Game Theoretic Multi-Agent Algorithms for the Job Shop Scheduling Problem

Orwa Horace Owiti*, Elisha T. Opiyo Omulo, William Okelo-Odongo and Bernard Manderick
School of Computing and informatics
University of Nairobi,
P.O Box 30197 - 00100 GPO Nairobi
*Email: horaceowiti {at} gmail.com

Abstract--- **Job shop scheduling problem is a problem of scheduling *n* jobs on *m* machines with each job having a set of equal number of operation that are to be process in unique machine routes. The Job Shop Scheduling (JSSP) is one of the hardest combinatorial optimization problems and has been researched over the decade. This study proposes a new approach to solve a Job Shop Scheduling problem by structuring the problem as multi-agent system (MAS) and using 3 game theoretic algorithms to achieve the scheduling objectives. The objective of this study is to minimize the makespan. This approach is meant to achieve feasible schedules within reasonable time across different problem instances. This research solves the scheduling of operation on different machine and defines the sequence of operation processing on the respective machine. Job Scheduling problem is a resource allocation problem which is mainly apparent in manufacturing environment, in which the jobs are allocated to various machines. Jobs are the activities and a machine represents the resources. It is also common in transportation, services and grid scheduling. The result and performance of the proposed algorithms are compared against other conventional algorithms. The comparison is on benchmark data used across multiple studies on JSSP.**

Keywords- ***Job shop scheduling problem(JSSP), Random token Game(RTG), Potential Games(PG), Random Games(RG), Game theory, Makespan, Q-learning, Reinforcement learning, Multi-agent system.***

I. INTRODUCTION

A Scheduling problem can be defined as the problem of allocation of limited shared resources over time to competing activities. Scheduling problems have over the years attracted interest in much research and has been the subject of a significant amount of literature in the operations research and artificial intelligence fields. Job-shop scheduling is one of the most commonly researched about problems in the domain of scheduling problems. Job-shop scheduling problem is a scheduling problem where, there are ***m*** machines and ***j*** jobs where ***m>1***, each job has a set of operations ***o*** and has associated a processing order assigned for its operations. Job-Shop scheduling is a known NP-Hard. The objective of this can be classified under the following;

Minimize the Makespan- The Makespan is the total length of the schedule, that is, the time it takes all jobs finish processing. This is formulated as

$$\mathbf{M = max\{ C_1 \ldots\ldots\ldots C_n\}} \qquad \mathbf{(1)}$$

Where,

$\mathbf{C_j}$**=** the earliest time job ***j*** finishes processing.

Minimize Tardiness- In situations where the jobs ***j*** have deadlines d_j, tardiness is the duration of time delays past its deadline. The tardiness of the schedule **T** is,

$$\mathbf{T} = \sum_{j=1}^{n} max\{c_j - d_j, 0\} \qquad \mathbf{(2)}$$

Minimize lateness- Lateness for a job is defined as,
The Lateness of the schedule **L**

$$\mathbf{L} = \sum_{j=1}^{n} max\{c_j - d_j\} \qquad \mathbf{(3)}$$

This study concentrate on minimize of makespan for scheduling problems. We structure the problem as multi-agent system (MAS) and we introduce three algorithms based on game theory, reinforcement learning. We describe MAS as a computerized system composed of multiple interacting intelligent agents within an environment. Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve. Because of its nature it lends its self to solving problems where distributed decisions are necessary. We define resources (either machine or jobs operations) as agent and define games in which they participate to achieve a feasible solution.

A. Problem Description and formulation

We formalize the job shop scheduling problem as follows; A problem instance ***P= (M, O, J)*** in job shop scheduling consists of

- A set ***M*** of Machines,

- A set ***O*** of operations ***o***, each associated with a machine ***M(o)Є M*** and having a duration ***d(o) Є N*** and
- A set ***J*** of jobs $\mathbf{J(o_1 \ldots\ldots\ldots\ldots o_n)}$ (each operation has exactly one occurrence.)

A given Schedule ***S*** for ***P*** assigns to every operation ***o*** a starting time ***T(o)***: on the relevant machine time

T (o)≥0 for all ***o Є O***

We define an operations processing time ***P(o)*** *as*

$$P(o) = T(o) + d(o) \qquad (4)$$

We define a precedent constraint on ***T (o')on T (o)*** such that

$$T(o) \geq T(o') + d(o') \qquad (5)$$

for operations ***o'*** preceding **o** in the same job.

Our objective function in the problem is to minimize the makespan in search of a near optimal schedule. We defined the Makespan as the time the last machine finishes process the last operation, therefore the makespan of a schedule ***M(S),*** can be defined as

$$M(S) = MAX(T(o_1) + d(o_1), T(o_2) + d(o_2), \ldots\ldots\ldots\ldots T(o_n) + d(o_n)) \qquad (6)$$

To define this as a multi-agent system, We adopt and extend [Opiyo et al,2009]'s definition of a multi-agent system for parallel machine scheduling. Same as their study had, we make the following considerations;

- A multi agent system to be a system that consists of the agents, the agents act as autonomous entities that can sense and react to the changes in their environments.
- Game theory as the study of interactions in contexts where the participants make the choices to affect the overall status in the game. A game is a structure that consists of a set of the agents, a set of the agent actions or choices and a set of the agent payoffs associated with their actions. A situation where schedules are generated by agents as they choose machines can be considered as a game [Opiyo et al. 2008b].

B. Literature review

Job shop scheduling is among the hardest combinatorial optimization problems and is NP-*complete (Garey and Johnson, 1979).* An NP-complete or NP-hard problem is where no algorithm exists (unless P=NP) that in polynomial time is able to solve all possible instances of the problem. Hence, the solution time risks increasing exponentially with the number of jobs. *(Karin Thörnblad, 2013).* According to *(Karin Thörnblad, 2013)* JSSP remains a NP-complete problem despite the objective function selected. Over the past forty years different solution approaches have been proposed to address the JSSP. These approaches can be categorized in two, these are;

Optimization Algorithms: These are usually mathematical programming based approaches that work toward achieving optimal solutions. According to *(Azizizoglu and Kirca 1999a)* they involve the process like formulating Mathematical models for the problem, and using exact algorithm such as branch-and-bound algorithms or mathematical formulation to solve the problem. These methods simply build an optimum solution from the problem data by following a simple set of rules which exactly determine the processing order. Optimization algorithms have been known to solve a given problem optimally with a requirement that increases polynomially with respect to the size of the input. Optimization approaches usually process like formulating Mathematical models for the problem. These approaches form the earliest of approaches in solving scheduling problems, The first example of an efficient method and probably the earliest work in scheduling theory is *(Johnson ,1954)* who develops an efficient algorithm for a simple two machine flow shop whose objective function was to minimizes the maximum flow time. The two most common methods in these approaches are ; Branch-and-bound algorithms and Mathematical formulation.

Approximation Algorithms: These are usually heuristic/Meta-heuristic algorithms based approaches that aim to give an approximately near optimal solution rather than the optimal solution. These methods are usually preferred and are better for larger problem/dynamic problems/ problems with multiple constraints as they are more likely to converge to a good enough solution much earlier than optimization methods can achieve an optimal solution. In most problem instance successful algorithm have shown that the solution derived from approximation approaches are usually close to enough to the optimal solution. Since the solution is close to optimal and generated in much less time, (Blum and C.Roli, A. 2003) argue that the benefit of having using far less resources outweighs the disadvantage of not arriving to an absolute optimal solution. The main classification of approximation algorithms, that is, priority dispatch rules, bottleneck based heuristics, artificial intelligence and local search methods.

A multi-agent system (M.A.S.) is a computerized system composed of multiple interacting intelligent agents within an environment. Multi-agent systems are centered on the concept of a rational agent. An agent is anything that can perceive its environment through sensors and act upon that environment through actuators (Russell and Norvig, 2003). According to (G WeiB, 2000) interest in multi-agent systems is largely founded on the insight that many real world problems are best modeled using a set of agents instead of a single agent. In particular, multi-agent modeling makes it possible to cope with natural constraints like the limitations of the processing power of a single agent or the physical distribution of the data to be processed and allow us to profit from inherent properties of distributed systems like robustness, fault tolerance, parallelism and scalability.

(V lesser,1995) state that The current set of multi-agent applications can be classified into three broad areas. The firstly, distributed situation assessment Applications, such as distributed network diagnosis, emphasize how (diagnostic) agents with different spheres of awareness and control (network segments)should share their local interpretations to arrive at consistent and comprehensive explanations and responses. Secondly distributed expert systems applications, such as concurrent engineering, emphasize how agents negotiate over collective solutions (designs) given their different expertise and criteria. The next generation of applications alluded to will probably involve all the emphases of these generic applications and more. Finally as is in our case, distributed resource planning and allocation applications, such as distributed factory scheduling, emphasize how (scheduling) agents (associated with each work cell) should coordinate their schedules to avoid and resolve conflicts over resources and to maximize system output.

According to (Neumann and Oskar Morgenstern,1944) ,Game theory is an economic theory that models interactions between rational agents as games of two or more players that can choose from a set of strategies and the corresponding preferences. It is the mathematical study of interactive decision making in the sense that the agents involved in the decisions take into account their own choices and those of others. Choices are determined by stable preferences concerning the outcomes of their possible decisions, and agents act strategically, in other words, they take into account the relation between their own

II. ALGORITHMS AND METHODS

In this section we define our job Shop algorithm as a Multi-agent system environment and we formulate three game theoretic algorithms for solving job shop scheduling problems.

A. Random Token Game

In defining algorithms for parallel machine scheduling [Opiyo et al, 2008] define random choice games are those in which the agents make choices at random without considering any other matters. In their definition an agent are allowed to make moves in turn and each agent in its turn makes random decision which machines they would like to be processed on and select the earliest available time slot on the machine. After all the agents have made their move the resultant schedule is evaluated. This process is repeated in several rounds and at the end the most suitable/ shortest schedule is select as a feasible solution. This work was able to demonstrate that it is possible to achieve a relatively feasible schedule using random select of schedule in a schedule search space. It gives us great insight on the distribution of solution in the search space. We try to define a similar algorithm for job shop scheduling. Unlike in parallel machine scheduling, job shop scheduling as the following complications when trying to employ a pure random strategy in selection of a feasible solution from the search space; Agents/operations are tired to a machine, that is, the machine is already pre-selected and precedence constraints among agents, that is the start time $S(A)$of an agent A ,

$$S(A) \geq S(A') + T(A') \qquad (7)$$

Where A' is the preceding Agent with a processing time of $T(A')$.

These constraints limit the flexibility of an agent in machine selection and put a constraint its selection of a time slot on a machine. To achieve similar a random selection of solution in a such space with the above constraints in job shop scheduling, we introduce a random token notion. The Random token randomizes the playing turn for the agents. This works as follows; we divided the game in two stages for all rounds, the stages are as follows;

Selection Stage; this stage allows random selection of agent turns which will result in random ordering of agents in the machine. This works by introducing a random token in the environment that is assigned to an agent at random. The agent that has the token is allowed to order itself on its respective machine waiting queue on an assigned a priority on first come first serve basis. To formally state this, If O_{mn} Represents an agent O with processing time on machine m and it was the n^{th} agent to make a selection on the machine, then its priority value $P(O_{mn})$ (lower value signifying higher priority) is;

$$P(O_{mn}) \leq P(O'_{m(n+1)}) \leq P(O''_{m(n+2)}) \qquad (8)$$

Where O' and O'' followed agent O in selection of the machine in that respective order.

Allocation Stage; the allocation stage was motivated by shift bottle neck paradigm. In shift Bottleneck, an initial selection of a schedule is selected as we have done in the selection stage without actual time share allocation. If we were to evaluate the schedule as it is now with the agents arranged in a first come first serve order, the schedule will have multiple delays among the operations and we will have unnecessary idle time on the machine The shift bottle neck algorithm recognizes that in a schedule the is always at least one point/bottleneck that affects its performance. The aim of the shift bottleneck is slow minimize/shift the bottle in several iterations. We adopt a similar iterative approach but in our algorithm it's the agent that makes the decision whether to shift or stay based on their internal states, The agent act for the social good and if an agent consider itself a possible bottleneck, it shifts self to remove the bottle neck if not its stays .All the waiting agents with the highest priority on each machine's waiting queue are allowed to a turn, there status changes to active and they are allowed to evaluate their position. If an agent see that they could be possible bottlenecks they will choose to move to the back of the queue assume the lowest priority on the queue and status change back to waiting. The Agent suspects it may bottleneck using the following criteria;

If an agent ***A*** has a predecessor and the predecessor has not been schedule yet (acquired a time share), then ***A*** knows it's a might be a bottleneck on a machine if there exists other agents on the machine with a lower priority. In this case the agent will move to the back of the queue.

If an agent ***A'*** has a predecessor ***A*** that has already been scheduled and its difference between ***A*** 's expected processing end time , ***P(A)*** and the *'next available start time'* on the machine **M** , **E(M)** is twice as big as the average processing time of the all the agents queued on the machine, then the agent suspect itself to be a bottleneck. A Machines *'next available start time'*, ***E(M)***, is the sum of all the agents that have been scheduled on the machine. If a machine M has 3 agents scheduled on it, An agent defined as $A_{(job, machine)}$.

$$E(M)= (P(A_{1M}) + P(A_{2M}) + P(A_{4M}))/3 \quad (9)$$

If (A'_{1m}) is the one evaluating it situation and it has a predecessor **A**, the processing end time of **A,**

$$P(A) =S(A)+D(A) \quad (10)$$

Where;-

S(A) is the processing start time of ***A,***

D(A) is the processing time/duration of ***A.***

A at this point would consider its 'possible' processing start time, $S(A'_{1m})$, as equal to the processing end time , ***P(A),*** of ***A.*** $S(A'_{1m}) = P(A)$ ***(11)***

(A'_{1m}) consider itself a bottleneck in the schedule, and move to the back of the machines waiting queue. If the agent (A'_{1m}) has evaluated its situation and does not consider itself a bottleneck, the agent will be scheduled on the machine by selecting the earliest possible start time on the machine. This would be the greater of ***P(A)***and ***E(M).*** That is, if

$P(A) \geq E(M)$ then $S(A'_{1m}) = P(A)$ *else*
$S(A'_{1m}) = E(M)$

The same steps are repeated for each agent for the number of Iteration needed till all the agents have been successfully scheduled. A complete **selection stage** followed by a complete **allocation stage** constitute a round in the game, each round produces a candidate schedule from the search space. At the end of the game, the makespan ***m*** is evaluated of all the candidate solution **s** Є ***S*** where ***S*** represents the search space and selects a feasible solution ***f(s)*** using the following criteria.

$$f(m_s)= MIN(m_1, m_2, m_3, \ldots\ldots\ldots m_s) \quad (12)$$

Because we achieved a random initial selection by using a randomized token. We can say that we are selecting schedules at random from the search space and thus we have achieved a similar effect that [Opiyo, et al] achieved with their random games in parallel machine scheduling. Therefore we can state for a typical job shop problem there is a random distribution of solution on the search space.

B. Potential Game

[Opiyo et al, 2008] described potential games as those in which the incentive of all players to change their strategy is expressed in one global function called the potential function. The progressive actions of the participants lead to a stable state. In this section we defined a game that behaves in this way. In our interpretation we define a function that reward's/penalize agents based of the action it takes in the environment. As agents take actions the gain a bit of appreciation of their environment as their actions are reinforced by their reward/penalty system. To achieve this we borrow concepts from reinforcement learning, which transform our game into a policy search function, That is, the aim of the game is meant to teach an agent what to base their actions (what policy to use) and at the end of a learning phrase is able to make decision on a certain state based on their experience with on that particular state. We utilize markov decision process to model our agent learning as follows;

A Markov Decision Process (MDP) is a 4-tuple [S, A, T, R] where:

- **S = s1, ..., sn** denotes a finite set of states;
- Set of actions **A**, and **A(s)** Є **A**, where **A(s)** is the finite set of available actions in state **s** Є **A**;
- **T : S × A × S →[0, 1]** is the transition function, **T(s, a, s')** specifies the probability of ending up in state **s0** when performing action **a** in state **s**;
- **R : S × A × S' → R** is the reward function, R(s, a, s') denotes the expected reward for the transition from state **s** to state **s'** after taking action **a**.

For MDPs, the Markov property assures that the transition from s to s' and the corresponding reward **R(s, a, s')** depend only on the state s and the action a, and not on the history of previous states and actions.

Q-learning provides a way of determining utility for agent decisions using the utility function; (13)

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Where;

$Q(s,a)$ - The utility of state **s** defined recursively the update rule above

α - is a learning rate.

γ - Discount rate of subsequent action.

r- Reward of taking action ***a*** on state ***s***

The Psuedocode of the exploration/learning process is as following

- Initialize **Q-values** arbitrarily
- for each episode do
    - Initialize **s**
    - for each episode step do
        - Choose **a** from **s**
        - Take action **a**, observe state **s'** and **r**

- Update
  $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
  - end for
- end for

By using Q-learning in a an environment where the agent has partial observation of the environment, an agent can learn a finite set of the search space and using the potential function, derive utility for each its decisions and finally select a series of decision policy that are beneficial to it. If we utilize Q-learning we are able to achieve (opiyo, et al, 2008)'s description of a potential game where there is a global function that guides agent in decision making. The reward function and utility function act together to guide the agent in decision making.
Our potential Game environment will therefore consist of the following;

Actor only agents, these agents will be responsible for selection of policy during the exploration/learning stage of the game. The agent at each time step in schedule formation will employ a certain policy as they seek to achieve a complete schedule.

A critic only agent will responsible for evaluating the policies employed by each agent at every time step and will give feedback to the agents in terms of a reward/penalty. In our algorithm the critic's memory structure will also be responsible for storing the learned utility of each state. Once the exploration is done information learned by actor agents is used by them to select successive policies, as they form what they consider to be an optimal schedule.

Action Policies. These represent dispatch rules, the rules act as policies that is available for the actor to choose from when selecting the next action, that is, whenever an actor makes a specific move, their move has to be based on a specific policy. At learning stage an actor would tryout one or more policies and will observe the reward/penalty using that policy on that particular state. The aim of our global function is to define a series of policies that an agent can employ that would lead to an optimal schedule .The different policies the agent can employ at a specific place are, **FIFO**- First In First Out ,**LIFO**-Last In First Out, **SPT**-Shortest Processing time and **LPT**-Longest Processing time.

Global Potential function. This function is used by the critic agent to appraise and influence the action of the actors. As we had earlier demonstrated in the Q-learning algorithm, the function assigns a reward on agent actions and defines the utility of each Q-pair. The utility will finally influence the agents' decision on which policy to employ at the selection stage.

Reward structure. The reward structure is used by the critic agent to appraise the agent actions. A reward is quantification of how good the selected policy in the current state is. We define our reward a penalty and quantify it as total process of all waiting Jobs. This is a sum of the processing time ***p(o)*** of the ***n*** jobs that are waiting global dispatch queue and on waiting queues of machine after all actor agents have selected a single action. That is,

$$r = -1\left(\sum_{0}^{n} p(o)_n\right) \quad \textbf{(14)}$$

C. Random games

The last game we define is the random games. This game borrows the same concept as potential games where there are actor only agents that represent a machine. The MAS environment is structured as follows; Our environment will therefore consist of the a global dispatch queue, this is a queue that holds all jobs before they can be moved to a machines/agents waiting queue. Jobs move to from the dispatch queue to the machine queue when there is no constraint to their processing e.g. they have no predecessor or their predecessor has already been scheduled for processing. Our environment also consists of Actor only agents, these agents are responsible for selecting the next operation to process.

The only difference between random and potential games is that in random games the agent does not employ any policy to base its actions on it simply selects the next operation to process at random. There the agent explores the search space randomly. This description of a random game is the same as described in (opiyo et al, 2008) where game has multiple presets ***n*** number of iterations and in each iteration, an agents select operations at random from there waiting queues until all operations have been schedule. This forms a candidate solution $\boldsymbol{S_0}$ and its makespan is noted. At the end of the game a candidate solution $\boldsymbol{S_i}$ is selected as the feasible solution with the near optimal schedule. The formula below shows the mode of selection of this schedule.

$$\min_n S(S_0, S_1, S_2, \ldots\ldots\ldots S_n) \quad \textbf{(15)}$$

III. RESULTS AND DISCUSSIONS

For our test we use benchmark problems used in Beasley's operation research library complied by Professor Beasley J (Beasley 2005), this is found on Brunel's universities website. The benchmarks offers a list of different instances of job shop problems complied by different researchers in there works. The specific instances of problems selected are from ;

- ABZ 5 problems of 2 sizes proposed by (Adams, Balas and Zawack 1989). ABZ 5 and ABZ 6 instances

of size 10×10 with processing times from the intervals [50,100] and [25,100] respectively and ABZ 7 – 9 instances of size 20×15 and processing times

- la01-la40 are from "Resource constrained pr scheduling: an experimental investigation of heu scheduling techniques" by S. Lawrence.
- Car1-car8 are from "Ordonnancements a contra disjonctives" by J. Carlier.
- orb1-orb10.

We also compare our algorithm to the following algori which are heuristic based.

Table 1: Benchmark algorithms

Problem	*Symbol*	Types of problem	Instances Sizes Handle
"Multi-resource shop scheduling with resource flexibility and blocking." (Y Mati and X Xie, 2011).	***MX***	Job Shop Scheduling	10 X 10, instance only
"*Use of an Artificial Immune System for Job Shop Scheduling*", (CAC Coello et al, 2003)	***AIS***	Job Shop Scheduling	Multiple
"A contribution to the stochastic flow shop scheduling problem", (M. Gourgand et al, 2003)	***SD***	Flow Shop Scheduling	Multiple
"Job-Shop with Generic Time-Lags: A Heuristic Based Approach". (P. Lacomme, 2011)	***GLT***	Flow Shop Scheduling Job Shop Scheduling	Multiple

The following are the sample results on the benchmark problems

Figure: 1 Random games performance

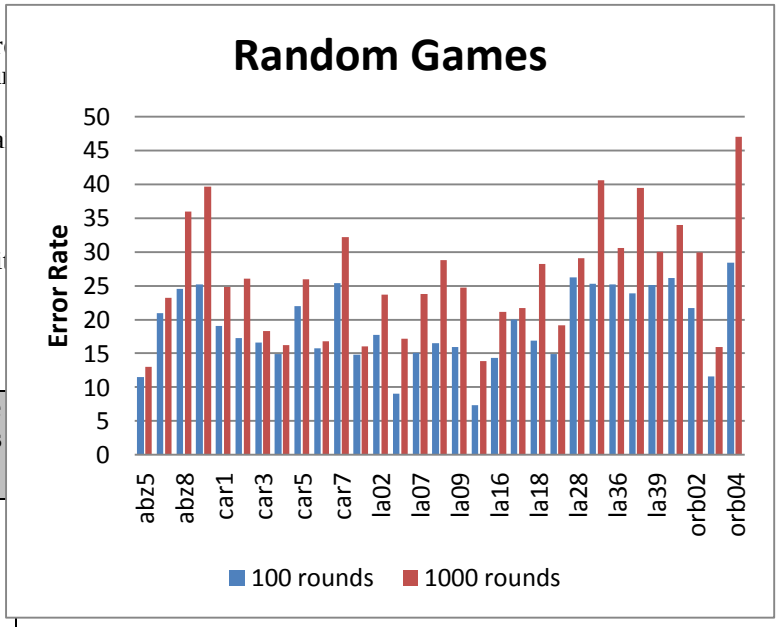

Figure: 2 Potential games performance

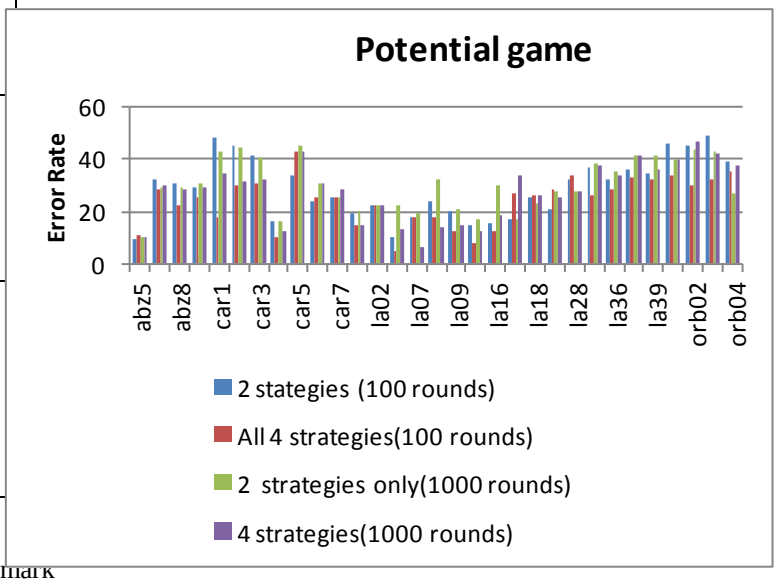

Figure: 3 Random token games performance

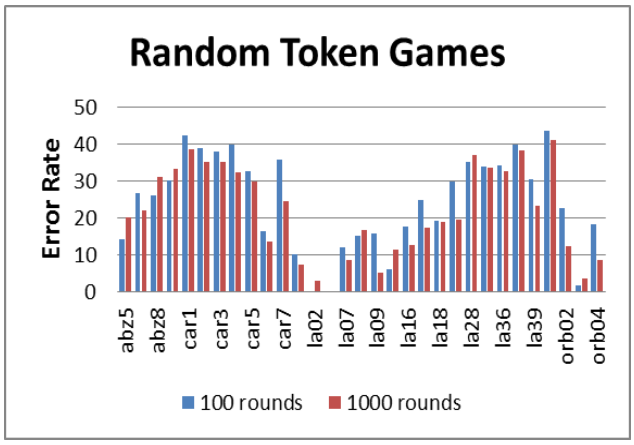

Table 2: Algorithm comparisons

		PG	AIS	MX	GTL	RG	RTG
la01	10 X 5	15	16	32	31	15	8
la02	10 X 5	22	18	37	37	18	0
la05	10 X 5	5	4	25	48	9	0
abz5	10 X10	9	19	38		12	14
la16	10 X10	12	16	28	69	14	13
la17	10 X10	17	16	30	65	20	17
la36	15 X 15	28	23	62	38	25	33
la38	15 X 15	33	27	65	42	24	38
abz7	15 X 20	28	26			21	22
abz8	15 X 20	23	25			25	26
la07	15 X 5	6	8	36	26	15	9
la08	15 X 5	14	12	46		17	15
la09	15 X 5	13	7	45		16	5
la10	15 X 5	8	2	36		7	6
la28	20 X 10	28	28	96	64	26	35
la29	20 X 10	26	22	89		25	34
orb03	10 X10	32	27	30		12	2
orb04	10 X 10	27	21	28		28	9
car1	11 X 5	18			96	19	39
car2	13 X 4	30				17	35
car3	12 X 5	31				17	35
car4	14 X 4	10				15	32
car5	10 X 6	34			77	22	30
car6	8 X 9	24				16	14
car7	7 X 7	25			67	25	25

Our results provided the following insights

The potential games algorithm perform relative better with more strategies used. This is because it increases breadth of choice and actions available to an agent. This increases the learning experience of an agent and increases the chance of learning a more favorable solution. The different in quality of solution produce when using only SPT and LPT compared to all four strategies, increase with the sizes of the instance. This is because using only 2 strategies limits the game to a subset of solutions in the search space. Limiting the experience scope of the agent.

Both the Potential game and Random games do not show improvement the quality of solution designed when the number of paths was increased significantly from 100 to 1000. This is because the quality of schedule generated for these more on the number of strategies used as they increase the breadth of choose or scope of learning for the agent. Increasing the number learned paths learned without increasing the number of strategies available to the agents only leads the agent to learn multiple similar schedules, thus the agents is already limited to a certain range quality of solutions they can achieve.

Increasing the number of rounds in the random token game does show improvement in the quality of schedule generated. This is because it increases the number of solution the algorithms has to choose from the search space increase the probability of selecting a more favorable solution.

The tests on Potential and Random games also shows relatively poor performance on flow shop problems(car1-car7) compared to the job shop problem this can be attributed to the fact that because of the nature of a flow shop problem which leads to some agents having a larger action set(operations to choose from) than others. Machines/agents the process the initial operations of the jobs end up being the only ones playing at the beginning of the game. The lower the number of agent learning at each stage reduces the learning experience and also reduces the chances of achieving favorable solutions.

The test shows that quality of solution of Random Games and Potential Games are affecting by the sizes of the problem instance. Quality reduces when dealing with large problem instances. This can be attributed to the fact increasing the size of instance significantly increases the size of search space. Since this games are based on learning a subset of the search space based on the strategies selected and searching for a solution within that subset, the large the search space the harder it is to get a quality subset.

Only the Random Token Game doesn't show better adaptation to change in instance problem size. This can be attributed to the fact that it works by selecting solutions from the workspace at random and refining them, thus not greatly affected by the size of the search space.

Our algorithms have shown relatively good performance compared to the selected benchmark problems. On average we achieved better or equal performance across all problem instances.

IV. CONCLUSIONS

This paper deals with defining three game theoretic algorithms for solving job shop scheduling problems. Our algorithms have shown relatively good performance on the benchmark data and we were able to converge to a feasible solution in relatively good time. We have also been able to demonstrate by defining the job shop problem as a multi-agent system we are able to provide algorithms that provide good solution across different sizes of problem instances. From the study we can recommend the following further studies.

- This paper has dealt with job shop scheduling where scheduling is static and job are scheduled as a batch. In the real world problems tend to be further research work can be done to the algorithms to apply the two dynamic job shop scheduling.
- Our study choose a basic where of structuring the reward/reinforcement function based on total processing time of un-scheduled jobs at any given point. Further work can be done to refine the algorithm by defining better reward structure to improve the learning of an agent.

REFERENCES

[1] A. AitZai and M. Boudhar, (2013). "Parallel branch-and-bound and parallel PSO for the job shop sche duling with blocking", Int. J. Operational Research, vol. 16, No. 1.
[2] Adams J, Balas E, Zawack D. (1988).The shifting bottleneck procedure for job shop scheduling. Management Science, 34(3): 391-401.
[3] Anant Singh Jain and Sheik Meeran, (1998). A State-Of-The-Art Review Of Job-Shop Scheduling Techniques Department of Applied Physics, Electronic and Mechanical Engineering University of Dundee, Dundee, Scotland, UK.
[4] Balas E, Lenstra J K, Vazacopoulos A. (1995).The one machine problem with delayed precedence constraints and its use in job shop scheduling. Management Science, 41(1):94{109.
[5] Beasley J (2005) Or-library http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html.
[6] Blum, C.; Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison 35 (3). ACM Computing Surveys. pp. 268–308.
[7] CAC Coello et al, (2003), "Use of an Artificial Immune System for Job Shop Scheduling" .Department of electrical engineering. National Polytechnic Institute, Mexico.
[8] Chu, C., Portmann, M. C. and Proth, J. M. (1992) A Splitting-Up Approach to Simplify Job-Shop Scheduling Problems, International Journal of Production Research 30(4), 859-870.
[9] Elisha T. O. Opiyo, Erick Ayienga, Katherine Getao, William Okello-Odongo, Bernard Manderick, and Ann Nowé. Game Theoretic Multi-Agent Systems Scheduler for Parallel Machines. International Journal of Computing and ICT Research, Special Issue Vol. 1, No. 1, pp. 21-27
[10] G WeiB. (2009) Learning to Coordinate Actions in multi-agent Systems, Institut fur lnformatik, Technische Universitat Miinchen Arcisstr. 21, 8000 Miinchen 2, Germany
[11] G Weiss (2013). A Modern Approach to Distributed Modern Approach to Artificial Intelligence, Multiagent Systems , MIT press , Cambridge , Massachusetts, USA.
[12] Garey, M. R. and Johnson, D. S. (1979) Computers and Intertractability: A Guide to the Theory of NPCompleteness, W. H. Freeman, San Francisco.
[13] H. Chen, P.B. Luh, (2003). European Journal of Operational Research 149 (2003) 499–512-2003
[14] J.F. Muth and G.L. Thompson. (1963) Industrial Scheduling. Prentice-Hall, Englewood Cliffs, N.J.
[15] Johnson, S. M. (1954). Optimal Two- and Three-Stage Production Schedules with Set-Up Times Included, Naval Research Logistics Quarterly, vol 1, 61-68.
[16] Karin Thörnblad(2013), Mathematical Optimization in Flexible Job Shop Scheduling: Modelling, Analysis, and Case Studies, Department of Mathematical Sciences, Chalmers University of Technology and the University of Gothenburg.
[17] M. Gourgand et al, 2003. A contribution to the stochastic flow shop scheduling problem, European Journal of Operational Research 151 (2003) 415–43
[18] Manne, A. S. (1960) On the Job-Shop Scheduling Problem, Operations Research, vol 8, 219-223.
[19] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E., (1953), "Equation of State Calculations by Fast Computing Machines", The Journal of Chemical Physics, Vol. 21, Issue 6, pp. 1087-1092.
[20] Nikos Vlassis(2005). A Concise Introduction to Multiagent Systems and Distributed AI, Intelligent autonomous Systems Informatics Institute University of Amsterdam.
[21] Osborne, M. J. and Rubinstein, A. (1994). A Course in Game Theory. MIT Press.
[22] P. Brucker and O. Thiele. (1996). A branch and bound method for the general shop problem with sequence-dependent setup times. Operations Research Spektrum, 18:145-161.
[23] P. Brucker, P. Jurisch, and B. Sievers. (1994). A fast branch and bound algorithm for the job-shop scheduling problem. Discrete Applied Mathematics, 49:107-127.
[24] P. Lacomme, N. Tchernev and M.J. Huguet. (2012). Job-Shop with Generic Time-Lags: A Heuristic Based Approach". 9th International Conference of Modeling, Optimization and Simulation - MOSIM'12 June 06-08, 2012 – Bordeaux - France
[25] Panwalkar, S., Iskander, (1977). A Survey of Scheduling Rules. Operations Research 25(1) 45–61
[26] Pinedo M, Singer M.(1995) A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. Naval Research Logistics, 46(1): 1-17.
[27] R. Sutton and A. Barto. Reinforcement Learning. An Introduction. MIT Press/A Bradford Book, Cambridge, USA, 1998.
[28] S. Russell and P. Norvig(2003). Artificial Intelligence { A Modern Approach. Prentice Hall, Englewood Cliffs, USA.
[29] Thomas Gabel. (2009).Multi-Agent Reinforcement Learning, Approaches for Distributed,Job-Shop Scheduling Problems, Tag der wissenschaftlichen Aussprache: 26.06.
[30] Uzsoy R, Wang C S. Performance of decomposition procedures for job shop scheduling problems with bottleneck machines. International Journal of Production Research, 2000, 38(6): 1271-1286.
[31] V Lesser(1995), Mullti-agent. Systems: An Emerging Sub-discipline of AI. ACM Computing Surveys, Vol 27, No 3, September 1995
[32] von Neumann, J., Morgenstern, O., Theory of Games and Economic Behaviour, Princeton University Press, 1944.
[33] Y Mati and X Xie "Multi-resource shop scheduling with resource flexibility and blocking." (2011) ,IEEE transactions on automation science and engineering.
[34] Yailen Martínez Jiménez. (2012). A Generic Multi-Agent Reinforcement Learning Approach for Scheduling Problems. Brussels University Press
[35] Zhang, C.Y., P. Li, Y. Rao, Z. Guan. 2008. A very fast TS/SA algorithm for the job shop scheduling problem. Computers and Operations Research 35 282–294.