Industrial Engineering & Management
ISSN: 2169-0316

Chen and Rabelo, Ind Eng Manage 2017, 6:3
DOI: 10.4172/2169-0316.1000215

Research Article Open Access

Deadlock-Detection via Reinforcement Learning

Chen M* and Rabelo L

Department of Industrial Engineering and Management Systems, University of Central Florida, Orlando, USA

Abstract

Optimization of makespan in scheduling is a highly desirable research topic, deadlock detection and prevention is one of the fundamental issues. Supported by what learned from this class, a reinforcement learning approach is developed to unravel this optimization difficulty. By evaluating this RL model on forty classical non-buffer benchmarks and compare with other alternative algorithms, we presented a near-optimal result.

Keywords: Reinforcement learning; Optimization makespan; DL detection

Introduction

Due to buffer-less setting, deadlock (DL) occurs frequently in resource sharing environment and concurrent computing systems. A deadlock is a state in which each member of a group of actions is waiting for some other member to release a lock. [1] Once this DL state occurred, workflow would stack in a fixed loop and never discharged.

Figure 1 present a typical scheduling problem: 3 jobs need to be operated on 3 different machines following different sequences, each machine can only operate one job each time. How to schedule jobs in specific sequence to minimize total makespan aka processing time without deadlock is a typical optimization problem. Due to limitation of resources, deadlock happened frequently, other than a feasible solution, to find the global optimal deadlock free solution is difficult.

There are certain methods to solving deadlock problems: 1. Do nothing, 2. Kill the workflow, 3. Preempt and rollback. Other than kill the workflow, deadlock detection algorithms are more efficient in most cases and additionally, deadlock free scheduler would enable the real-time control for engineering system. Preventing or avoiding deadlock helping maintain system performance aka makespan stay at positive level.

A simple head-tail scheduling example is present in Figure 2. The rectangle stands for resource Ri, symbol Pi stands for jobs, if rectangle is empty them means no job is operating on that resource. Arrow stands for the action of each job. Second level DL has been addressed in the previous articles. Here on Figure 3, a deck lock is presented. Job P3 is moving to resource 3 then resource 1, but once it moved it will

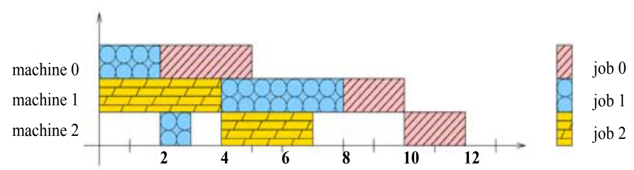

Figure 1: 3 machine 3 job scheduling problem.

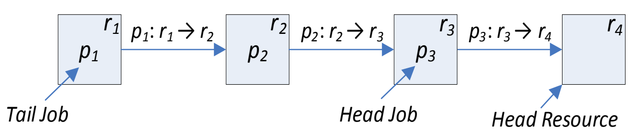

Figure 2: Deadlock I level.

be a deadlock because P1 and P2 will be non-moveable. If P2 move to resource 3 first, then P2 and P3 will be a deadlock as they heading to other's occupation while there is no buffer.

In this paper we present a new reinforcement learning approach solving this scheduling optimization problem. We will test our algorithm's on the classical buffer-less benchmark and compare with the optimal solution.

Related Work

Local and global deadlock-detection in component-based systems are NP-hard [2]. Wysk et al. [3] developed a deadlock detection via integer programming in finding the optimum makespan. Specifically, they added constraints to ensure agents not release resource unless being assigned the next resource. However, integer programming method can only be used on small size problem as it would take very long time to run. Their integer programming formulation is shown below.

X_{iK}: Completion time of last operation of job i(term K);

X_{ikj}: Completion time of job i operation k on machine j;

T_{ikj}: processing time of job i operation k on machine j;

Y_{prqaj}: {o,1}: 1 if job p operation r follows job q operations on machine j; 0 if otherwise.

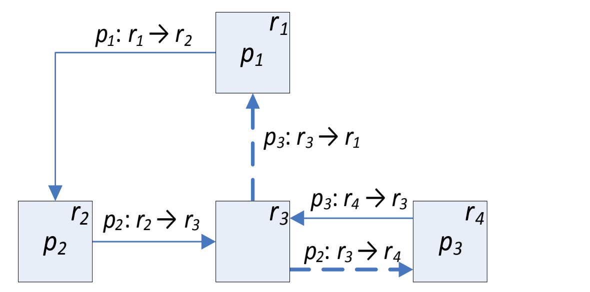

Figure 3: Deadlock II level.

***Corresponding author:** Mengmeng Chen, Department of Industrial Engineering and Management Systems, University of Central Florida, Orlando, USA, Tel: 407 823-2204 ; E-mail: CHENMM@Knights.ucf.edu

Received May 02, 2017; **Accepted** June 01, 2017; **Published** June 16, 2017

Citation: Chen M, Rabelo L (2017) Deadlock-Detection via Reinforcement Learning. Ind Eng Manage 6: 215. doi:10.4172/2169-0316.1000215

Citation: Chen M, Rabelo L (2017) Deadlock-Detection via Reinforcement Learning. Ind Eng Manage 6: 215. doi:10.4172/2169-0316.1000215

H: big positive number,

E: small positive number,

I: set of all jobs {1,2,3..N},

J: set of all machines {1,2,3..M}.

Formulation:

$$Min\ Z=\sum_{i=1}^{N} X_{iK}$$

$X_{ikj}-T_{ikj}\geq X_{i(k-1)l}$

$X_{qrj}-X_{qrj}+H(1-Y_{prqaj})\geq T_{qrj}$ $[\forall j\in J;\ \forall(p,q)\in I]$

$X_{qaj}-X_{qrj}+HY_{prqaj}\geq T_{qaj}$ $[\forall j\in J;\ \forall(p,q)\in I]$

$X_{p(r-1)l}-X_{qaj}+H(1-Y_{prqaj})\geq E$ $[\forall j\in J;\ \forall(p,q)\in I,\ l\in J]$

$X_{q(r-1)}-X_{qrj}+HY_{prqaj}\geq E$ $[\forall j\in J;\ \forall(p,q)\in I,\ w\in J]$.

In view of the modeling frameworks in the existing literature, three strategies for processing DL and corresponding research work are as follows:

- Deadlock Prevention, which organizes resource usage by each process to ensure that at least one process is always able to get all the resources it needs [4]. Mixed integer programming [5] and region theory [6] are used to solve elementary siphon [7-9] to avoid deadlock. Edsger et al., Gen and Cheng [6,10] developed Banker's algorithm in 2006. However, these algorithms have many limitations: need fixed processing numbers; no further processing can be started when executing, and also need fixed resources amount.

- Deadlock Avoidance based on the current system state and agents' future resource request, by restricting the resource allocation to avoid the deadlock. Petri Net model [3,11-14] are complete developed in this area. Di-graph model and Auto-mata model [6,9,12,15,16] were also built to handle avoidance problems. However, current avoidance algorithms are not able to handle high level DL.

- Deadlock Detection and Recovery is more focusing on quickly deadlock issue once detected. This can be completing in a couple ways [6,14], such as aborting certain action or add additional buffer. Still, detect deadlock and schedule a deadlock free path may be more convenient.

Lots of artificial intelligence and operation research effort has been applied to in scheduling problems. Zhang and Dietterich [17] were the first who applied reinforcement learning here. Mahadevan et al. [18] proposed a reinforcement learning algorithm combines different scheduling problem for the optimization of transfer-lines in manufacturing systems. Another maintenance-based approach based on simplified reinforcement learning is suggested by Zeng and Sycara [19].

Research Methodology

First let's give definition of deadlock on different levels. The first level of a DL is a set of agents that each request collects the resources held by another agent. The second-level DL is a set of agents any action will result in a first-level DL. The high level DL is a set of agents moving any action will result in a second-level DL. Figure 4 deliver a graph of how a high level DL happens.

We would like to use ranking matrix formulate this action system: $S=[s_{ij}]_{M\times N}$ stand for the state matrix of the system with i∈l={1,2,…, M} and j∈j={1,2,…, N}

Proposition 1

For agents/ jobs A={a_i, i=1,…, M_A} in the detection system, we have that,

For i=1 then no existence of b∈I, x∈*J* where $s_{a_i x}=0$, $s_{bx}=1$

For 1<i<M_A, then y∈*J* where $s_{a_i y}=0$, $s_{a_{i-1} y}=1$

For i=M_A, then z∈*J* and $s_{j_k z}=1$, $\prod_{\forall h\in I} s_{hz}\neq 0$ (Figure 5).

a_1 defined as the Last agent of $\forall a_i$∈A and denoted by TJ(a_i)=a_1, a_{M_A} defined as the First agent where $\forall a_i$∈A, denoted by $HJ(a_i)=a_{M_A}$, z defined as the head resource of $\forall a_i$∈A, denoted by HR(a_i)=z. After P_k changing every state, a new matrix will be derivative: if i ≠ *k* then $\bar{S}=[\bar{s}_{ij}]$ as $\bar{s}_{ij}=s_{ij}$, or $\bar{s}_{ij}=s_{ij}-1$ while *i=k*. using set R={r *R*={r_j, *j*∈*J*={1,2,…, M}} stand for M resources, using set *P*={p_i, *i*∈*I*={1,2,…, N}} stand for N agents/jobs, all agents follows the predefined working procedure $S=[s_{ij}]_{M\times N}$ stand for the system state at certain point. s_{ij} stand for the position when agent/job p_i being processed on resource r_i. (Figure 6.1).

If s_{ij}>0 means r_i not being occupied by agent/job p_i yet; if s_{ij}=0 then job p_i is occupying resource r_i; if s_{ij}>0 then job p_i finished tasks on r_i. Hence, the ranking matrix of state changed from S_1 to state S_2 while agent/job p_1 transferring position from r_2 to r_4,

$$S_1=\begin{bmatrix}3&0&2&1\\0&1&-1&2\\4&3&2&1\end{bmatrix}\xrightarrow{p_1:\ r_2\to r_4} S_2=\begin{bmatrix}2&-1&1&0\\0&1&-1&2\\4&3&2&1\end{bmatrix}$$

Proposition 2

The second level deadlock exist under necessary and sufficient condition as l_s⊆l where;

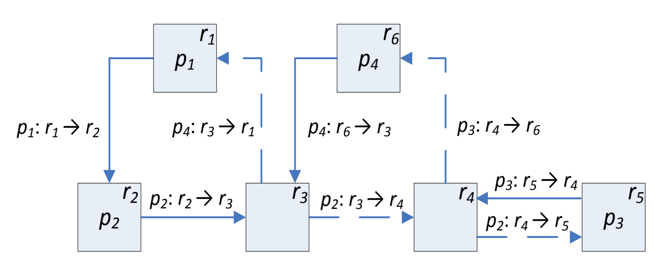

Figure 4: Deadlock III level.

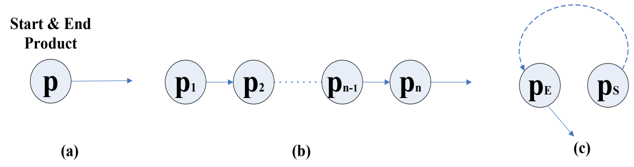

Figure 5: Simple system.

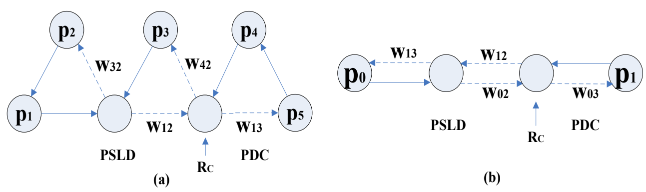

Figure 6.1: System with 2nd level deadlock.

Citation: Chen M, Rabelo L (2017) Deadlock-Detection via Reinforcement Learning. Ind Eng Manage 6: 215. doi:10.4172/2169-0316.1000215

(a) $k \in J$ as for $\forall a \in l_{s,}$ HR(a)=k.

(b) $\forall_b \in l_s$ as HJ(b)=b there exist cl_s, $\ell \in J$ that $s_{b\ell}$=2, $s_{c\ell}$=0, and $HJ(c) \neq b$ (Figure 6.2).

If apply Proposition 2 to the previous example in ranking matrix. The state matrix will be:

$$S = \begin{bmatrix} 0 & 1 & X & X \\ X & 0 & 1 & 2 \\ 2 & X & 1 & 0 \end{bmatrix}$$

Proposition 3

The high level deadlock exist under exist necessary and sufficient condition as $l_T \subseteq l$ which:

(a) x,y$\in J$ for

(i) for $\forall a l_T$, [HR(a)-x].[HR(a)-y]=0

(ii) $\sum_{\forall a \in l_T} [HR(a)-x] \cdot \sum_{\forall a \in l_T} [HR(a)-y] \neq 0$.

(b) $\forall$b$\in l_T$ as HJ(b)=b

(i) $k \in J$ as s_{bk}=2;

(ii) if c$\in l_T$ as s_{ck}=0, then $HJ(c) \neq b$; o.w., (k-x) (k-y)=0, as $d \in l_T$ and $\ell \in J$ by s_b=3, $s_{d\ell}$=0, HR(d) $\neq$ HR(b) (Figure 7).

If apply Proposition 3 to the previous example in ranking matrix. The state matrix will be:

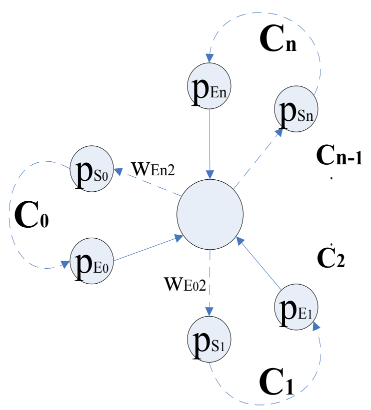

Figure 6.2: System with 2nd level deadlock.

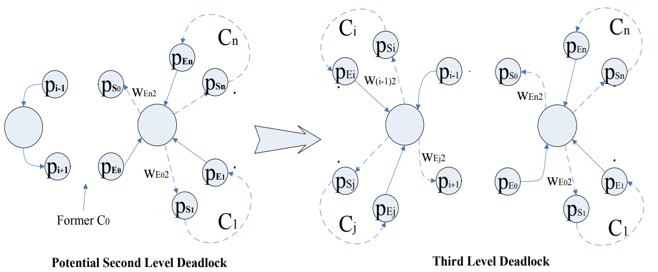

Figure 7: System with 3rd level deadlock.

$$S = \begin{bmatrix} 0 & 1 & X & X & X & X \\ X & 0 & 1 & 2 & 3 & X \\ X & X & X & 1 & 0 & 2 \\ 2 & X & 1 & X & X & 0 \end{bmatrix}$$

1. HR(1)=HR(2)=$HR(4)$=3. HR(3)=HR(5)=4 so condition (a) satisfied.
2. HJ(2)=2, HJ(3)=3, HJ(4)=4 and s_{24}=s_{36}=s_{41}=2 so condition [b(i)] satisfied.
3. Resource 4 will be required by Agent 2 in two steps and available; also have s_{25}=3, s_{25}=0 and HR(2)=3$\neq$$HR(3)$=4. Condition [b(ii)] satisfied as s_{36}=2.

3.1 s_{36}=2, s_{46}=0, and HJ(3)=3$\neq$$HJ$(4)=4.

3.2 s_{41}=2, s_{11}=0, and HJ(4)=4$\neq$$HJ$(1)=2.

As mentioned above, we consider all collaborative action teams seeking to optimize global rewards, and we assume that we can use our reinforcement learning approach to model the corresponding multi-action stochastic systems and provide a search algorithm. Therefore, there is at least one action sequence that maximizes the expected return of all movements [20-26].

Definition 1

Set $S_i \subseteq S$ be the system state of i, where si={A (π_1), A (π_2)…A (π_i)}, {A_i} denote actions will be executed, π_I denotes the policies of action. The action A_i at state I and the reward value R (ℓ_i), represent by:

$$R(\ell) = \frac{\sum_{i,k=1}^{n} (J_i + M_k)}{C_{MAX}}$$

C_{max} denotes the makespan, $\sum_{i,k=1}^{i} (J_i + M_k)$ represent the summary of jobs and machines been involved. Here P (π) represents the performance of policy π, and R (ℓ) represents the local action reward parameter.

Definition 2

Under the set $S_o \subseteq$S, the policy:

$$P(\pi) = E[R_0 \mid s_0 \in S_0, \pi] = E[\sum_{i=1}^{n} (\gamma^i R(i)) \mid \pi] = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} R(\ell)$$ is

defined as the expected local reward R(ℓ) and set discount factor γ to 1.

Evaluation

There are 40 classical scheduling benchmark problems for testing. The design of these problems adopts complex structure to increase difficulty. Additionally, if these systems are buffer-less, find scheduling will harder. Gantt chart can be drawn based on a DL-free timesheet obtained by each scheduling benchmark [27-31].

As shown in Figures 8 and 9, they present benchmark LA08 (15 × 5) and benchmark LA16 (10 × 10). We test the performance of our algorithm in this 40 benchmarks with backtracking counting's, we also compare the running time between with and without DL detection. Algorithms is written in Matlab, and the workflow of our RL algorithm is shown in Figure 10.

The results of 40 benchmark testing with our algorithm are given in Table 1. From the results, we found that:

Citation: Chen M, Rabelo L (2017) Deadlock-Detection via Reinforcement Learning. Ind Eng Manage 6: 215. doi:10.4172/2169-0316.1000215

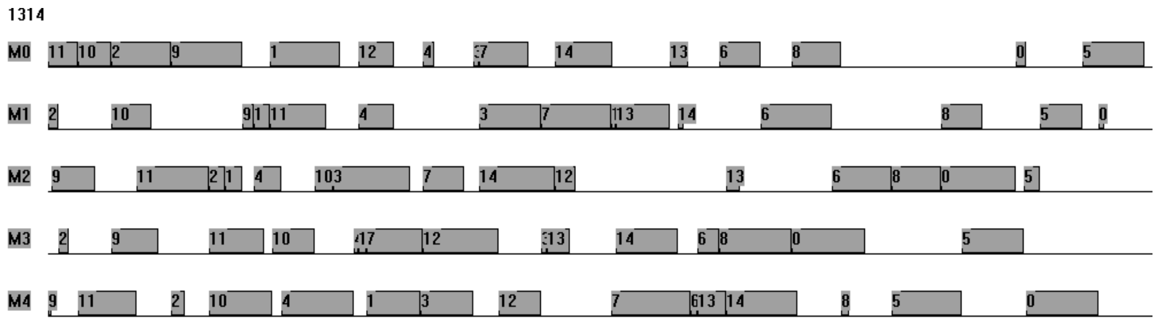

Figure 8: Optimal schedule of LA08.

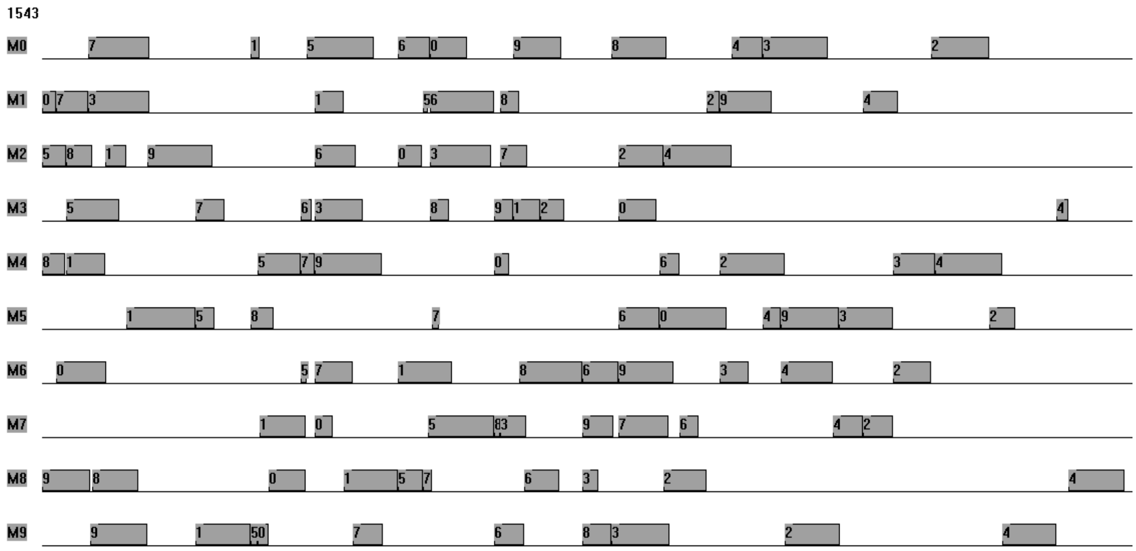

Figure 9: Optimal schedule of LA16.

(1) All 40 benchmarks can be solved via algorithm within acceptable time frame.

(2) Our results are very close to the optimal solution. These shows our policy-based RL approach is effective in reducing time and cost.

(3) Due to the system difficulty, once the system becomes larger, the number of backtracking increases. Backtracking numbers are 0 for first 15 benchmark, and the number increases as the states increases. However, for all benchmark problems, our number of backtracking used is kept at a low level.

(4) Once a DL event occurs, our scheduling algorithm can rearrange and generate a new DL-free timesheet within 1 seconds. Therefore, we can assuming that our DL-free algorithm would be applied to other similar structure systems. Additionally, under more power computation system this algorithms making itself a qualified tool for real-time operation system (Table 1).

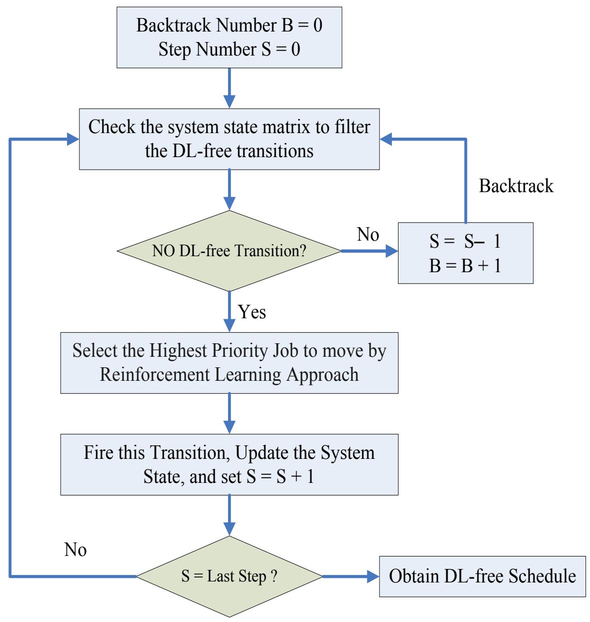

Figure 10: DL detection algorithm via RL.

Conclusion

Based on the ranking matrix, graph model and reinforcement learning, a new corresponding DL detection algorithm is proposed by us, and using that the author analyzed the general pattern of high-level DL detection problem based on discrete system, using the classical

Problem (size)	2LD		3LD		Makespan	Optimal Makespan
	Time	Backtrack	Time	Backtrack		
FT 06	<1 sec.	0	<1 sec.	0	512	512
LA 01 (10 × 5)	<1 sec.	0	<1 sec.	0	1096	1073
LA 02 (10 × 5)	<1 sec.	0	<1 sec.	0	1025	1025
LA 03 (10 × 5)	<1 sec.	0	<1 sec.	0	857	817
LA 04 (10 × 5)	<1 sec.	0	<1 sec.	0	933	827
LA 05 (10 × 5)	<1 sec.	0	<1 sec.	0	879	879
LA 06 (15 × 5)	<1 sec.	0	<1 sec.	7	1390	N/A in 48 hours
LA 07 (15 × 5)	<1 sec.	0	<1 sec.	13	1337	N/A in 48 hours
LA 08 (15 × 5)	<1 sec.	0	<1 sec.	19	1314	N/A in 48 hours
LA 09 (15 × 5)	<1 sec.	0	<1 sec.	0	1609	N/A in 48 hours
LA 10 (15 × 5)	<1 sec.	0	<1 sec.	8	1525	N/A in 48 hours
LA 11 (20 × 5)	<1 sec.	0	<1 sec.	3	1873	N/A in 48 hours
LA 12 (20 × 5)	<1 sec.	0	<1 sec.	17	1726	N/A in 48 hours
LA 0 13 (20 × 5)	<1 sec.	0	<1 sec.	14	1895	N/A in 48 hours
LA 14 (20 × 5)	<1 sec.	0	<1 sec.	0	1901	N/A in 48 hours
LA 15 (20 × 5)	<1 sec.	0	<1 sec.	0	2015	N/A in 48 hours
LA 16 (10 × 10)	<1 sec.	1	<1 sec.	16	1498	N/A in 48 hours
LA 17 (10 × 10)	<1 sec.	0	<1 sec.	113	1187	N/A in 48 hours
LA 18 (10 × 10)	<1 sec.	0	<1 sec.	12	1478	N/A in 48 hours
LA 19 (10 × 10)	<1 sec.	6	<1 sec.	31	1412	N/A in 48 hours
LA 20 (10 × 10)	<1 sec.	0	<1 sec.	0	1514	N/A in 48 hours
LA 21 (15x10)	<1 sec.	21	<1 sec.	409	2051	N/A in 48 hours
LA 22 (15 × 10)	<1 sec.	29	3 sec.	12053	1811	N/A in 48 hours
LA 23 (15 × 10)	<1 sec.	143	<1 sec.	1339	2032	N/A in 48 hours
LA 24 (15 × 10)	<1 sec.	22	<1 sec.	888	1934	N/A in 48 hours
LA 25 (15 × 10)	<1 sec.	108	<1 sec.	12430	1983	N/A in 48 hours
LA 26 (20 × 10)	<1 sec.	38	45 sec.	624349	2666	N/A in 48 hours
LA 27 (20 × 10)	<1 sec.	36	<1 sec.	221	2730	N/A in 48 hours
LA 28 (20 × 10)	<1 sec.	6	<1 sec.	799	2600	N/A in 48 hours
LA 29 (20 × 10)	<1 sec.	196	<1 sec.	8683	2621	N/A in 48 hours
LA 30 (20 × 10)	<1 sec.	23	<1 sec.	591	2774	N/A in 48 hours
LA 31 (30 × 10)	<1 sec.	33	<1 sec.	3174	3701	N/A in 48 hours
LA 32 (30 × 10)	<1 sec.	73	47 sec.	295725	3997	N/A in 48 hours
LA 33 (30 × 10)	<1 sec.	71	4 sec.	46982	3791	N/A in 48 hours
LA 34 (30 × 10)	<1 sec.	94	4 sec.	26426	3929	N/A in 48 hours
LA 35 (30 × 10)	<1 sec.	68	<1 sec.	9705	4076	N/A in 48 hours
LA 36 (15 × 15)	<1 sec.	69	Not available in 3hrs		2543	N/A in 48 hours
LA 37 (15 × 15)	<1 sec.	239	<1 sec.	1765	2800	N/A in 48 hours
LA 38 (15 × 15)	<1 sec.	339	8 min.		2301	N/A in 48 hours
LA 39 (15 × 15)	<1 sec.	35	<1 sec.	1922	2386	N/A in 48 hours
LA 40 (15 × 15)	<1 sec.	415	15 min.	8518357	2578	N/A in 48 hours

Table 1: Evaluation table.

forty benchmark problems. However due to the heavy computation, some work might took very long term, but this can be solved in time while the computation speed is exponential increasing.

This algorithm is developed under the buffer less environmental which is much more difficulty compare to real world. Therefore, it is worth believing that our algorithm should be extended to other resource sharing systems.

Based on this DL detection algorithms, relax some certain constrains new limited buffer DL detection algorithms can be developed and can be widely applied in the mechanical system, parallel computing system, and the future is quite bright.

References

1. Coulouris GF, Dollimore J, Kindberg T (2005) Distributed systems: concepts and design. Pearson education, Chicago.
2. Minnameier C (2007) Local and global deadlock-detection in component-based systems are NP-hard. Information Processing Letters, Chicago 103: 105-111.
3. Wysk RA, Yang NS, Joshi S (1991) Detection of deadlocks in flexible manufacturing cells. IEEE Transactions on Robotics and Automation 7: 853-859.
4. Dijkstra EW (2006) Een algorithme ter voorkoming van de dodelijke omarming. Chicago.
5. Chu F, Xie X (1997) Deadlock analysis of Petri nets using siphons and mathematical programming. IEEE Trans Robot Automat 13: 793-804.
6. Ezpeleta J, Colom JM, Martinez J (1995) A Petri net based deadlock prevention policy for flexible manufacturing system. IEEE Trans Robotics Automat 11: 173-184.
7. Fanti MP, Maione B, Mascolo S, Turchiano B (1997b) Low-Cost deadlock avoidance policies for flexible production systems. Int J Model Simulation 17: 310-316.
8. Fanti MP, Zhou MC (2004) Deadlock control methods in automated manufacturing systems. IEEE Trans on Systems, Man and Cybernetics - Part A: Systems and Humans 34: 5-22.

9. Gebraeel NZ, Lawley MA (2001) Deadlock detection, prevention and avoidance for automated tool shared systems. IEEE Trans on Robotics and Automation 17: 342-356.

10. Gen M, Cheng R (2000) Genetic Algorithms and Engineering Optimization. John Wiley & Sons.

11. Hsieh F, Chang S (1994) Dispatching-driven deadlock avoidance controller synthesis for flexible manufacturing systems. IEEE Trans on Robotics and Automation 10: 196-209.

12. Wu NQ (1999) Necessary and sufficient conditions for deadlock-free operation in flexible manufacturing systems using a colored Petri net model. IEEE Trans Syst Man Cyber Part C 29: 192-204.

13. Lawley MA, Reveliotis SA, Ferreira P (1998) The application and evaluation of Banker's algorithm for deadlock freebuffer space allocation in flexible manufacturing systems. International Journal of Flexible Manufacturing Systems 10: 73-100.

14. Taubin A, Kondratyev A, Kinshinevsky M (1998) Deadlock prevention using Petri nets and their unfoldings. Int J Manufact Technol 14: 750-759.

15. Reveliotis SA, Ferreira PM (1996) Deadlock avoidance policies for automated manufacturing cells. IEEE Transactions on Robotics and Automation 12: 845-857.

16. Viswanadham N, Narahari Y, Johnson TL (1990) Deadlock prevention and deadlock avoidance in flexible manufacturing system using Petri net models. IEEE Trans on Robotics and Automation 6: 713-723.

17. Yalcin A, Boucher TO (2000) Deadlock avoidance in flexible manufacturing systems using finite automata. IEEE Trans on Robot Auto 16: 424-429.

18. Xu G, Wu ZM (2004) Deadlock free scheduling strategy for automated production cell. IEEE Trans Systems Man and Cybernetics part A 34: 113-122.

19. Ramirez SA, Benhabib B (2000) Supervisory control of multiworkcell manufacturing systems with shared resources. IEEE Trans Syst Man Cybern part B 30: 668-683.

20. Reingold EM, Nievergelt J, Deo N (1997) Combinatorial.

21. Gold EM (1978) Deadlock prediction: easy and difficult cases. SIAM J Com 7: 320-336.

22. Kolonko M (1999) Some new results on simulated annealing applied to the job shop scheduling problem. European Journal of Operation Research 113: 123-136.

23. Kumaran TK, Chang W, Cho H, Wysk RA (1994) A structured approach to deadlock detection, avoidance and resolution in flexible manufacturing systems. Int J Prod Res 32: 2361-2379.

24. Li Z, Zhou MC (2004) Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems IEEE Trans on System Man and Cybernetics Part A: Systems and Humans 34: 38-51.

25. Shi XQ, Wu ZM (2005) Deadlock free scheduling method for FMSs using beam search. IEEE Inter. Conf. on System Man and Cybernetics 2: 1188-1193.

26. Xie X, Jeng M (1999) ERCN-merged nets and their analysis using siphons. IEEE Trans on Robotics and Automation 15: 692-703.

27. Sutton R, Barto A (1998) Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, USA 3: 321-360.

28. Mahadevan S, Marchalleck N, Das T, Gosavi A (1997) Self-improving factory simulation using continuous-time average-reward reinforcement learning. Proceedings of the 14th International Conference on Machine Learning, Nashville, USA, pp: 202-210.

29. Zeng D, Sycara K (1997) Using case-based reasoning as a reinforcement learning framework for optimization with changing criteria. Proceedings of the 7th International Conference on Tools with Artificial Intelligence, pp: 56-62.

30. Schneider J, Wong W, Moore A, Riedmiller M (1999) Distributed value functions. Proceedings of 16th International Conference on Machine Learning, Beld, Slovenia, pp: 371-378

31. Chen M (2013) Job shop scheduling with high level deadlock detection (dissertation, Iowa State University).