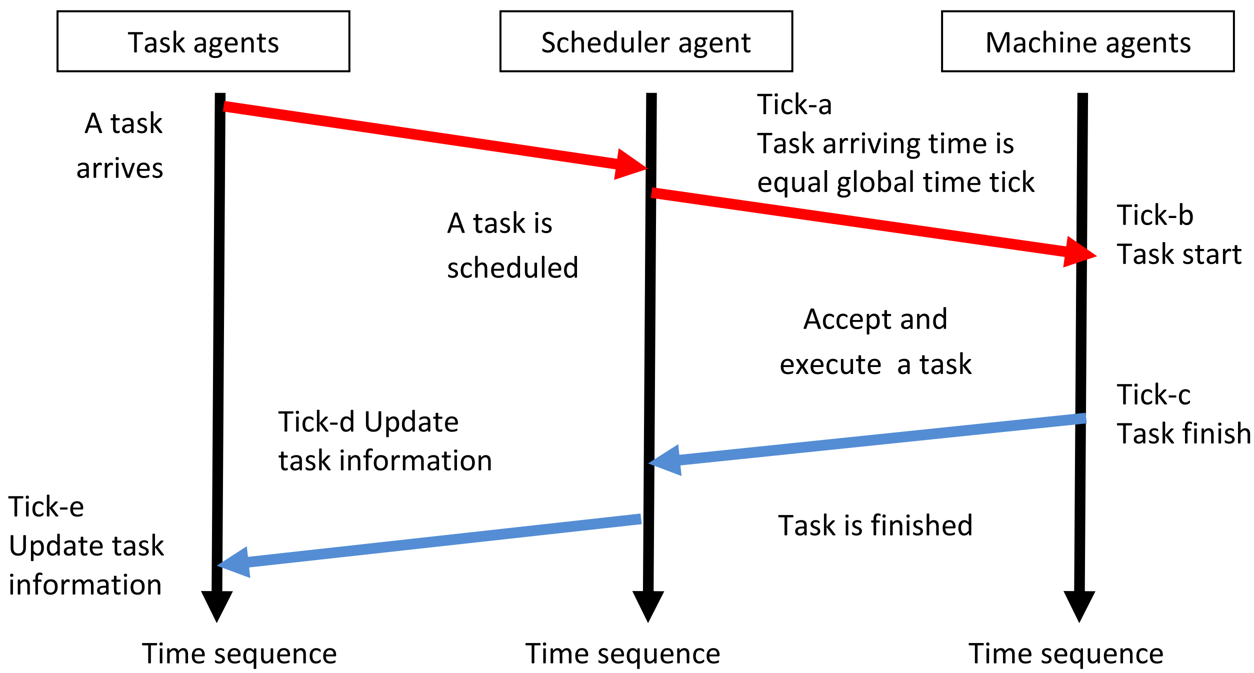


Figure 2: Interaction between agents based on time sequence

A global time tick is used in the simulation. This global time tick is advanced by "N" time ticks. "N" can be any number. We advanced one time tick each time. The global time tick is used as the wall clock and we used it to check the task arriving time. We also use it to monitor task activities such as task waiting, task scheduling, task execution, and task finishing. We used the global time tick to collect performance data. When a task's arriving time is equal to the current global time tick, this task agent will ask the scheduler agent to scheduler agent to find an available machine agent to execute it. Figure 2 shows the interaction between agents based on the time advanced sequence.

In Figure 3, we show the simulation NetLogo program architecture. "Ask" is the keyword used to query each agent about its status and expect activities. "Ask" also represents the required interactive activities between agents. A task agent checks its task-arriving time and the global time tick and sees if its task is ready to be scheduled. The scheduler agent use a selected heuristic method to find an available machine "X" and then ask the machine agent "X" to accept the arriving task and execute it. These interactive activities among agents are continuing until all tasks are scheduled and finishing execution. We collected performance data during the whole simulation process.

One of our goals for this simulation project is to provide education tolls for task scheduling problems. We adapted a visualized and interactive approach to build this simulation. We let users to define the run-time environment while we provided run-time animations of task scheduling activities and in-time performance data display during the whole simulation process.

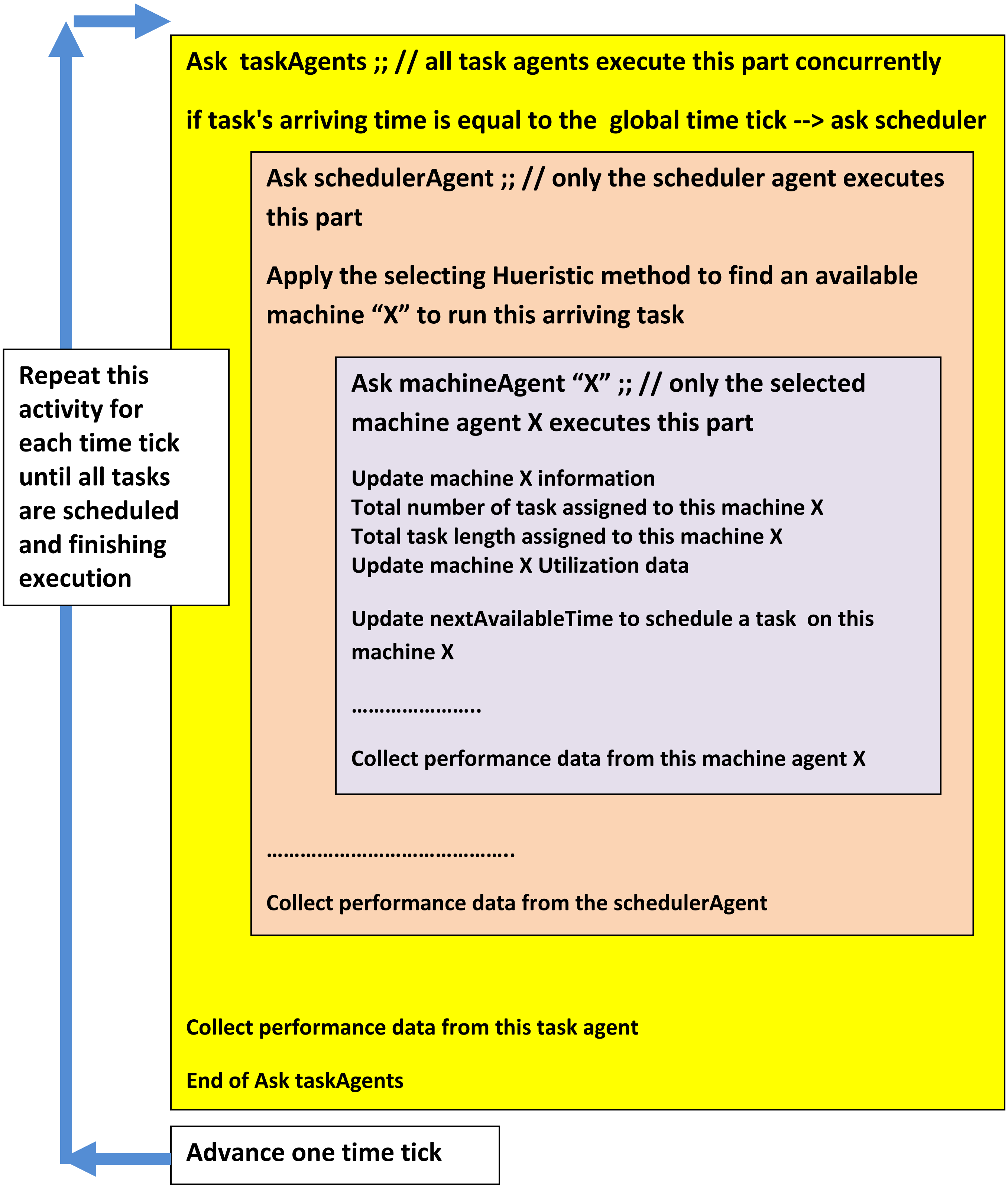


Figure 3: NetLogo simulation program -interaction between agents

2.6 Main Screen Design and Implementation

The Global tick count is shown in Figure 4-1.

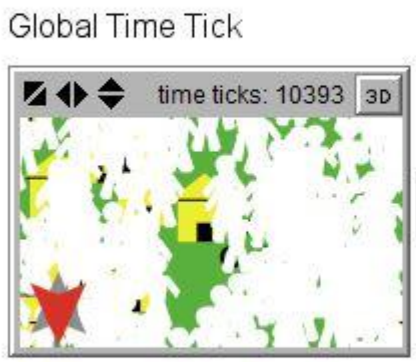


Figure 4-1: Global time tick counter

Users define parameters for simulation (Figure 4-2)

- the number of task agents used in each simulation,
- the range of task arriving time distribution,
- the range of task execution time distribution,
- the number of machine used in simulation

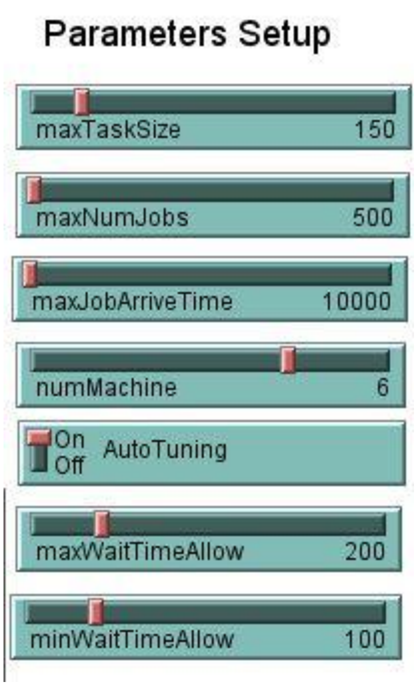


Figure 4-2: Setup testing parameters

Users select a random number generator used in simulation (Figure 4-3).

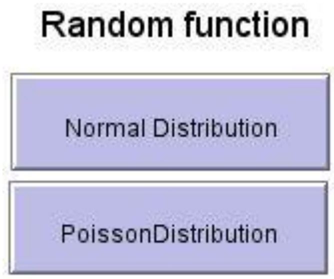


Figure 4-3: Select random number generator

Users select a scheduleing method used in simulation (Figure 4-4).

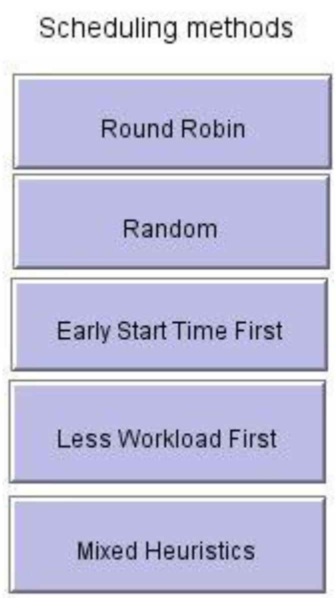


Figure 4-4: Select schedule method

Users interactively control simulation action (Figure 4-5).

Figure 4-5: Run time control bottom

Monitoring run time performance data update and display (Figure 4-6)

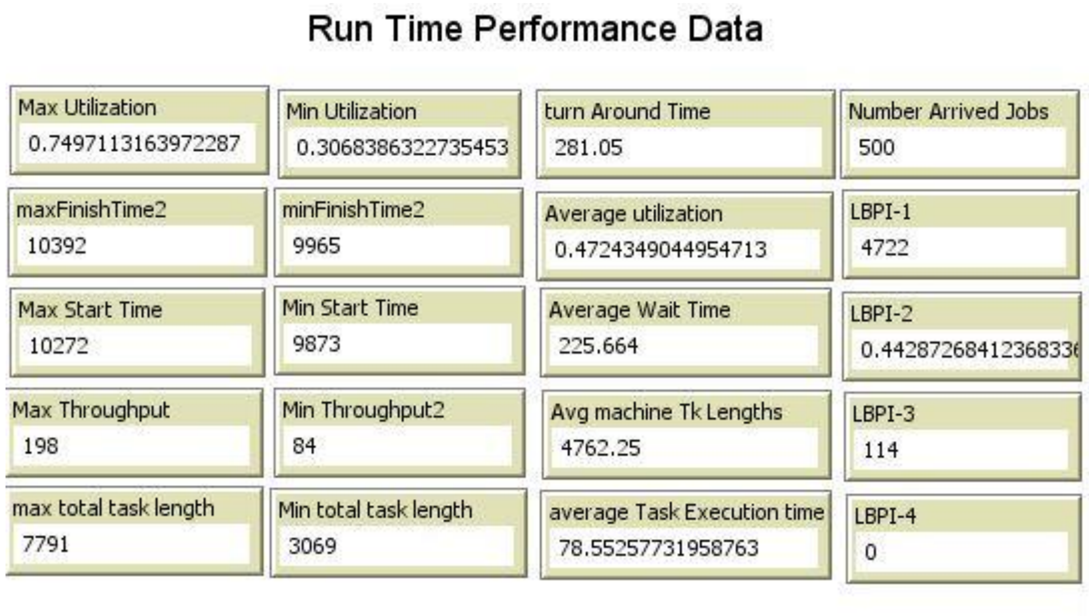


Figure 4-6: Rum Time Performance data update and Display

Messages area for displaying Testing Setup information and Run time activities is shown in (Figure 4-7)

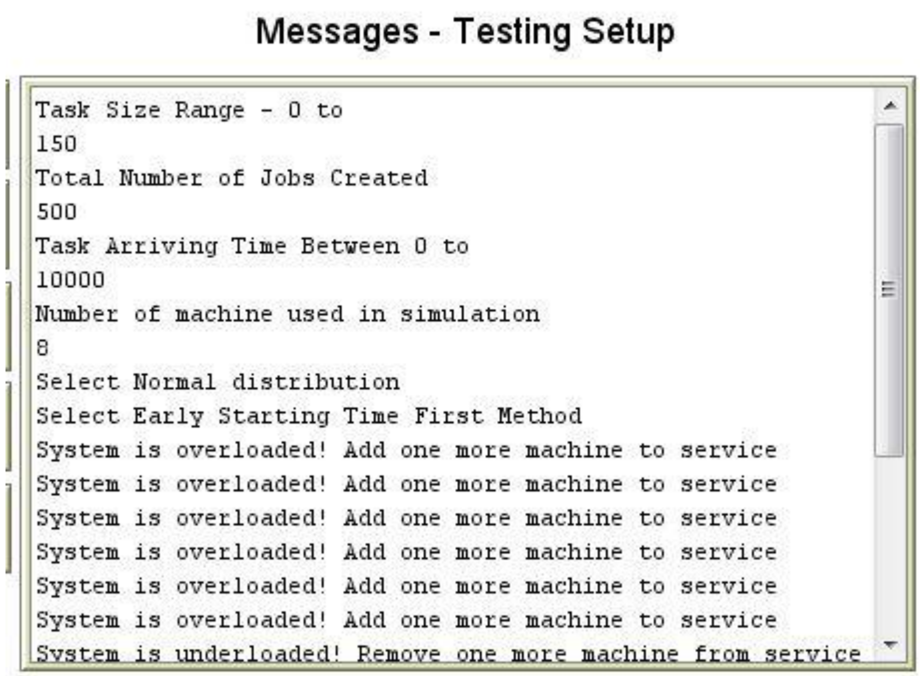


Figure 4-7: Message area for Testing setup and activities

Run Time visualization display for task scheduling activities is shown in Figure 4-8.

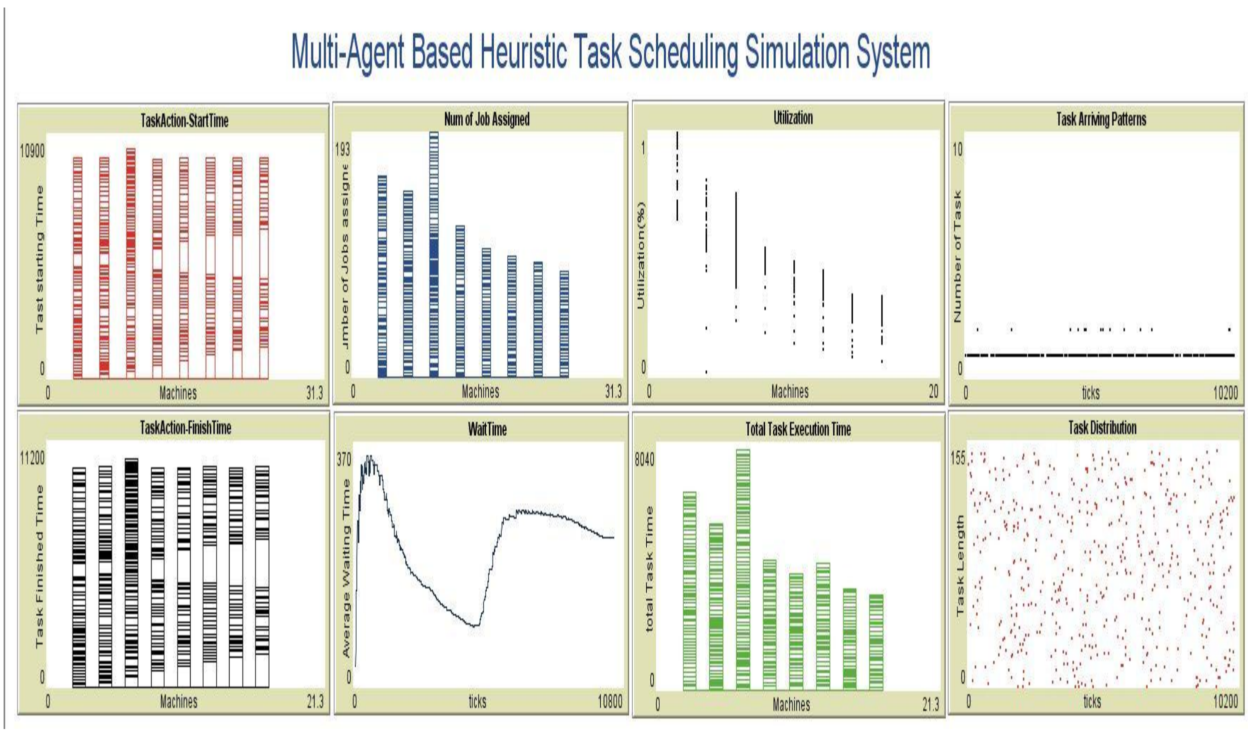


Figure 4-8: Run time visualization area

Figure 4-9 is the main screen for our multi-agent task scheduling simulation system.

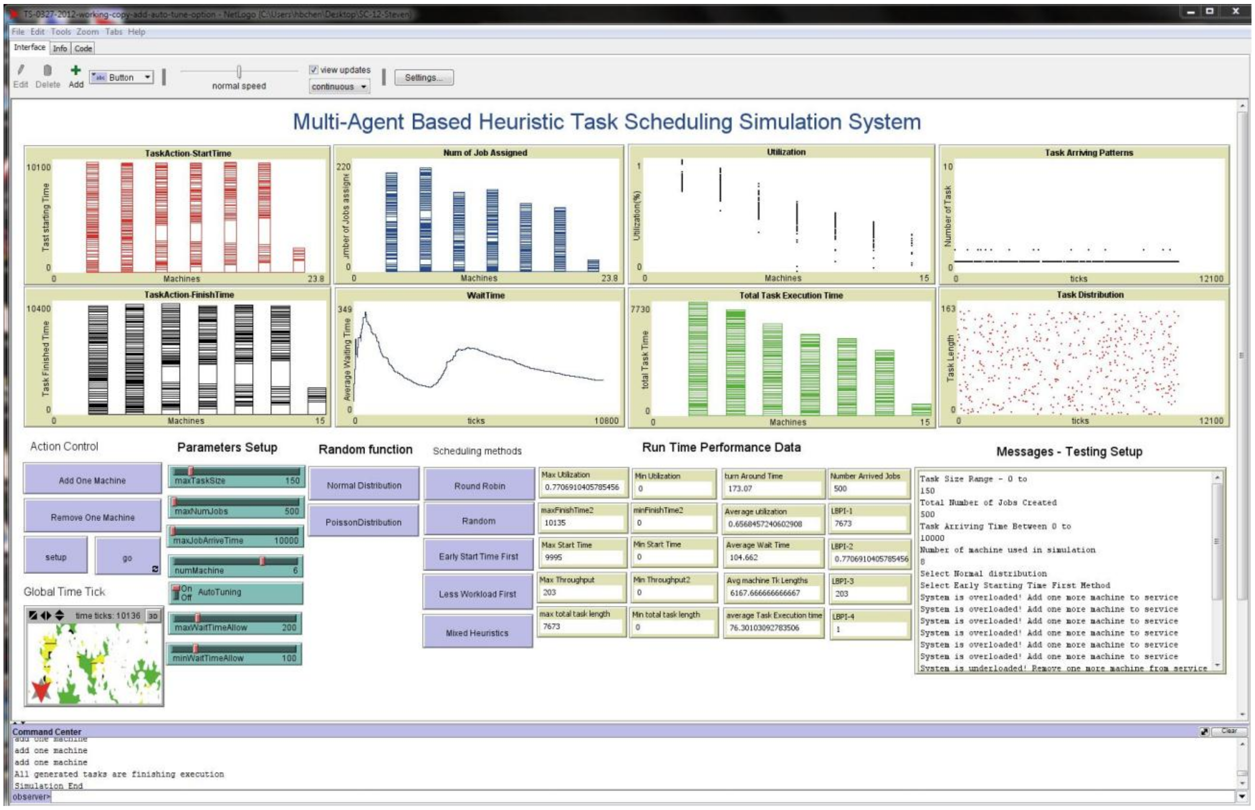


Figure 4-9: Main screen design

3 Testing and Performance data

We focused on the following performance Index:

a) The average number of tasks waiting in line on a machine agent - The number of tasks waiting in line can be interpreted in several ways. Short waiting lines can result from relatively constant task arrivals (no major surges in demand) or by the organization having excess capacity (too many machines open). On the other hand, long waiting lines can result from poor server efficiency, inadequate system capacity, and/or significant surges in demand.

b) The average time tasks spend on waiting in a queue,

c) The average time a task spends in the system - turnaround time

d) The system utilization rate - Measuring capacity utilization shows the percentage of time the machines are busy. Management's goal is to have enough machines to assure that waiting is within allowable limits but not too many machines as to be cost inefficient.

3.1 Performance index definitions

We defined the following parameters in our simulation program and then collected them as performance data.

Global information - can be viewed and accessed by all agents

ticks : the global time tick as the reference wall clock

Number of Task: N, $Task_i$, 1 = 1 to N

Number of machine: M, $Machine_j$, j= 1 to M

Num of Scheduler: 1 , Scheduler

Task agent information:

Number of Task agent created : N, $Task_i$, 1 = 1 to N , i is referenced as the task ID

A Task i: $Task_i$

Task i execution time : $TaskLength_i$

A random number generator is used to create a task's execution time

Task i arriving time : $TaskArrive_i$

A random number generator is used to create a task's arriving time

Task i start executing time: $TaskStart_i$

Task i finish execution time : $TaskFinish_i$

$TaskFinish_i = TaskStart_i + TaskLength_i$

Task i Waiting time in queue: $TaskWait_i$

is the time between task's arriving time and the actual task's start execution time

$TaskWait_i = TaskStart_i - TaskArrive_i$

TotalTaskWaitingTime(Sum of $TaskWait_i$, i= 1 to N)

AverageWaitingTime: Average task waiting time

AverageWaitingTime = TotalTaskWaitingTime / Number of Task arrived

Task i Turnaround time: $TaskTRtime_i$, the amount of time spend on waiting and execution

$taskTRtime_i = TaskFinish_i - TaskArrive_i$

TotalTaskTurnaroundTime(Sum of $TaskTRTime_i$, i= 1 to N)

AverageTurnaroundTime: Average task turnaround time

AverageWaitingTime = TotalTaskTurnaroundTime / Number of Task Finished

Machine agent information

Number of machine: M, $Machine_j$, j= 1 to M, j is referenced as the machine ID

We create two version of simulation. One is using eight machine agents and the other is using sixteen machine agents

A machine j : $Machine_j$

$MachTotalTaskTime_j$: Total task execution time on a machine j

$MachAvailableTime_j$: the current available time to add a new task on a machine j

MachIdleTimej : Machine j Idle time upto the current golbal time ticks

MachIdleTimej = ticks - $MachTotalTaskTime_j$

MinimunTotalTaskTime ($MachTotatlTaskTime_{j,}$ j = 1 to M)

The smallest total task time on a machine agent

MaximunTotalTaskTime($MachTotalTaskTime_{j,}$ j = 1 to M)

The biggest total task time on a machine agent

$MachUtilization_j$: A machine j current utilization

$MachUtilization_j = MachTotalTaskTime_j / ticks$

The less utilized machine : MiumumUtil($MachUtilization_j$, j = 1 to M)

The most utilized machine : MaxmumUtil($MachUtilization_j$, j = 1 to M)

The average machine utilization : Average($MachUtilization_j$, j = 1 to M)

EarlyStartTime(Minumum($MachAvailableTime_j$, j = 1 to M)): the early start time for a task from all machine agents

$MachFinishTask_j$: Number of tasks finished on a machine agent j at current global time ticks

This is the throughput on a machine agent j

MinimunThroughput(MachFinishTask$_{j,}$ j = 1 to M)

MaximunThroughput(MachFinishTask$_j$, j= 1 to M)

TotalThroughput(MachFinishTask$_j$, j= 1 to M)

TaskWaitingEueueLength$_j$: waiting queue lenght on a machine agent j

MinimunQueuLength(TaskWaitingEueueLength$_j$, j=1 to M)

The minimum queue length

MaximunQueuLength(TaskWaitingEueueLength$_j$, j=1 to M)

The maxinum queue length

Load Balance Performance Index :

Load balance Performance Index1 (LBPI1) =

MaximumTotalTaskTime - MinimumTotaltaskTime

Check the total task execution time assigned on a machine agent

Calculate the biggest gap among machine agents

Load balance Performance Index2 (LBPI2) = MaximumUtil - MinimumUtil

Check the machine utilization on a machine agent

Calculate the biggest gap of utilization among machine agents

Load balance Performance Index3 (LBPI3) = MaximumThroughout - MinimumThroughout

Check the total task execution time assigned on a machine agent

Calculate the biggest gap of throughout among machine agents

Load balance Performance Index4 (LBPI4) =

MaximumQueueLength - MinimumQueueLength

Check the total task execution time assigned on a machine agent

Calculate the biggest gap of waiting queue length among machine agents

We defined these four Load Balance Performance Index to measure the capability of providing a local balance run-time environment for each task scheduling method. The less variation of the LBPI value indicates a better load balancing result that is there is not a large difference between the "min" and "max". For example, we use the LBPI2 to check whether a scheduling method can assign even workload to each machine.

3.2 Testing cases

We used the task pattern

- Task size distribution: range 0 to 720 time ticks
- Number of Task agents : 1000 task agents created
- Task arriving time distribution: range from 0 to 100000 time ticks
- Using the default normal random number generator

We measured both strong scaling and weak scaling performance.

3.2.1 Strong scaling testing cases

In this strong scaling test case, the problem size stays fixed but the number of machines used is increased.

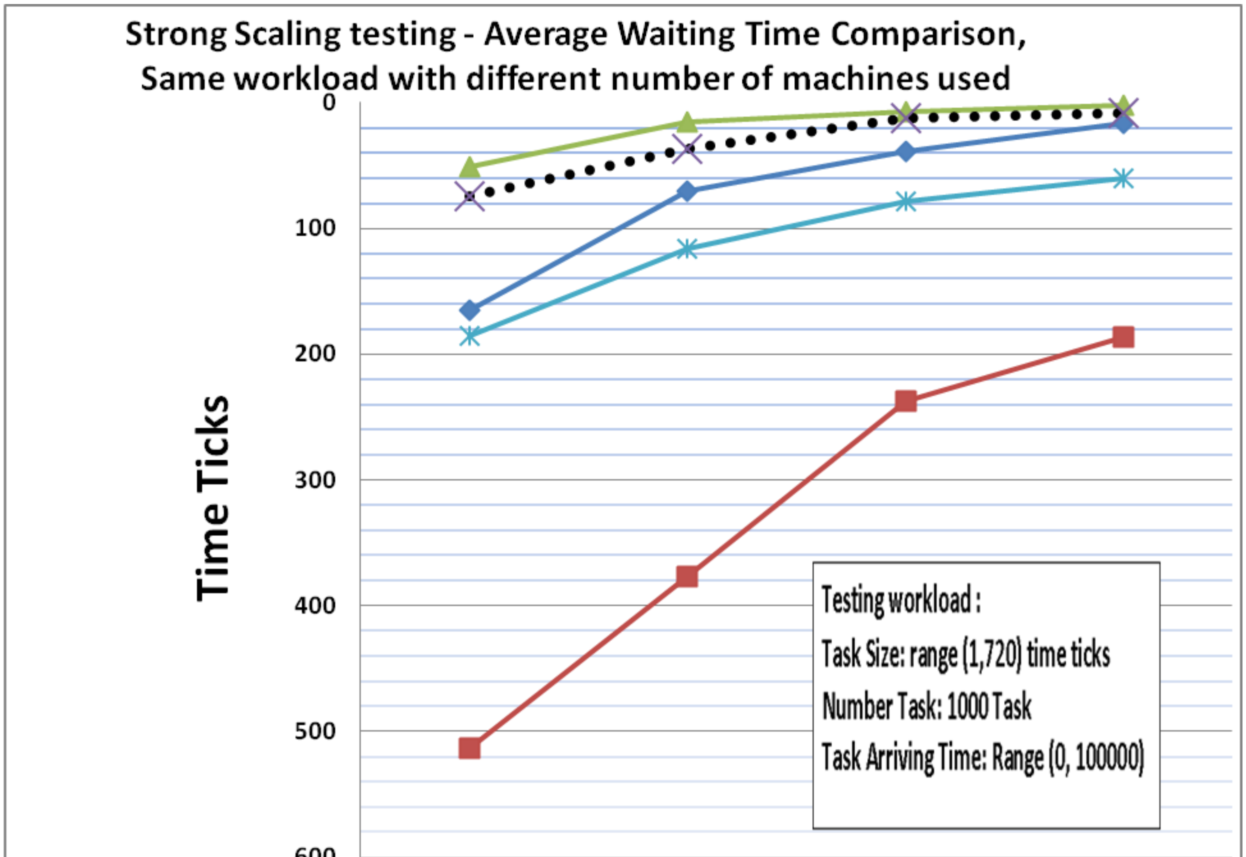
In strong scaling testing, we apple the above task pattern and start with using five machine in testing. For each round of testing, we applied the same workload and then increased one machine for each round of testing until we used up to eight machines.

Figure 5-1 shows the result of average waiting time comparison. The result is shown that every scheduling can reduce the waiting time when more machines are added to the simulation. We use a normalized ScalingIndex to compare the average waiting time. Table-1 shows the formula we used to calculate the ScalingIndex. The higher ScalingIndex value indicates a better scaling result.

	Average Waiting time	Normalized ScalingIndex-Average waiting time
Using 5 machines	**A**	**A/A = 1**
Using 6 machines	**B**	**A/B**
Using 7 machines	**C**	**A/C**
Using 8 machines	**D**	**A/D**

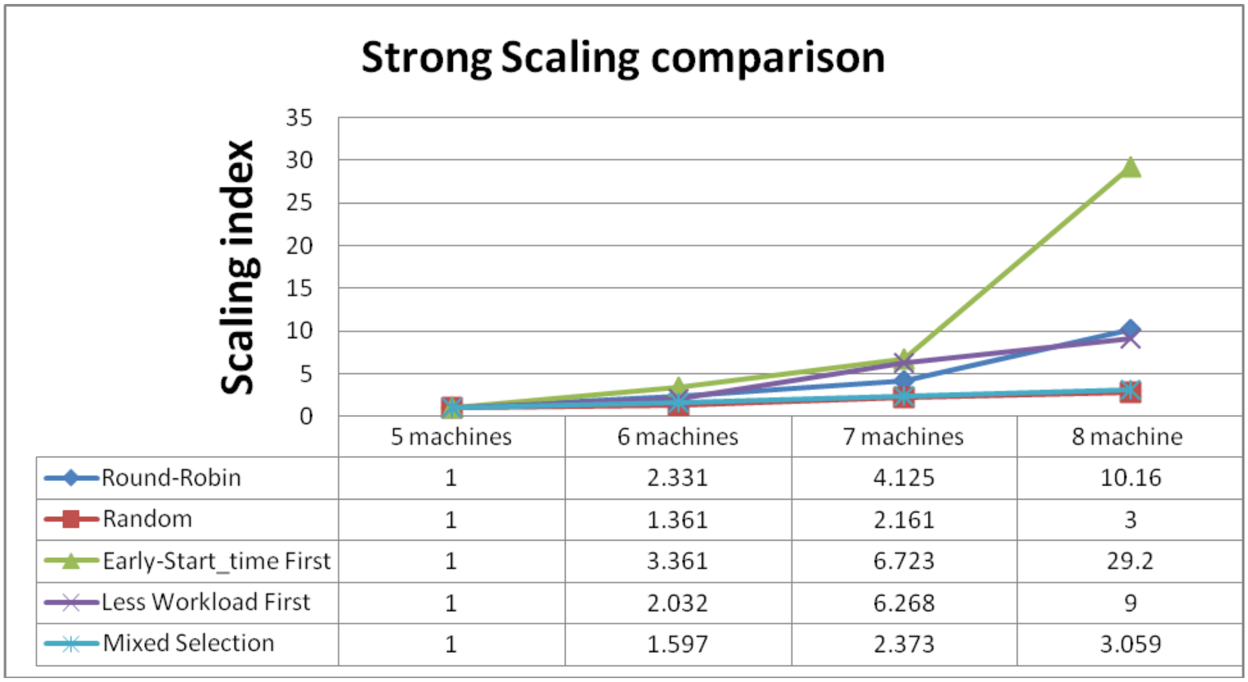
Table-1: Normalized ScalingIndex for Average Waiting Time

Figure 5-2 shows the ScalingIndex: Average Waiting Time comparison. The early Start-Time First method demonstrates as a better scheduling method in terms of strong scaling comparison.



	5 machines	6 machines	7 machines	8 machines
Round Robin	164.6	70.6	38.3	16.2
Random	512.9	376.8	237.3	186.4
Early Start-Time First	51.1	15.2	7.6	1.75
Less Workload First	74.6	36.7	11.9	8.2
Mixed Selection	185.1	115.9	78	60.5

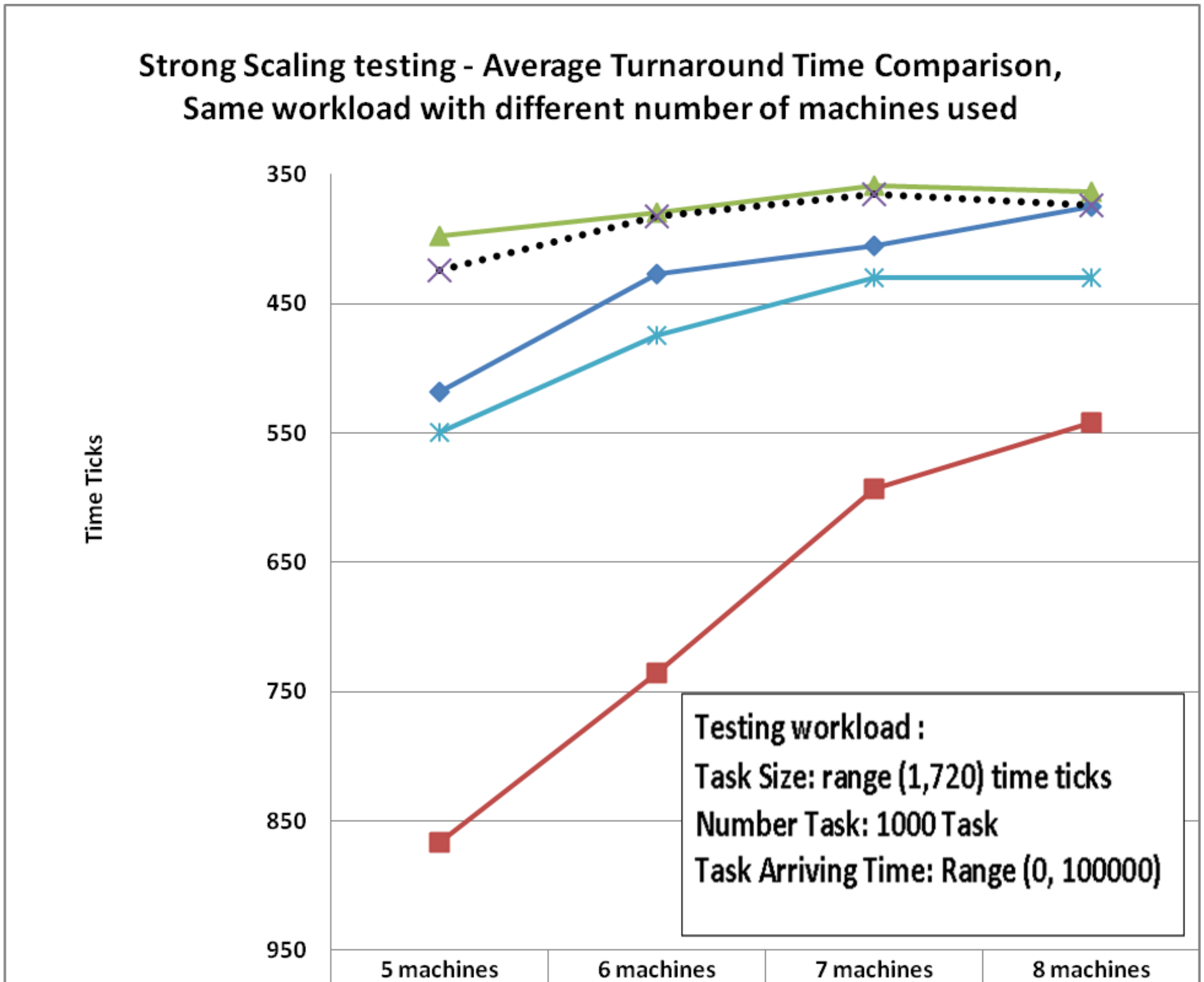
Figure 5-1: Strong scaling testing and average waiting time comparison, same workload with different number of machines



	5 machines	6 machines	7 machines	8 machine
Round-Robin	1	2.331	4.125	10.16
Random	1	1.361	2.161	3
Early-Start_time First	1	3.361	6.723	29.2
Less Workload First	1	2.032	6.268	9
Mixed Selection	1	1.597	2.373	3.059

Figure 5-2: Normalized ScalingIndex for Average waiting Time comparison

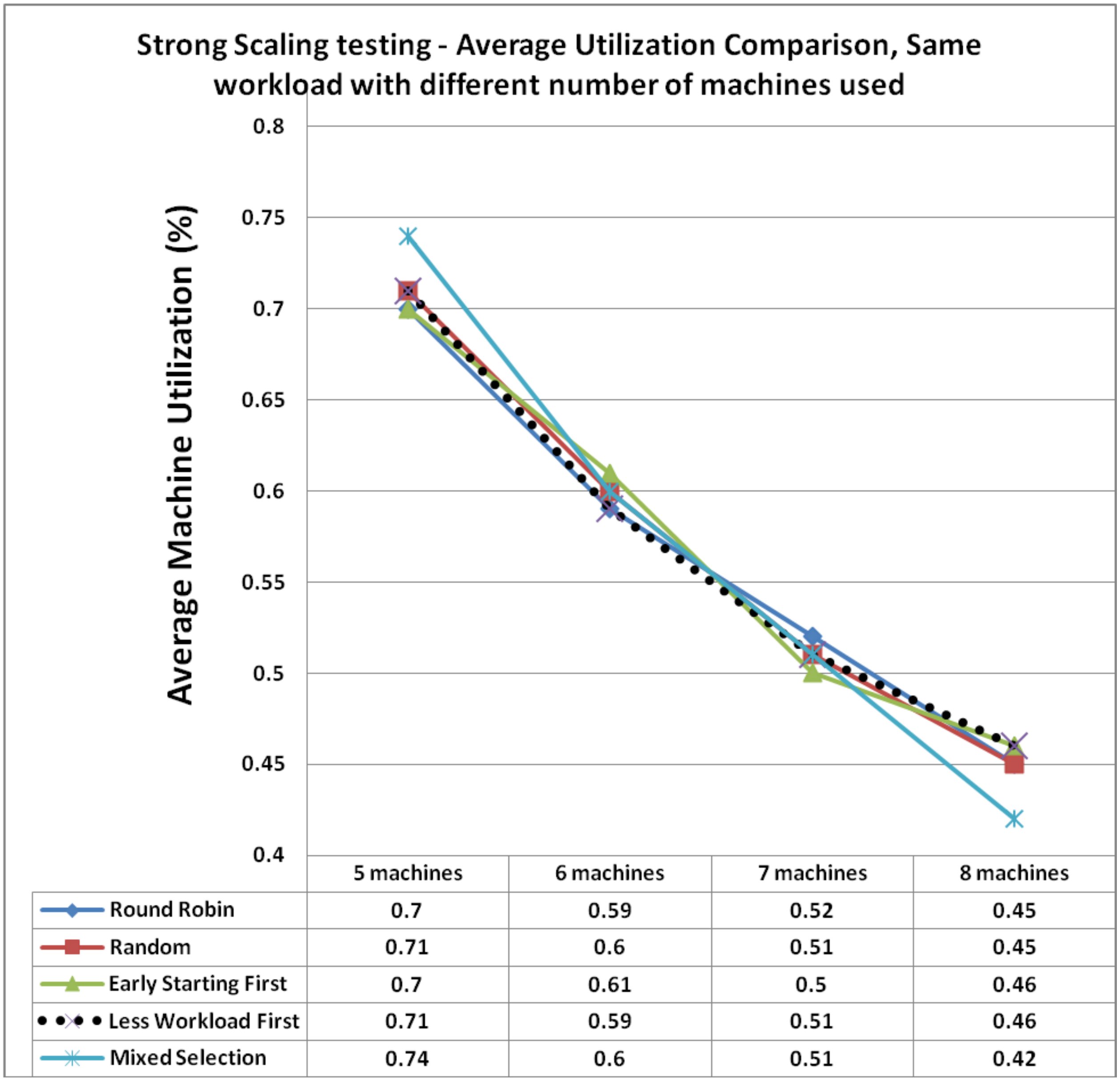
Figure 5-3 shows the result of average turnaround time comparison. This result shows that every scheduling can reduce the average turnaround time when more machines are added to the simulation. The Early Start-Time First heuristic can obtain the lower average turnaround time.



	5 machines	6 machines	7 machines	8 machines
Round Robin	518.6	426.8	404.9	374.4
Random	867	735.2	593.1	541.5
Early Starting First	397.5	379.8	358.2	363.3
Less Workload First	423.8	382.2	365.6	373.4
Mixed Selection	550	474.6	429.5	430.1

Figure 5-3: Strong scaling testing and average turnaround time comparison, same workload with different number of machines

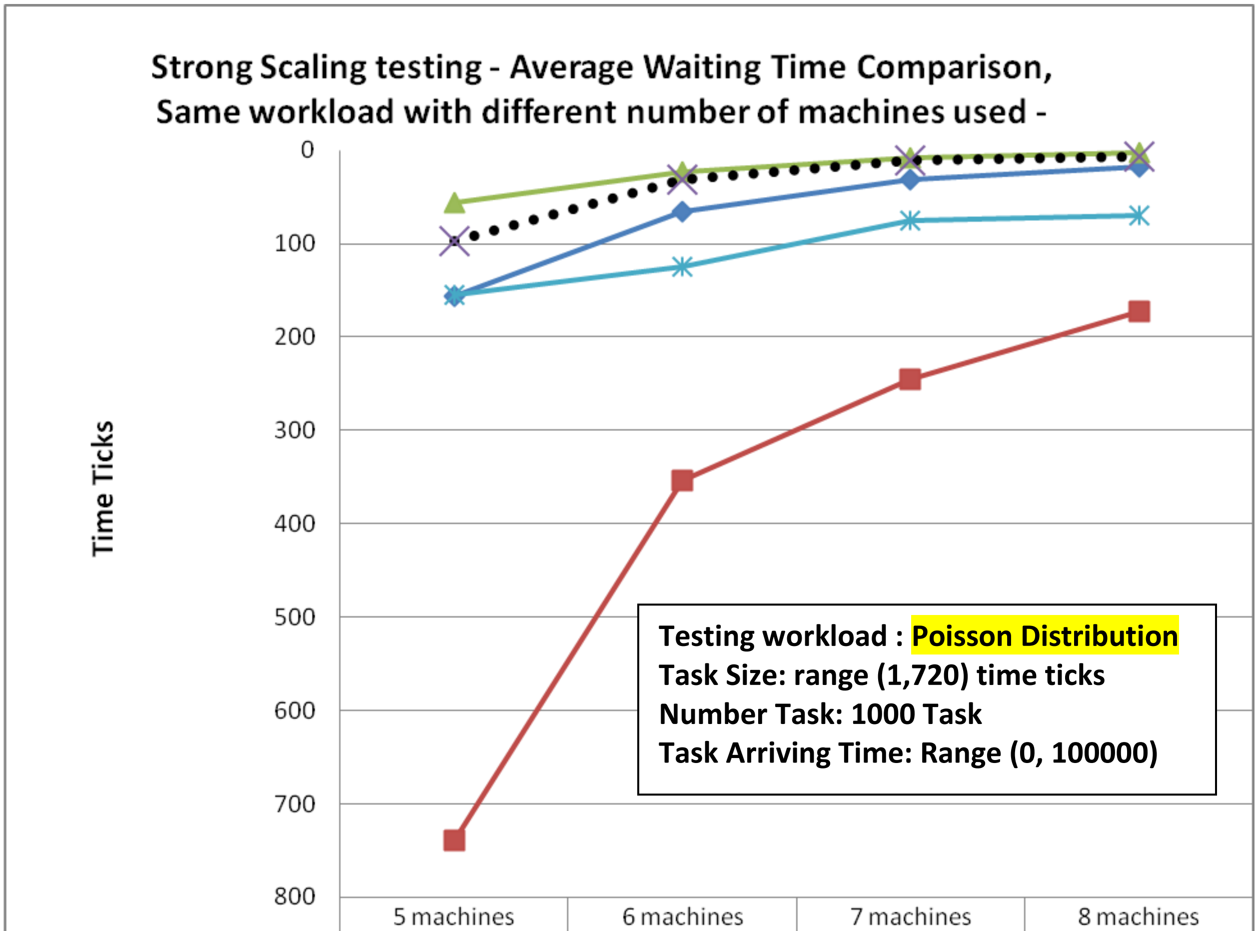
Figure-5-4 shows the average machine utilization in string scaling testing case. We observed that there is not a large variation among all five scheduling methods. It seems that the strong scaling testing cannot clearly help us pinpoint the pros and cons of each scheduling heuristic.



	5 machines	6 machines	7 machines	8 machines
Round Robin	0.7	0.59	0.52	0.45
Random	0.71	0.6	0.51	0.45
Early Starting First	0.7	0.61	0.5	0.46
Less Workload First	0.71	0.59	0.51	0.46
Mixed Selection	0.74	0.6	0.51	0.42

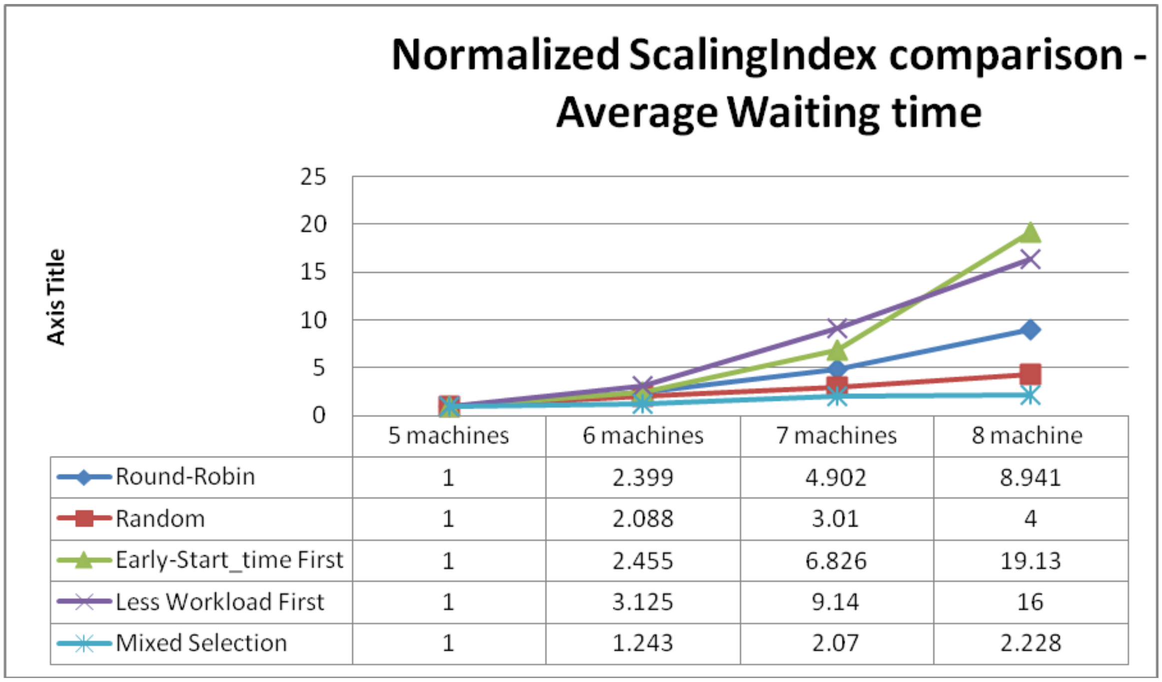
Figure 5-4: Strong scaling testing and average utilization comparison, same workload with different number of machines

We also used the Poisson-random-number generator and repeated the strong scaling testing above. We present results in Figure 6-1, Figure 6-2, Figure 6-3, and Figure 6-4. Our results has show that there is no much difference of using the Normal random number generator and the Poisson random number generator provide in the NetLogo simulation system. The only big difference is that the Less Workload First has a better normalized ScalingIndex when using eight machines.



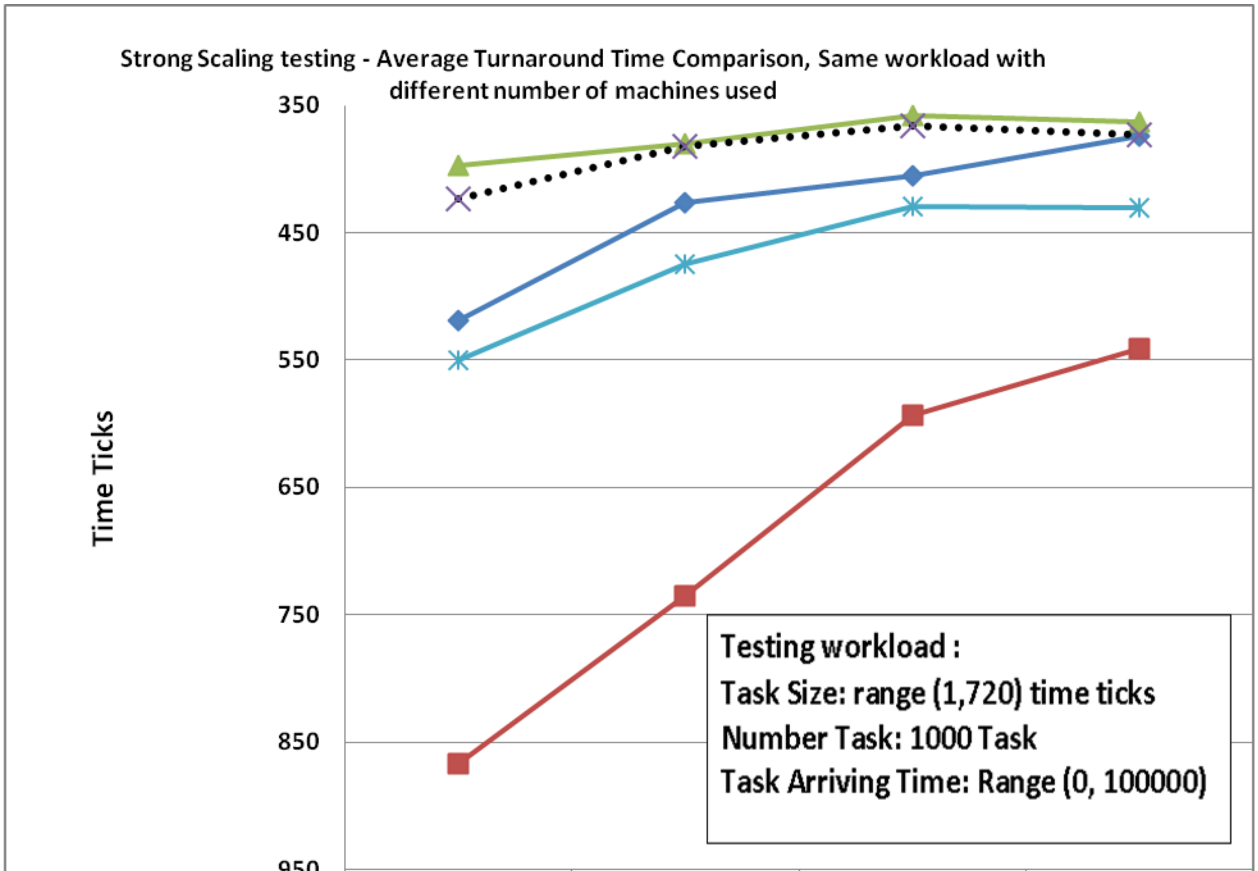
	5 machines	6 machines	7 machines	8 machines
Round Robin	156.4	65.1	31.9	17
Random	739.1	353.9	245.5	172.6
Early Starting First	55.5	22.6	8.13	2.9
Less Workload First	97.8	31.3	10.7	6
Mixed Selection	155.1	124.8	74.9	69.6

Figure 6-1: Strong scaling testing and average waiting time comparison, same workload with different number of machines



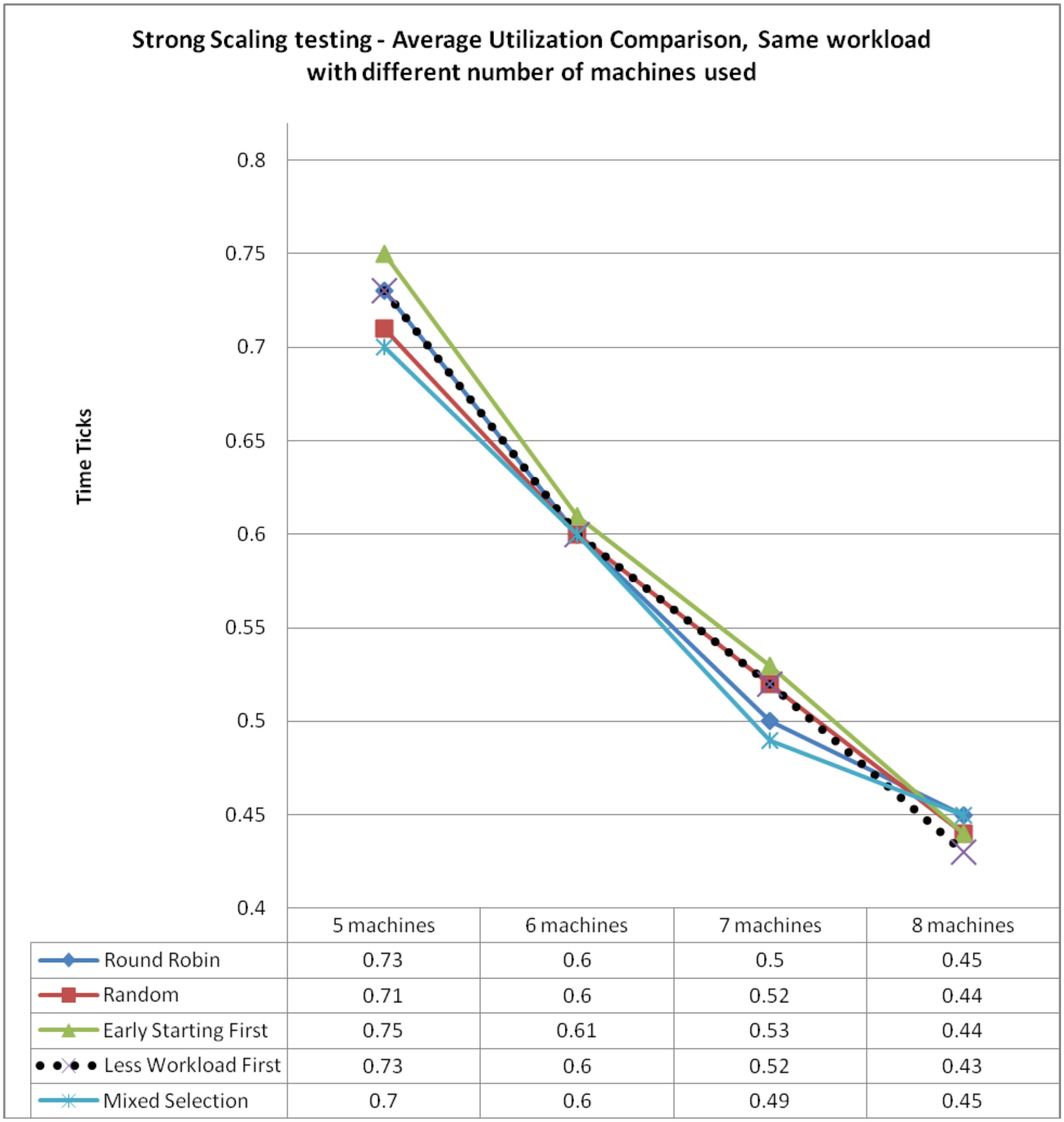
	5 machines	6 machines	7 machines	8 machine
Round-Robin	1	2.399	4.902	8.941
Random	1	2.088	3.01	4
Early-Start_time First	1	2.455	6.826	19.13
Less Workload First	1	3.125	9.14	16
Mixed Selection	1	1.243	2.07	2.228

Figure 6-2: Normalized ScalingIndex comparison - average waiting time



	5 machines	6 machines	7 machines	8 machines
Round Robin	518.6	426.8	404.9	374.4
Random	867	735.2	593.1	541.5
Early Starting First	397.5	379.8	358.2	363.2
Less Workload First	423.8	382.2	365.6	373.4
Mixed Selection	550	474.6	429.5	430.1

Figure 6-3: Strong scaling testing and average turnaround time comparison, same workload with different number of machines



	5 machines	6 machines	7 machines	8 machines
Round Robin	0.73	0.6	0.5	0.45
Random	0.71	0.6	0.52	0.44
Early Starting First	0.75	0.61	0.53	0.44
Less Workload First	0.73	0.6	0.52	0.43
Mixed Selection	0.7	0.6	0.49	0.45

Figure 6-4: Strong scaling testing and average utilization comparison, same workload with different number of machines

3.2.2 Weak scaling testing cases

We also conducted weak scaling testing cases. In a weak scaling testing case the problem size (workload) assigned to each machine stays constant. Table-2 shows the workload distribution when using different number of machines.

Number of machine used	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Workload(#task agent used)	**400**	**800**	**1200**	**1600**	**2000**	**2400**	**2800**	**3200**

Table-2: Workload distribution for weak scaling testing

Figure 7-1 shows that the Early Start-time First and the Less Workload First methods have better stable/static decreasing average waiting time when increasing machines in simulation. The random scheduling method shows an up/down or unstable average waiting time when increasing more machine in simulation.

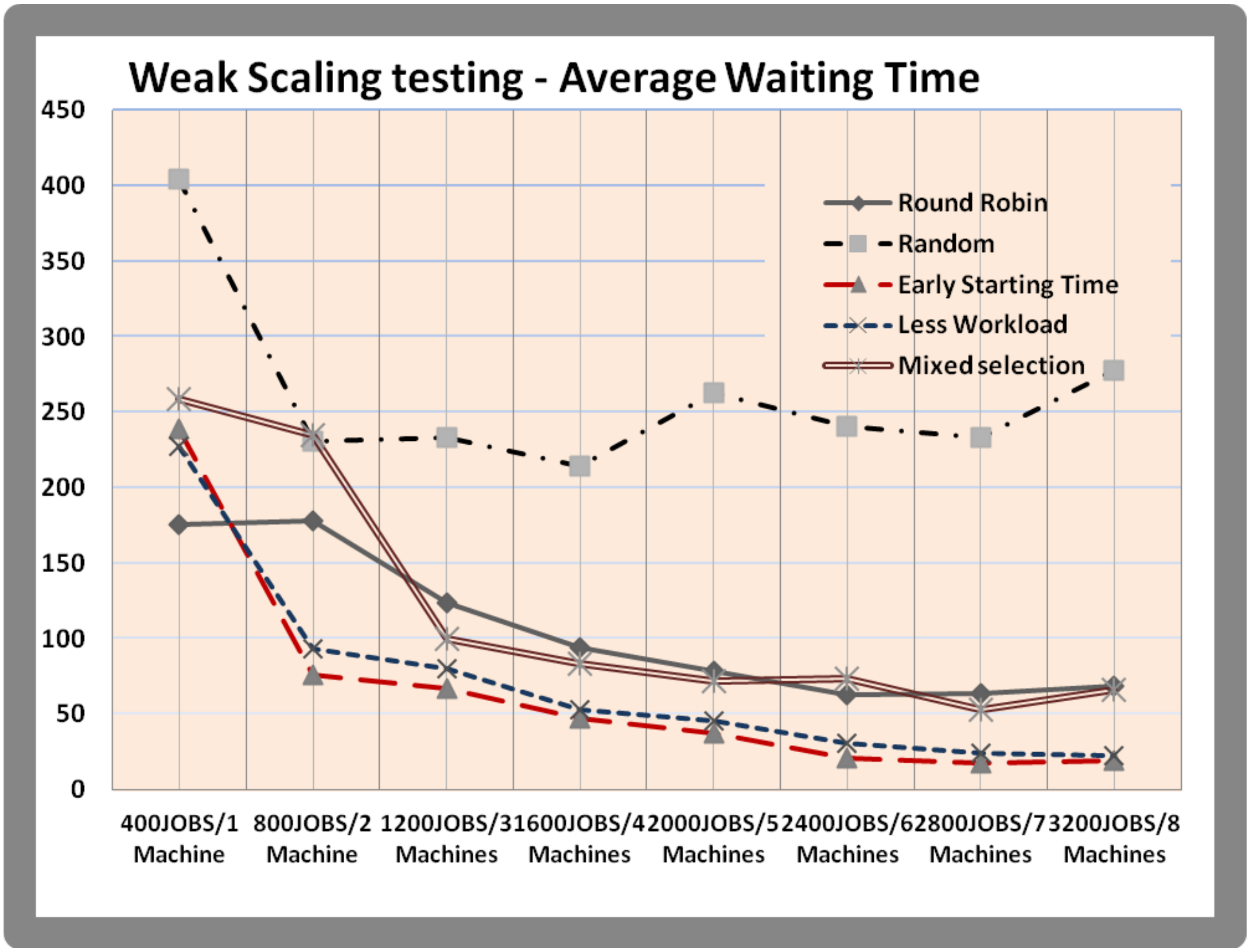


Figure 7-1: Weak scaling testing - Average waiting time comparison

Figure 7-2 shows the result of weak scaling testing cases in terms of average turnaround time comparison. We show that the Early Start-time First and the Less Workload First methods can decrease average waiting time when increasing machines in simulation. The random scheduling method shows an unpredictable.

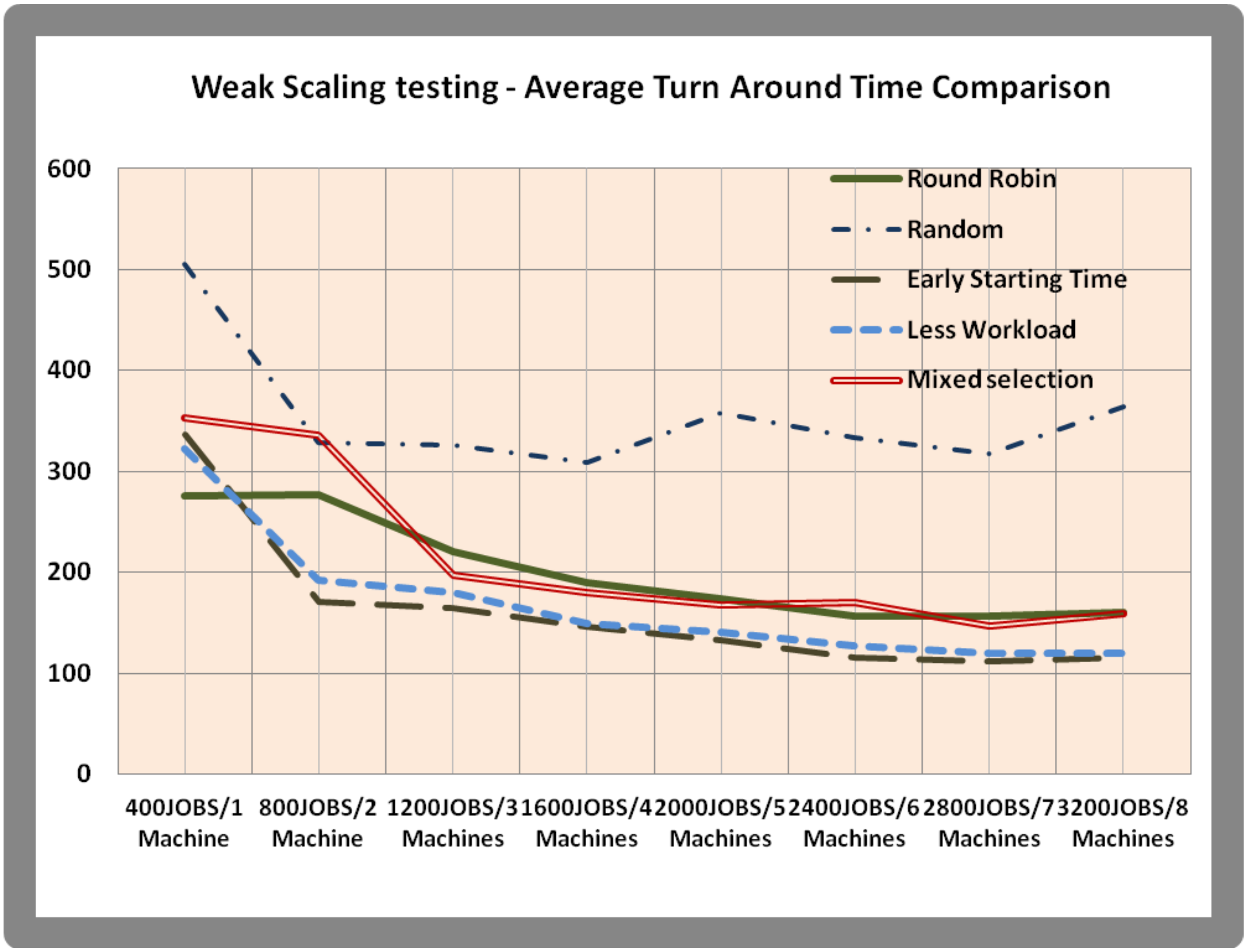


Figure 7-2: Weak Scaling testing - Average turnaround time comparison

3.2.3 Auto tuning feature

We also implemented an "Auto Tuning" feature. We used the "switch" to turn on an off the "Auto Tuning feature." We used the "slider" to set two parameters to control the "auto tuning" feature. They are:

- maxWaitTimeAllow - If the average waiting time is greater than this value, we then add one more machine to the service.
- minWaitTimeAllow - If the average waiting time is below this value, we then remove one machine from the service.

We used this "Auto Tuning" feature to dynamically control the run-time wait-line situation. A sample run-time screen-shot is shown in Figure 8-1, Figure 8-2, and Figure 8-3.

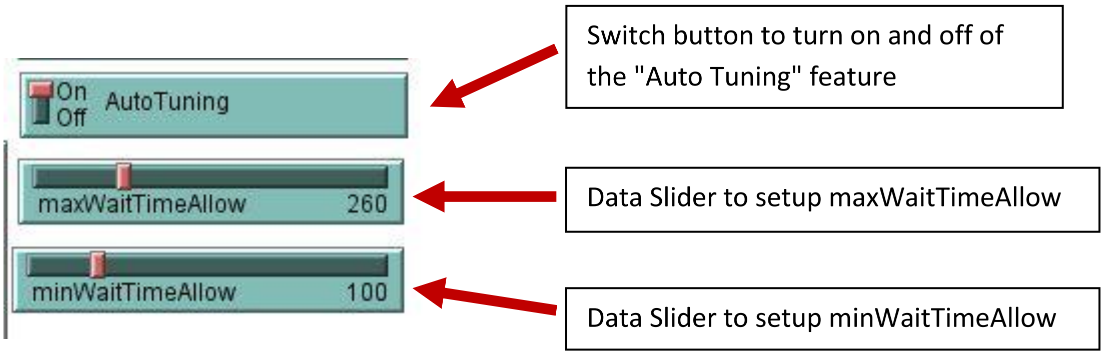


Figure 8-1: Setup: Auto Tunning"

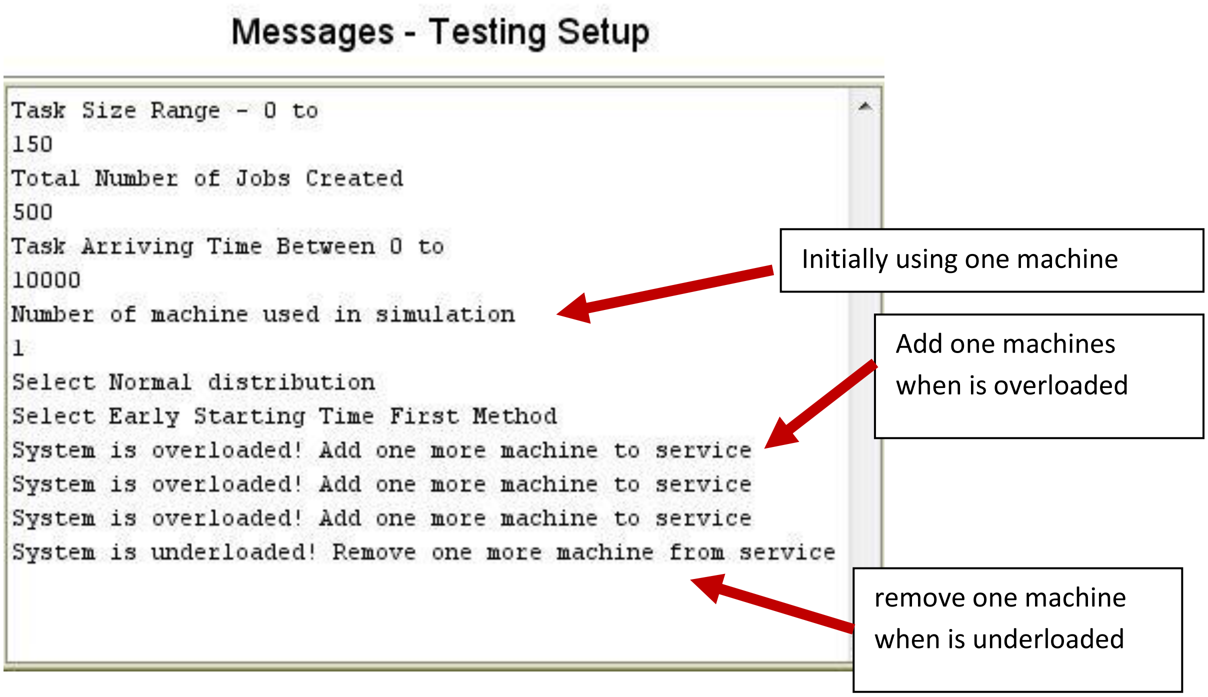


Figure 8-2: Run time messages show activities of the "Auto Tuning"

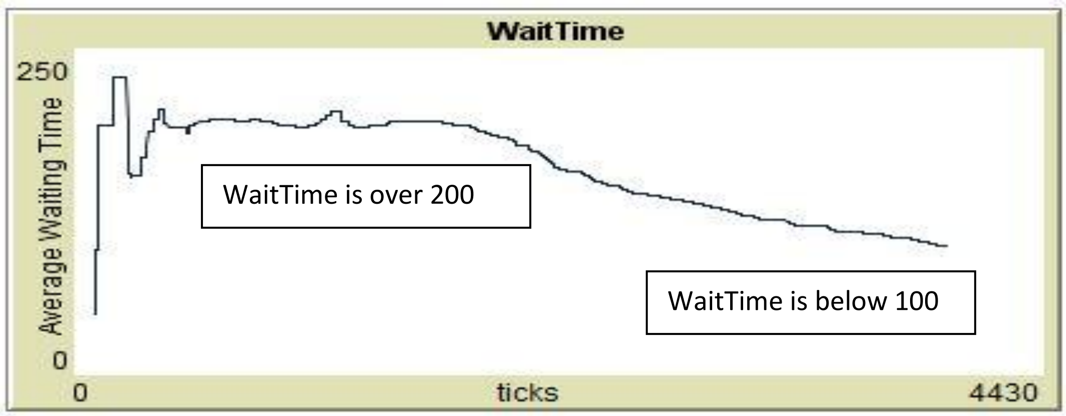


Figure 8-3: AverageWaitTime visualization data over the global tick counter

4 Conclusion

We successfully created a multi-agent task scheduling simulation system using NetLogo programming language and NetLogo run-time environment. We provided an interactive user interface and rich visualization of run-time information. We collected performance data and discuss the pro and con of each heuristic task scheduling methods. We showed that some simple heuristic could help to get better results. The interactive and visualized approach used in the simulation system proves to be useful and interesting. It helps to learn various task patterns such as light workload and heavy workload. It helps to understand performance matrices such as the strong scaling and weak scaling. It demonstrates how difference workload can have different impacts on the waiting time and machine utilization. Using our simulation system, we can conduct testing cases of various real-lives waiting line and queuing problems.

The major contributions of the Team 73 are: (1) the first ever to implement a multi-agent task scheduling simulation program, (2) comparing five heuristic scheduling methods, and (3) implementing a useful educational simulation program that can assist those who wish to study real-life waiting line and queuing problems.

Through this project, we have learned the following things: (1) how to use the NetLogo programming system, (2) what is the waiting line and queuing problem, (3) what is heuristic scheduling method, (4) how to convert a real-life problem to a multi-agent model, (5) how to design interactive user-interface, and (6) how to work together as a team towards a common goal and eventually finish this project on time.

5 Future works

We would like to extend our current simulation program to cover more real-life problems involving waiting line and queuing. There are several interesting areas that we plan to add to or modify our existing simulation program including Real-time task scheduling problem, Dependent task scheduling problem, Work Flow simulation, Multi-server and multi-line task scheduling, Add multiple phases to the existing simulation, and more interactive run-time visualization display such as Gantt chart etc. The initial implementation of the "Auto Tuning" feature is demonstrating our future enhancement plan to provide an intelligent scheduling environment for waiting line queue problems.

Acknowledgements

First, we would like to thank the Super-computing Challenging program and the committee members. The kickoff program at New Mexico Tech was especially helpful. Through this training, we gained knowledge, useful information, and a variety of idea to implement in our simulation program. Furthermore, we would like to thank the comments and suggestions from our intern reviewing judges. We would also like thank our Supercomputing challenge project teacher Mrs. Pauline Stephens for sponsoring our project and her constant encouragement throughout these in the past five months. In addition, we also would like to thank Ariel Chen for reviewing our report and HB Chen for mentoring our team.

Finally we would like to thank our parents for helping us prepare posters, editing the final report, and setting up different testing environments.

Bibliography and References

NetLogo from North Western University - http://ccl.northwestern.edu/netlogo/

NetLogo 5.0 User manual - North Western University

Seth Tisue and Uri Wilensky , "NetLogo: A Simple Environment for Modeling Complexity," ICCS 2004

Frederic Haziza, "Scheduling Algorithms," Department of Computer Science, Uppsala University

Terence C. Ahern, "Bridging the Gap: Cognitive Scaffolding to Improve Computer Programming for Middle School Teachers," 39th ASEE/IEEE Frontier in Education Conference, October 18-21, 2009, San Antonio, TX

E.O. Oyetunji, "Some Common Performance Measures in Scheduling Problems: Review Article," Research Journal of Applied Sciences, Engineering and Technology 1(2): 6-9, 2009

"Process Scheduling", class note, Department of Computer Science, University of Texas at Austin, Professor Lorenzo Alvisi

Sartaj K. Sahni, "Algorithms for Scheduling Independent Tasks,", Journal of ACM, Vol 23, Issues 1, Jan., 1976

Yiqiu Fang, Fei Wang and Junwei Ge, "A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing," Lecture Notes in Computer Science, 2010, Volume 6318/2010, 271-277

Yingzi Li, Xiaodong Zhang, and Shou Zhang, "Multi-agent Simulation System Study on Product Development Process,", Applied Mathematics and Information Science, 2011

Sebastien Paquest, Nicolas Bernier, and Brahim Chaib-Dras, "Multiagent System Viewed as Distributed Scheduling Systems: Methodology and Experiments," Proceedings of the 18th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2005, Victoria, Canada, may 2005

Waiting line and Queuing problem - Queuing theory:
http://businessmanagementcourses.org/Lesson21QueuingTheory.pdf

Appendix - NetLogo Simulation Source code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Modeling Task Scheduling Problems using Multi-Agent Based Simulation System - a heurictic
;; approaches
;;
;; Steven Chen and Andrew Tang
;; Los Alamos Middle School
;; Los Alamos, New Mexico 87544, USA
;; 04/2012
;;
;; This NetLogo simulation project is for the 2012 New Mexico Supercomputing Challenge
;;
;; Using NetLogo release 4.1.3 &  NetLogo 5.0.1
;;
;;
;; Task scheduling problem is a hard problem. Where an exhaustive search for an optimal
;; solution is impractical, we choose heuristic methods to speed up the process of finding a
;; satisfactory solution.
;;
;; We have implemented five different heuristic scheduling methods
;; 1 - Round-Robin : This is a fair-shared approach. We let each machine take turn to
;;              receive an arriving task.    Round-robin scheduling is simple, easy to
;;              implement, and starvation-free.
;;              The name of the algorithm comes from the round-robin principle known from
;;              other fields, where each machine takes an equal share of responsibility to
;;              run a task in turn.
;;
;; 2 - Random Selection : A randomized algorithm is an algorithm which employs a degree of
;;              randomness as part of its logic. We use the builtin random number generator
;;              and decice when machine to run an arriving task.
;;              it may not give us a good solution but it is a good indicator when we
;;              do the compariatvie studies.
;;
;; 3 - Less Workload First -
;;              Each machine keep the total task ececution time assgiend to it.
;;              We pick a machine with the least amount total task execution time.
;;              We only focus on a specific and continuous time period so tbis heuristic
;;              make sense in this simulation. We also expect to a good performance from this
;;              heuristic scheduling method.
;;
;; 4 - Early Starting Time First - We can use two different distribution to define a task arrining time, the
;;              random (normal distribution) and the randon-poisson (Poisson distribution)
;;              We think that we can use the early available to start a task on a machine as the
;;              machine selection index.  We pick a machine the early starting time to run a
;;              task and assign this task to it. We think that this is a reasonable heuristic method. We
;;              implemented it in our simulation.
;;
;; 5 - A Mixed Selection - We then come out this interesting heuristic method.
;;              We plan to implement four different heuristic scheduling methods. Why not try a
;;              combination approach to select an available machine.
;;              the idea is to use a random number to decide which method should be used to pick
;;              a machine.
;;              It is an interesting "demo" to see how good is this mixed method.
```

```
;;              We called it a heuristic of heuristics.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; We have defined four different agents
;; Task agent - many up to 99999
;; Scheduler agent - one scheduler agent
;; Machine agent - eight machine agents, can be more
;; Decos agent - for random display and decoration only
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Task agent:  generate task information -
;;          task arriving time, task execution time
;; each task agent sends a task to the scheduler and asks
;; for a machine to run it
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
breed [tasks taskA]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Machine agent
;; each machine keeps its current machine status and waits for
;; a request from the scheduer to run aa assigned taks
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
breed [machines machine]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Scheduler agent
;; check is there a arrriving task and find an available
;; machine ro run it
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
breed [schedulers scheduler]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Decos agent
;; this is for decoration only
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
breed [decos deco]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Global variables, can be accessed by all agents
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
globals [
  totalMachineReserved
  numMachineOccupied
  numMachineAvailable
  waitingTime
  idleTime
  averageIdleTime
  arrivingTime
  taskExecTime
  finishedJobs
  lengthJob
  markSpan
  numJobCreated
```

```
numJobFinished
totalJobs
totalTime
wallClock
throughPut
averageTurnAroundTime
totalTurnAroundTime
utilization
averageResponseTime
averageWaitTime
averageTaskLengthTask
numJobArrived
numJobWaiting

autoTuneTickCount
nextautoTuneTickCount

FlagAutoTune
;; Load Balance Performance Index
LBPI1
LBPI2
LBPI3
LBPI4

;; machine workload information
machWorkLoad1
machWorkLoad2
machWorkLoad3
machWorkLoad4
machWorkLoad5
machWorkLoad6
machWorkLoad7
machWorkLoad8
machWorkLoad9
machWorkLoad10
machWorkLoad11
machWorkLoad12
machWorkLoad13
machWorkLoad14
machWorkLoad15
machWorkLoad16

lessWorkloadMachineID
averageTaskLengthMachine

;; performance data
maxStartTime

minStartTime
maxFinishTime
minFinishTime
maxThroughput
minThroughput
maxTaskLength
minTaskLength
maxUtilization
```

```
  minUtilization
  maxStartTime2
  minStartTime2
  maxFinishTime2
  minFinishTime2
  maxThroughput2
  minThroughput2
  maxTaskLength2
  minTaskLength2
  maxUtilization2
  minUtilization2
  loadBalanceIndex
  maxJobWait
  minJobWait
  maxJobFinish
  minJobFinish
  maxJobWait2
  minJobWait2
  maxJobFinish2
  minJobFinish2
  jobPatterns
  numJobTick
  throughPutTimeTick
  selectedMethod

]


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; only task agent can access its own variables
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
tasks-own [
   taskID
   taskLength
   arriveTime
   startTime
   finishTime
   waitTime
   markSpanTime
   assignedMachine
   taskDone
   taskWaiting
   taskExecution
   taskFinish
   turnAroundTime
]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; only the scheduler agent can access these variables
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
schedulers-own [
 schedulerID
 numJobArrive
 numJobStart
 numJobFinish
 numJobWait
```

```
 currentmachineID
]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; each machine can access its own variables
;; there is no array ro multi-dimension data structure
;; but this is useful. Each agent can define its own variables
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
machines-own [
 machineID
 numJobArrive
 numJobStart
 numJobFinish
 numJobWait
 waitTimeM
 idleTimeM
 currentStartTime
 nextAvailTime
 totalTaskTime
 currentTaskID
 utilizationM
 totalTurnAroundTimeM
 averageTurnAroundTimeM
 throughputM

]



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; the "setup" button"
;; use this "setup" button before run each simulation
;; after the "setup" also you need to select distribution
;; method and scheduling method
;; default is set to
;; normal distribution and round-robin scheduling
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to setup
  ;; (for this model to work with NetLogo's new plotting features,
  ;; __clear-all-and-reset-ticks should be replaced with clear-all at
  ;; the beginning of your setup procedure and reset-ticks at the end
  ;; of the procedure.)
  __clear-all-and-reset-ticks ;; this will clean out all button variables and plot areas

  init-globals  ;; initialize global variables

  ;;  create one scheduler
  ;; turtle 0 :  schedule
  create-schedulers 1

  ;;  create eight machines
  ;; turtle 1-8: machines
  create-machines 8

  ;; create two watchdog turtles for some extra activities
  ;;  turtle ID 9 adn 10
```

```
  cro 2

  ;; create task list
  ;; we used 100 tasks here for
  ;; it can be change to different number later
  ;; ID 11 - 110 remap to task ID 1 - 100
  create-tasks maxNumJobs
  set  numJobCreated maxNumJobs

  create-decos 100 ;; decoration only

  ;; setup basic information for agents
  setup-scheduler
  setup-machines
  setup-task-info

  output-print "Task Size Range - 0 to "
  output-print maxTaskSize
  output-print "Total Number of Jobs Created"
  output-print maxNumJobs
  output-print "Task Arriving Time Between 0 to "
  output-print maxJobArriveTime
  output-print "Number of machine used in simulation"
  output-print numMachine

  ;; reset the tick counter value to zero
  reset-ticks
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; setup schedule basic information
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to setup-scheduler
  ask scheduler 0 [
    ;;print "scheduler setup done"
    set schedulerID  0
    set numJobArrive 0
    set numJobStart 0
    set numJobFinish 0
    set numJobWait 0
    set currentMachineID 0
  ]

  ask patches [
    set pcolor green
  ]

end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; initialize global variables
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to init-globals
  ;;set numMachine 8
```

```
set totalMachineReserved 8
set numMachineOccupied 0
set numMachineAvailable 0
set waitingTime 0
set idleTime 0
set averageIdleTime 0
set arrivingTime 0
set taskExecTime 0
set finishedJobs 0
set lengthJob 0
set markSpan 0
set totalJobs 0
set numJobFinished 0
set totalTime 0
set wallClock 0
set throughPut 0
set averageTurnAroundTime 0
set totalTurnAroundTime  0
set utilization 0
set averageResponseTime 0
set averageWaitTime 0
set numJobArrived 0
set numJobFinished 0
set numJobWaiting 0

set machWorkLoad1 0
set machWorkLoad2 0
set machWorkLoad3 0
set machWorkLoad4 0
set machWorkLoad5 0
set machWorkLoad6 0
set machWorkLoad7 0
set machWorkLoad8 0
set machWorkLoad9 0
set machWorkLoad10 0
set machWorkLoad11 0
set machWorkLoad12 0
set machWorkLoad13 0
set machWorkLoad14 0
set machWorkLoad15 0
set machWorkLoad16 0


set throughPutTimeTick 0


set lessWorkloadMachineID 0

set maxStartTime 0
set minStartTime 0
set maxFinishTime 0
set minFinishTime 0
set maxThroughput 0
set minThroughput 0
set maxTaskLength 0
set minTaskLength 0
```

```
  set maxUtilization 0
  set minUtilization 0
  set maxJobWait 0
  set minJobWait 0

  set maxJobFinish 0
  set minJobFinish 0


  set maxStartTime2 0
  set minStartTime2 0
  set maxFinishTime2 0
  set minFinishTime2 0
  set maxThroughput2 0
  set minThroughput2 0
  set maxTaskLength2 0
  set minTaskLength2 0
  set maxUtilization2 0
  set minUtilization2 0

  set maxJobWait2 0
  set minJobWait2 0

  set maxJobFinish2 0
  set minJobFinish2 0
  set autoTuneTickCount 0
  set FlagAutoTune 1
  set LBPI1 0
  set LBPI2 0
  set LBPI3 0
  set LBPI4 0

  set numJobTick 0
  set selectedMethod 1
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; setup resource machine infor
;;; we have used 8 machines in this simulation
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to setup-machines
  ask machine 1 [
    set machineID 1
    set numJobArrive 0
    set numJobStart 0
    set numJobFinish 0
    set numJobWait 0
    set currentStartTime 0
    set nextAvailTime 0
    set currentTaskID 0
    set idleTimeM 0
    set waitTimeM 0
    set utilizationM 0
    set totalTaskTime 0
    set averageTurnAroundTimeM 0
```

```
 set totalTurnAroundTimeM 0
]

ask machine 2 [
 set machineID 2
 set numJobArrive 0
 set numJobStart 0
 set numJobFinish 0
 set numJobWait 0
 set currentStartTime 0
 set nextAvailTime 0
 set currentTaskID 0
 set idleTimeM 0
 set waitTimeM 0
 set utilizationM 0
 set totalTaskTime 0
 set averageTurnAroundTimeM 0
 set totalTurnAroundTimeM 0
]

ask machine 3 [
 set machineID 3
 set numJobArrive 0
 set numJobStart 0
 set numJobFinish 0
 set numJobWait 0
 set currentStartTime 0
 set nextAvailTime 0
 set currentTaskID 0
 set idleTimeM 0
 set waitTimeM 0
 set utilizationM 0
 set totalTaskTime 0
 set averageTurnAroundTimeM 0
 set totalTurnAroundTimeM 0
]

ask machine 4 [
 set machineID 4
 set numJobArrive 0
 set numJobStart 0
 set numJobFinish 0
 set numJobWait 0
 set currentStartTime 0
 set nextAvailTime 0
 set currentTaskID 0
 set idleTimeM 0
 set waitTimeM 0
 set utilizationM 0
 set totalTaskTime 0
 set averageTurnAroundTimeM 0
 set totalTurnAroundTimeM 0
]

ask machine 5 [
 set machineID 5
```

```
  set numJobArrive 0
  set numJobStart 0
  set numJobFinish 0
  set numJobWait 0
  set currentStartTime 0
  set nextAvailTime 0
  set currentTaskID 0
  set idleTimeM 0
  set waitTimeM 0
  set utilizationM 0
  set totalTaskTime 0
  set averageTurnAroundTimeM 0
  set totalTurnAroundTimeM 0
]

ask machine 6 [
  set machineID 6
  set numJobArrive 0
  set numJobStart 0
  set numJobFinish 0
  set numJobWait 0
  set currentStartTime 0
  set nextAvailTime 0
  set currentTaskID 0
  set idleTimeM 0
  set waitTimeM 0
  set utilizationM 0
  set totalTaskTime 0
  set averageTurnAroundTimeM 0
  set totalTurnAroundTimeM 0
]

ask machine 7 [
  set machineID 7
  set numJobArrive 0
  set numJobStart 0
  set numJobFinish 0
  set numJobWait 0
  set currentStartTime 0
  set nextAvailTime 0
  set currentTaskID 0
  set idleTimeM 0
  set waitTimeM 0
  set utilizationM 0
  set totalTaskTime 0
  set averageTurnAroundTimeM 0
  set totalTurnAroundTimeM 0
]

ask machine 8 [
  set machineID 8
  set numJobArrive 0
  set numJobStart 0
  set numJobFinish 0
  set numJobWait 0
  set currentStartTime 0
```

```
    set nextAvailTime 0
    set currentTaskID 0
    set idleTimeM 0
    set waitTimeM 0
    set utilizationM 0
    set totalTaskTime 0
    set averageTurnAroundTimeM 0
    set totalTurnAroundTimeM 0
  ]

end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; let each task agent to create a job
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to setup-task-info
    ask tasks [
      ;; Task ID
      set taskID random maxTaskSize
      set tasklength random maxTaskSize
      if taskLength = 0 [
         set taskLength 1
      ]

      ;; task arriving time generated from the selecting
      ;; distribution
      ;; the default is normal distribution
      if jobPatterns = 0 [
         set arriveTime random maxJobArriveTime
      ]

      ;; Poisson distribution is used if you select this one
      if jobPatterns = 1 [
         set arriveTime random-poisson maxJobArriveTime
      ]

      set startTime 0
      set finishTime 0
      set waitTime 0
      set markspanTime 0
      set assignedMachine 0
      set taskDone 0

      set taskWaiting 0
      set taskExecution 0
      set taskFinish 0
      set turnAroundTime 0
      ;;;;print "tasks  setup done"
    ]

end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;; button to select the Normal distribution
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to normalDistribution
  set jobPatterns 0
  output-print "Select Normal distribution"
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; button to select the Poisson distribution
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to poissonDistribution
  set jobPatterns 1
  output-print "Select Poisson distribution"
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; button to select the Round-robin scheduling method
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to roundRobin
  set selectedMethod 1
  output-print "Select Round Robin Method"
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; button to select the Random machine scheduling method
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to randomMethod
    set selectedMethod 2
    output-print "Select Random Selection Method"
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; button to select the Less Workload First scheduling method
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to lessWorkloadFirst
    set selectedMethod 3
    output-print "Select Less Workload First Method"
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; button to select the Early Starting Time First scheduling method
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to earlyStartTimeFirst
    set selectedMethod 4
    output-print "Select Early Starting Time First Method"
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; button to randomly select from the four heuristic methods
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to cockTailMixed
    set selectedMethod 5
    output-print "Select Mixed Heutistic Method"
end
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; get how many tasks has assigned to a machine at this time
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to get-machine-workload-number-of-jobs
  ask machine 1 [
    set machWorkLoad1 numJobArrive
    ;;;;print (word "machWorkLoad1 " machWorkLoad1)
  ]
    ask machine 2 [
    set machWorkLoad2 numJobArrive
    ;;;;print (word "machWorkLoad2 " machWorkLoad2)
  ]
  ask machine 3 [
    set machWorkLoad3 numJobArrive
    ;;;;print (word "machWorkLoad3 " machWorkLoad3)
  ]
  ask machine 4 [
    set machWorkLoad4  numJobArrive
    ;;;;print (word "machWorkLoad4 " machWorkLoad4)
  ]
  ask machine 5 [
    set machWorkLoad5 numJobArrive
    ;;;;print (word "machWorkLoad5 " machWorkLoad5)
  ]
  ask machine 6 [
    set machWorkLoad6 numJobArrive
    ;;;;print (word "machWorkLoad6 " machWorkLoad6)
  ]
  ask machine 7 [
    set machWorkLoad7 numJobArrive
    ;;;;print (word "machWorkLoad7 " machWorkLoad7)
  ]

  ask machine 8 [
    set machWorkLoad8 numJobArrive
    ;;;;print (word "machWorkLoad8 " machWorkLoad8)
  ]



end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; get what is the current early time to start a task on this machine
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to get-machine-early-start-time-first

  ask machine 1 [
    set machWorkLoad1 nextAvailTime
    ;;print (word "ESTF machWorkLoad1 " machWorkLoad1)
  ]

  ask machine 2 [
    set machWorkLoad2 nextAvailTime
```

```
    ;;print (word "ESTF machWorkLoad2 " machWorkLoad2)
  ]

  ask machine 3 [
    set machWorkLoad3 nextAvailTime
    ;;print (word "ESTF machWorkLoad3 " machWorkLoad3)
  ]

  ask machine 4 [
    set machWorkLoad4 nextAvailTime
    ;;print (word "ESTF machWorkLoad4 " machWorkLoad4)
  ]
  ask machine 5 [
    set machWorkLoad5 nextAvailTime
    ;;print (word "ESTF machWorkLoad5 " machWorkLoad5)
  ]

  ask machine 6 [
    set machWorkLoad6 nextAvailTime
    ;;print (word "ESTF machWorkLoad6 " machWorkLoad6)
  ]

  ask machine 7 [
    set machWorkLoad7 nextAvailTime
    ;;print (word "ESTF machWorkLoad7 " machWorkLoad7)
  ]

  ask machine 8 [
    set machWorkLoad8 nextAvailTime
    ;;print (word "ESTF machWorkLoad8 " machWorkLoad8)
  ]
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; get the current workload on each machine
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to get-machine-workload-less-job-first

  ask machine 1 [
    set machWorkLoad1 totalTaskTime
    ;;;;print (word "machWorkLoad1 " machWorkLoad1)
  ]

  ask machine 2 [
    set machWorkLoad2 totalTaskTime
    ;;;;print (word "machWorkLoad2 " machWorkLoad2)
  ]

  ask machine 3 [
    set machWorkLoad3 totalTaskTime
    ;;;;print (word "machWorkLoad3 " machWorkLoad3)
  ]

  ask machine 4 [
    set machWorkLoad4 totalTaskTime
```

```
    ;;;;print (word "machWorkLoad4 " machWorkLoad4)
  ]
  ask machine 5 [
    set machWorkLoad5 totalTaskTime
    ;;;;print (word "machWorkLoad5 " machWorkLoad5)
  ]

  ask machine 6 [
    set machWorkLoad6 totalTaskTime
    ;;;;print (word "machWorkLoad6 " machWorkLoad6)
  ]

  ask machine 7 [
    set machWorkLoad7 totaltaskTime
    ;;;;print (word "machWorkLoad7 " machWorkLoad7)
  ]

  ask machine 8 [
    set machWorkLoad8 totalTaskTime
    ;;;;print (word "machWorkLoad8 " machWorkLoad8)
  ]


end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; get the machineID with the less workload at this time
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to get-less-workload-machine

   let workload 9999999

   if numMachine >= 1 [
     if workload > machWorkLoad1 [
       set workload machWorkLoad1
       set lessWorkloadMachineID 1
     ]
   ]

   if numMachine >= 2 [
     if workload > machWorkLoad2 [
       set workload machWorkLoad2
       set lessWorkloadMachineID 2
     ]
   ]

   if numMachine >= 3 [
     if workload > machWorkLoad3 [
       set workload machWorkLoad3
       set lessWorkloadMachineID 3
     ]
   ]

   if numMachine >= 4 [
     if workload > machWorkLoad4 [
```

```
    set workload machWorkLoad4
    set lessWorkloadMachineID 4
  ]
 ]

 if numMachine >= 5 [
  if workload > machWorkLoad5 [
    set workload machWorkLoad5
    set lessWorkloadMachineID 5
  ]
 ]

 if numMachine >= 6 [
  if workload > machWorkLoad6 [
    set workload machWorkLoad6
    set lessWorkloadMachineID 6
 ]
 ]

 if numMachine >= 7 [
  if workload > machWorkLoad7 [
    set workload machWorkLoad7
    set lessWorkloadMachineID 7
  ]
 ]

 if numMachine >= 8 [
  if workload > machWorkLoad8 [
    set workload machWorkLoad8
    set lessWorkloadMachineID 8
  ]
 ]

end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; get the machine ID taht has the early task starting time
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to get-early-start-time-machine

 let workload 9999999

 if numMachine >= 1 [
  if workload > machWorkLoad1 [
    set workload machWorkLoad1
    set lessWorkloadMachineID 1
  ]
 ]

 if numMachine >= 2 [
  if workload > machWorkLoad2 [
    set workload machWorkLoad2
    set lessWorkloadMachineID 2
  ]
 ]
```

```
  if numMachine >= 3 [
    if workload > machWorkLoad3 [
      set workload machWorkLoad3
      set lessWorkloadMachineID 3
    ]
  ]

  if numMachine >= 4 [
    if workload > machWorkLoad4 [
      set workload machWorkLoad4
      set lessWorkloadMachineID 4
    ]
  ]

  if numMachine >= 5 [
    if workload > machWorkLoad5 [
      set workload machWorkLoad5
      set lessWorkloadMachineID 5
    ]
  ]

  if numMachine >= 6 [
    if workload > machWorkLoad6 [
      set workload machWorkLoad6
      set lessWorkloadMachineID 6
  ]
  ]

  if numMachine >= 7 [
    if workload > machWorkLoad7 [
      set workload machWorkLoad7
      set lessWorkloadMachineID 7
    ]
  ]

  if numMachine >= 8 [
    if workload > machWorkLoad8 [
      set workload machWorkLoad8
      set lessWorkloadMachineID 8
    ]
  ]


 ;;;print (word "select lessworkload machine " lessWorkloadMachineID )

end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; randomly select a machine
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to get-random-machine

  let LnumMachine numMachine + 1
```

```
  set lessWorkloadMachineID random LnumMachine
  if lessWorkloadMachineID = 0 [
    set lessWorkloadMachineID random LnumMachine
    if lessWorkloadMachineID = 0 [
       set lessWorkloadMachineID random LnumMachine
       if lessWorkloadMachineID = 0 [
         set lessWorkloadMachineID random LnumMachine
         if lessWorkloadMachineID = 0 [
           set lessWorkloadMachineID 1
         ]
       ]
    ]
  ]

end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; use the rounc-robin heuristic to select a machine
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to get-round-robin-machine
  ;;this is Round-robin scheduling heuristic
  if currentMachineID = numMachine [
    set currentMachineID 1
  ]
  if currentMachineID < numMachine [
    set currentMachineID (currentMachineID + 1)
  ]

end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; button - Add a machine during the run time
;; this is an usefuly interactive feature of using the NetLogo
;; we can dynamically add some machine(s) to the task simulation
;; when we find the system is overloaded we can add machines to
;; reduce the task's waiting time and increase the overall task thoughput
;; this is an interesting "demo" feture" to learn the task scheduling
;; concept in operating system
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to AddMachine
   if numMachine < 8 [
     set numMachine numMachine + 1
   ]
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; button - Remove a machine during the run time
;; this is an usefuly interactive feature of using the NetLogo
;; we can dynamically remove some machine(s) to the task simulation
;; when we find the system is underloaded we can remove machines to
;; save system resouece and still maintain a reasonable performance such as
;; waiting time, throughput, and system utilization
```

```
;; this is an interesting "demo" feture" to learn the task scheduling
;; concept in operating system
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
to RemoveMachine
   if numMachine >= 2 [
     set numMachine numMachine - 1
   ]
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; main control procedures
;; the "go" routine is set as "forever"
;; it will repeat the same executing sequence again and again
;; for each iteration we advance a tick acount to remembet the current
;; wall-clock information
;; the tick counter is used as the wall-clock.
;; we check the tick-count with task arriving time
;; if there is a match we should schedule this task ASAP.
;; becasue of this tick count , we then can generate performance data
;; such as task starting time, task finishing time, throughout, utilization,
;; waiting time...
;; we aslo "setup" a simulation-stop condition
;; that is when the create task number is equal to the number of task scheduled
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to go

  ;; LOCAL variables
  let taskScheduled 0
  let L2MachineID 0
  let LMachineID 0
  let idleTimeT 0
  let waitTimeT 0
  let LtaskLength 0
  let LtaskID 0
  let LstartTime 0
  let LfinishTime 0
  let LarriveTime 0
  let LpreviousStartTime 0
  let LmixedMethod 0
  set numJobTick  0
  let LselectedMethod 0
  set LselectedMethod selectedMethod
  let LrrPicked  0
  let LwaitTime 0
  let LidleTime 0
  let LturnAroundTime 0

  ;;output-print "num of task agetns "
  ;;output-print maxTaskSize

  ;; this is a "simulation-stop" condition checking
  ;; we stop the simuation when all created tasks are scheduled
```

```
if  numJobCreated = totalJobs [
  ;;print (word "All generaated task are scheduled")

  if maxFinishTime2 < ticks [
     print (word "All generated tasks are finishing execution ")
     print (word "Simulation End")
     stop

  ]

]




;; Check if there is an arriving task from task agent
;; only the un-scheduled task is picked
;;


;; ***********************************************************************************

ask tasks with [taskDone = 0] [
  ;;  only schedule an available taks that is
  ;; arriving time is equal to the wall clock
  ;;
  if arriveTime = ticks [  ;; am I ready to be scheduled

    set-current-plot "Task Distribution"
    plotxy ticks taskLength

    set  numJobTick  numJobTick + 1
    set-current-plot "Task Arriving Patterns"
    plotxy ticks numJobTick

    ;;print (word "tick " ticks " task arrive time " arriveTime  " length "tasklength)

    set taskScheduled 1
    set totalJobs totalJobs + 1
    set LtaskLength taskLength
    set LtaskID taskID
    set LarriveTime arriveTime


    ask scheduler 0 [ ;; Only one scheduler is using here

      ;; STEP 001: Selection an available machine based on the
      ;; selected scheduling method

      ;; user can select a "scheduling method from the button"
      ;; I have implemented five different heuristic scheduling methods
      ;; 1 - round-robin
      ;; 2 - random slection
      ;; 3 - lett workload first
      ;; 4 - early starting  time first
```

```
;; 5 - a mixed selection of the above four heuristic method
;;

;; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;; First method: round robin or fail shared method
if LselectedMethod = 1  [
  ;;get-round-robin-machine
  if currentMachineID = numMachine [
    set currentMachineID 1
    ;;;;print (word "HBHBHB 001 >>>>  " currentMachineID)
    set LrrPicked 1
  ]

  if currentMachineID < numMachine [
     if LrrPicked = 0 [
        set currentMachineID currentMachineID + 1
        ;;;;print (word "HBHBHB 002 >>>>  " currentMachineID)
     ]
  ]

  if LrrPicked = 1 [
     set LrrPicked 0
  ]
]

;; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;; Second method: randomly selecting a machine to run this task
if LselectedMethod = 2  [
  get-random-machine
  set currentMachineID lessWorkloadMachineID
]
;; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;; Third method: Select a machei with less accumulated workload
if LselectedMethod = 3  [
   get-machine-workload-less-job-first
   get-less-workload-machine
   set currentMachineID lessWorkloadMachineID
]
;; +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


;; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;; Fourth method: Select a machienthe with early starting time that can run this taks
if LselectedMethod = 4  [
   get-machine-early-start-time-first
   get-early-start-time-machine
   set currentMachineID lessWorkloadMachineID
]
;; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;; Fifth method: This is a mixed selection fo the above four heuristice methods
;; this is an experiment
if LselectedMethod = 5  [
```

```
  ;; use a random number to decide which method is going to select the next machine
  set LmixedMethod random 4
  if LmixedMethod = 0  [
    ;;get-round-robin-machine
    if currentMachineID = numMachine [
      set currentMachineID 1
      set LrrPicked 1
    ]
    if currentMachineID < numMachine [
      if LrrPicked = 0 [
        set currentMachineID currentMachineID + 1
      ]
    ]
    if LrrPicked = 1 [
      set LrrPicked 0
    ]
  ]

  if LmixedMethod = 1  [
    get-random-machine
    set currentMachineID lessWorkloadMachineID
  ]

  if LmixedMethod = 2  [
    get-machine-workload-less-job-first
    get-less-workload-machine
    set currentMachineID lessWorkloadMachineID
   ]

  if LmixedMethod = 3  [
    get-machine-early-start-time-first
    get-early-start-time-machine
    set currentMachineID lessWorkloadMachineID
  ]
]
;; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;; set local variables
set LMachineID currentMachineID
set L2MachineID currentMachineID
let MID currentMachineID

;; STEP 002: Assign this task to this selected machine
;;     and update information of this machine
ask machine currentMachineID [
  set numJobArrive numJobArrive + 1
  set numJobStart numJobStart + 1
  set numJobFinish numJobFinish + 1
  set totalTaskTime totalTaskTime + LtaskLength
  set LpreviousStartTime currentStartTime

  if nextAvailTime < LarriveTime [
    ;;print "nextAvailTime < LarriveTime ---------------"
    set currentStartTime LarriveTime
    set LidleTime LarriveTime - nextAvailTime
    set idleTimeM idleTimeM + LidleTime
```

```
          ;;print (word "0002A Machine currentStartTime " currentStartTime " task arrive time " LarriveTime " nextAvailTime " nextAvailTime)
        ]

        if nextAvailTime > LarriveTime [
          ;;print "nextAvailTime < LarriveTime ---------------"
          set currentStartTime nextAvailTime
          set LwaitTime nextAvailTime - LarriveTime
          set waitTimeM waitTimeM + LwaitTime
          ;;print (word "0002B Machine currentStartTime " currentStartTime " task arrive time " LarriveTime " nextAvailTime " nextAvailTime)
        ]

        set nextAvailTime currentStartTime + LtaskLength
        set LstartTime LpreviousStartTime
        set LfinishTime nextAvailTime
        set currentTaskID LtaskID
        set-current-plot "TaskAction-StartTime"
        plotxy LMachineID * 3  LstartTime
        ;;
        set-current-plot "TaskAction-FinishTime"
        plotxy LMachineID * 2  LfinishTime

        set-current-plot "Num of Job Assigned"
        let AverageThroughPut 0

        let tplot-flag 0
        set tplot-flag  ticks mod 100
        plotxy LMachineID * 3  numJobFinish

      ];; end of ask machine
    ];; end of ask scheduler

    set finishTime LfinishTime

    set waitTime LwaitTime
    set turnAroundTime LfinishTime - LarriveTime
    set LturnAroundTime turnAroundTime

    set assignedMachine     L2MachineID

    ask machine L2MachineID [
      set totalTurnAroundTimeM totalTurnAroundTimeM + LturnAroundTime
      if numJobFinish > 0 [
        set averageTurnAroundTimeM totalTurnAroundTimeM / numJobFinish
      ]

      ;;print (word "machine totalTaskTime " L2MachineID " totalTaskTime " totalTaskTime " nextAvailTime " nextAvailTime)
      set-current-plot "Total task Execution Time"
      plotxy L2MachineID * 2   totalTaskTime
      ;; plotxy L2MachineID * 3   totalTaskTime
      if nextAvailTime > 0 [
        if totalTaskTime > nextAvailTime [
         ;;print (word "totalTaskTime > nextAvailTime " totalTaskTime  " > " nextAvailTime " How could this happen?")
```

```
        ]

        if nextAvailTime > 0 [
          set utilizationM  totalTaskTime / nextAvailTime
        ]

        set-current-plot "Utilization"
        ;;clear-plot
        plotxy L2MachineID * 2   utilizationM
        ;; plotxy L2MachineID * 3   utilizationM
        ;;print  (word  "machine  totalTaskTime  "  L2MachineID  "  totalTaskTime  "  totalTaskTime  "
nextAvailTime " nextAvailTime  " utilization " utilizationM)
      ]
    ]

    set taskDone 1
    set finishedJobs finishedJobs + 1
    ]
  ] ;; end if ask tasks
  ;; ***********************************************************************************

  ;; Check starting time
  ;; Check execution time
  ;; check finish time
  ;; numJobArrive
  ;; numJobStart
  ;; numJobFinish
  ;; numJobWait
  ;; ***********************************************************************************

  let LMachID 0
  set LTaskID 0
  ask tasks with [taskDone = 1] [
     set LMachID assignedMachine
     set LTaskID taskID

     ;;print (word "taskID " taskID " MachineID "  LMachID  "  finishTime " finishTime  "global time tick
" ticks)


     if taskFinish = 0 [ ;; only check if this task is not yet finished : waiting, execution .....

        ;;print (word "I am not yet finished ")

        if finishTime =  ticks [ ;; I am finished

          ;;print (word "I am finished "  "finishTime " finishTime "tick " ticks)
          set taskExecution 2
          set taskWaiting 2
          set taskFinish 1
          ask machine LMachID [
             set numJobFinish numJobFinish + 1
             set numJobWait numJobWait  - 1
             ;;print (word " I am finished taskID " LTaskID  " numJobWait " numJobWait)
          ]
```

```
      ;; set taskDone 2
    ]




    if startTime = ticks [  ;; I am ready to be execute
      ;;print (word "I start execution "  "startTime " startTime "tick " ticks)
      set taskExecution 1
      set taskWaiting 2
      ask machine LMachID [
         set numJobStart numJobStart + 1
      ]
    ]

    if startTime < ticks [  ;; I am waiting
      if taskWaiting = 0 [
        ;;print (word "I am waiting "  "startTime " startTime "tick " ticks)
        set taskWaiting 1
        ask machine LMachID [
           set numJobWait numJobWait + 1
           ;;print (word " am waiting taskID " LTaskID  " numJobWait " numJobWait)

        ]
      ]
    ]

  ]


]


;; end of ask tasks taskdone = 1
;; ********************************************************************************


set totalTurnAroundTime totalTurnAroundTime + LturnAroundTime
if finishedJobs > 0 [
  set averageTurnAroundTime totalTurnAroundTime / finishedJobs
]

let plot-flag 0
set plot-flag  ticks mod 10

if plot-flag = 0 [

 if taskScheduled = 1 [
   ;;setup-task-info
   ask machines [
    set shape "house"
    set color yellow
    right random 360
    forward 1
    set-current-plot "machine"
```

```
      plotxy  random 1 random 1
     ]

     ask schedulers [
      set shape "car"
      set color green
      right random 360
      forward 2
      set-current-plot "machine"
      plotxy  random 1 random 1
     ]

     ask decos [
     ;;ask tasks [
      set shape "person"
      set color white
      right random 360
      forward 3
      set-current-plot "machine"
      plotxy  random 1 random 1
     ]

  ]

]

set totalTime sum [totalTaskTime] of machines
if numMachine > 0 [
   set averageTaskLengthMachine totalTime / numMachine
]

set throughPut sum [numJobFinish] of machines
if numMachine > 0 [
   set throughPut throughPut / numMachine
]

if taskScheduled = 1 [
  set numJobFinished numJobFinished + 1
  set averageTaskLengthTask  totalTime / numJobFinished
]

set averageWaitTime sum [waitTime] of tasks
if averageWaitTime > 0 [
  set averageWaitTime  averageWaitTime / totalJobs
  set-current-plot "WaitTime"
  plotxy ticks averageWaitTime
]

;;
;;  Auto Tuning operation if enable
;; add a machine when the waittile is too long
;; remove a machien whe the waiting time below minWaitTimeAllow
;;
if AutoTuning [

  if FlagAutoTune = 1 [
```

```
    ;;print (word "FlagAutoTune is 1")
   if averageWaitTime > maxWaitTimeAllow [
     set autoTuneTickCount  ticks
     set nextautoTuneTickCount ticks + maxWaitTimeAllow
     if numMachine < 8 [
       set numMachine numMachine + 1
       set FlagAutoTune 0
       print (word "add one machine ")
       output-print "System is overloaded! Add one more machine to service"
       ;;print (word "reset FlagAutoTune to 0")
       set-current-plot "TaskAction-StartTime"
       clear-plot
       set-current-plot "TaskAction-FinishTime"
       clear-plot
       set-current-plot "Num of Job Assigned"
       clear-plot
       set-current-plot "Total task Execution Time"
       clear-plot
       set-current-plot "Utilization"
       clear-plot
     ]
   ]

   if averageWaitTime < minWaitTimeAllow [
     set autoTuneTickCount ticks
     set nextautoTuneTickCount  ticks + minWaitTimeAllow
     if numMachine > 2 [
       set numMachine numMachine - 1
       set FlagAutoTune 0
       print (word "remove one machine ")
       output-print "System is underloaded! Remove one more machine from service"
       ;;print (word "reset FlagAutoTune to 0")
       set-current-plot "TaskAction-StartTime"
       clear-plot
       set-current-plot "TaskAction-FinishTime"
       clear-plot
       set-current-plot "Num of Job Assigned"
       clear-plot
       set-current-plot "Total task Execution Time"
       clear-plot
       set-current-plot "Utilization"
       clear-plot
     ]
   ]
  ]

  if ticks = nextautoTuneTickCount  [
    set FlagAutoTune 1
    ;;print (word "reset FlagAutoTune to 1")
  ]


  if FlagAutoTune = 0 [
   ;; print (word "FlagAutoTune is 0")
  ]
]
```

```
;;set plot-flag  ticks mod 5
set plot-flag 0

if plot-flag = 0 [

  set maxStartTime max-one-of machines   [ currentStartTime ]
  set minStartTime min-one-of machines   [ currentStartTime ]

  set maxFinishTime max-one-of machines  [ nextAvailTime    ]
  set minFinishTime min-one-of machines  [ nextAvailTime    ]

  set maxThroughput max-one-of machines  [ numJobFinish     ]
  set minThroughput min-one-of machines  [ numJobFinish     ]

  set maxTaskLength max-one-of machines  [ totalTaskTime    ]
  set minTaskLength min-one-of machines  [ totalTaskTime    ]

  set maxUtilization max-one-of machines [ utilizationM     ]
  set minUtilization min-one-of machines [ utilizationM     ]

  set maxJobWait max-one-of machines [ numJobWait     ]
  set minJobWait min-one-of machines [ numJobWait     ]

  set maxJobFinish max-one-of machines [ numJobFinish     ]
  set minJobFinish min-one-of machines [ numJobFinish     ]



  set maxStartTime2 max   [ currentStartTime ] of machines
  set minStartTime2 min   [ currentStartTime ] of machines

  set maxFinishTime2 max  [ nextAvailTime    ] of machines
  set minFinishTime2 min  [ nextAvailTime    ] of machines

  set maxThroughput2 max  [ numJobFinish     ] of machines
  set minThroughput2 min  [ numJobFinish     ] of machines

  set maxTaskLength2 max  [ totalTaskTime    ] of machines
  set minTaskLength2 min  [ totalTaskTime    ] of machines

  set maxUtilization2 max [ utilizationM     ] of machines
  set minUtilization2 min [ utilizationM     ] of machines

  set maxJobWait2 max [ numJobWait     ] of machines
  set minJobWait2 min [ numJobwait     ] of machines

  set maxJobFinish2 max [ numJobFinish     ] of machines
  set minJobFinish2 min [ numJobFinish     ] of machines

  set loadBalanceIndex maxTaskLength2 - minTaskLength2


  set LBPI1 maxTaskLength2 - minTaskLength2
  set LBPI2 maxUtilization2 - minUtilization2
```

```
    set LBPI3 maxJobFinish2 - minJobFinish2

    ;;print (word " M " maxJobWait " numJobWait " maxJobWait2)
    ;;print (word " M " minJobWait " numJobWait " minJobWait2)
  ;;   print (word " M " maxJobWait " numJobWait " maxJobWait2)
  ;;   print (word " M " maxJobWait " numJobWait " maxJobWait2)

    set LBPI4 maxJobWait2 - minJobWait2


  ]

  set utilization sum [utilizationM] of machines
  ;;set utilization  utilization / numMachine
  if numMachine > 0 [
    set utilization  utilization / numMachine
  ]

  ;; advancing one tick count
  tick

end ;; end of GO procedure
```