POLISH MARITIME RESEARCH Special Issue 2017 S3 (95) 2017 Vol. 24; pp. 102-109
10.1515/pomr-2017-0111

A DEEP Q-LEARNING NETWORK FOR SHIP STOWAGE PLANNING PROBLEM

Yifan Shen[1]
Ning Zhao[2*]
Mengjue Xia[1]
Xueqiang Du[2]
[1] Scientific Research Academy, Shanghai Maritime University, China
[2]Logistics Engineering College, Shanghai Maritime University, China
* corresponding author

ABSTRACT

Ship stowage plan is the management connection of quae crane scheduling and yard crane scheduling. The quality of ship stowage plan affects the productivity greatly. Previous studies mainly focuses on solving stowage planning problem with online searching algorithm, efficiency of which is significantly affected by case size. In this study, a Deep Q-Learning Network (DQN) is proposed to solve ship stowage planning problem. With DQN, massive calculation and training is done in pre-training stage, while in application stage stowage plan can be made in seconds. To formulate network input, decision factors are analyzed to compose feature vector of stowage plan. States subject to constraints, available action and reward function of Q-value are designed. With these information and design, an 8-layer DQN is formulated with an evaluation function of mean square error is composed to learn stowage planning. At the end of this study, several production cases are solved with proposed DQN to validate the effectiveness and generalization ability. Result shows a good availability of DQN to solve ship stowage planning problem.

Keywords: Deep Q-Leaning Network (DQN); Container terminal; Ship stowage plan; Markov decision process; Value function approximation; Generalization

INTRODUCTION

In Recent years more and more researchers devoted themselves to the study of marine science, port logistics and so on[1-3], cause ocean is one of the important resources for human beings. Especially in ports, researchers have made great contributions to container terminal equipment [4-7] and planning [8-10] intelligence. In container terminals, stowage plan is one of the most important and time consuming planning phase. In present time, stowage planning is mainly made by hand with computer assistance. Such manual planning management mode relies heavily on experience of planners, which costs labor and time. With automation currency in container terminal, the manual planning hinders management automation process. At the same time, container ships has been larger and larger in recent time, which increases planning labor and time consumption. Under such circumstances, stowage planning automation or intelligent stowage planning has been a critical technology to be broke through in container terminal management to optimize both cost and efficiency.

Previous studies of container ship stowage planning focus on stowage planning model and algorithms to solve this problem.

In terms of stowage planning model, Master Bay Plan and In-Bay Plan have been the mainstream. Among Master Bay Plan, Todd D S and Sen P [11] propose a Master Bay Plan model minimizing reshuffling, with trimming moment, heeling moment, ship stability and position as constraints. A GA is designed to solve this problem. Zhao N and Mi W J [12] made a multi-objective mixed integer programming model with ship stability factors and operation factors as constraints to optimize reshuffles and yard crane efficiency.

The proposed MIP model can only solve small scale problems with traditional planning solver. Moura A and Oliveira J et al [13] proposed a MIP model optimizing total transportation cost with shipping line in consideration. Amone In-Bay Plan, Avriel M and Penn M [14] proposed a MIP model minimizing reshuffles. Proposed algorithm can solve small scale problems. J.J.Shields [15] made a comparison between model solving outputs and actual loading outputs to validate proposed model. Imai et al [16-18] proposed multi-objective MIP model minimizing reshuffles. Numerical experiments reveals more binary variables and binary constraints would significantly increase complexity and significantly lower solving efficiency. Haghani and Kaisar et al [19] proposed a MIP model with turnaround time and ship parameters as constraints.

In terms of stowage planning algorithm, most researches prefer intelligent optimization algorithms. Álvarez et al [20] proposed a tabu-search algorithm with multiple initial solutions to solve the problem optimizing moving distance of stackers, shuffles and container weight distribution in ship. Numerical experiments show that proposed tabu-search algorithm can solve cases with more than 100 containers in short time while MIP solvers cannot solve cases with more than 40 containers. Kim et al [21-24] proposed beam-search algorithm to solve stowage planning problem. Y.Lee et al [25] decomposes stowage planning problem into smaller scale sub-problems using hierarchy theory to solve stowage planning problem. An Ant Colony Optimization-Tabu Search hybrid algorithm is proposed, and numerical experiment shows superiority of proposed hybrid algorithm over original independent algorithms.

These analyses shows that stowage planning studies at the moment concentrate on composing a MIP model and design a heuristic algorithm to solve the model. Such method performs well in small scale cases while has limitations such as poor performance in large scale cases and weak ability of generalization. Thus in this paper a deep reinforcement learning algorithm is proposed to solve stowage planning problem. Intelligent agent of stowage planning is trained to solve stowage planning problem efficiently and maintain better generalization.

CONTAINER SHIP STOWAGE PLANNING PROBLEM

DECISION FACTORS

In stowage planning process, several factors needs to be considered to ensure seaworthiness of container ship and improve operation efficiency.

1. Ship slot location and sequence relationship factor

To ensure efficiency during ship loading process, ship slots has relative loading sequence relationship such as slot 8401 can only be loaded when the slot right under slot 8401 which is 8201, and relative sea side slots should better be loaded before land side slots. This sequence relationship between slot locations varies between Deck Stowage Plan and Hold Stowage Plan. Deck has more constraints to ensure ship stability and operation safety.

2. Ship slot weight limit factor

Before In-Bay Planning, Master Bay Plan has preplanned allocation to suggest a weight limit for each ship slot to guarantee ship stability and securing capacity. Thus each slot has a weight range constraint.

3. Heavy-over-light limit factor

Theoretically, heavy containers should be loaded under light containers to ensure ship stability. While in actual planning, heavy containers are allowed to load over light containers if these containers weighs close. Thus, a heavy-over-light limit factor is applied to formulate this constraint.

OPTIMIZATION OBJECTIVES

1. Staircase shape sequencing in deck stowage planning

In terms of loading sequence, containers should better be loaded in stair shape, which means avoid insert a container between containers to improve loading operation efficiency and safety.

2. Minimizing reshuffles

When a container needs to be loaded before containers over itself in container yard, reshuffle is needed. Reshuffles caused by stowage plan should be minimized during planning to improve loading efficiency.

3. Minimizing yard crane shifts

When containers with adjacent planned loading sequence locates in different yard bays or even in different yard blocks, yard crane needs to shift from one bay to another to load these two containers. Unreasonable plan causes yard crane to shift back and forth to pick containers, which affects loading efficiency. Thus, yard crane shifts should be minimized to improve loading efficiency.

DEEP REINFORCEMENT LEARNING ALGORITHM FOR STOWAGE PLANNING PROBLEM

MARKOV DECISION PROCESS

L. S. Shapley first proposes Markov Decision Processes (MDP) in stochastic games research. R.Bellman then proposes dynamic programming method to solve general sequencial problem. R.A.Howard and D.Blackwell proposed general theoretical framework and effective method for MDP. A MDP is a 5-tuple (s_t, a_t, r_t, T, π), where

s_t is a finite set of states,

a_t is a finite set of actions available from state s_t,

r_t is immediate reward (or expected immediate reward),
T is the transit function from state s_t to s_{t+1},
π is the strategy or policy

The problem of MDP is to find a policy π that specifies actions that the decision maker will choose when in state s_t.

MDP FOR STOWAGE PROBLEM

MDP for stowage planning problem is formulated according to basis of MDP. Fig. 2 shows a example of stowage planning MDP.

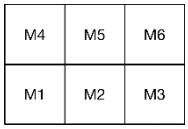

Fig. 1. Slot Scheme.

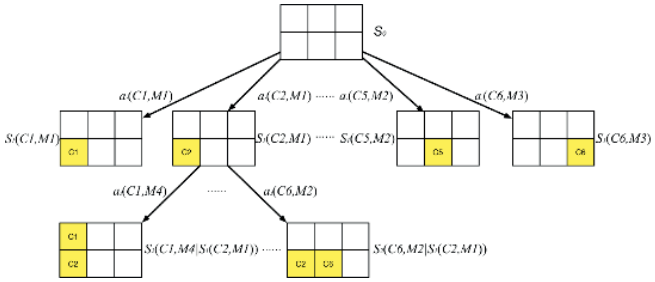

Fig. 2. MDP for Stowage Planning Problem

1. Stowage State

In stowage planning, t is stowage sequence, t=0 is the initial state when no container is loaded, t=1 is the next state when the first container is loaded, t=2 is the state when the second container is loaded, and so on. In Fig. 2, S_0 is the initial state when the whole bay is empty, S_1 is the next state when the first container C1 is stowed, and then S_2. As is shown, when in S_0, there are several available actions, which is to stow which container in which available slot. $S_1(C2,M1)$ means stow C2 into M1 slot in state S_1, $S_2(C6,M2|S_1(C2,M1))$ means two containers are stowed, first stow C2 into M1 slot and then stow C6 into M2 in state S_2.

2. Stowage Action

In stowage planning, an action is to mate a container with a slot, which means stow this container in this slot. In different state, available actions are different. In Fig. 2, when in S_0, if stowage constraints are ignored, there are 36 available actions or mate of containers to slots.

3. Stowage Reward

In reinforcement learning, reward represents the environment. In stowage planning learning, reward mainly expresses objectives and constraints. Since the result of a stowage plan is evaluated by availability, reshuffling, yard crane shifting, and these evaluations have different scales of importance, the stowage reward is as follow.

$$r_1 = \begin{cases} -500, if & stowage\ plan\ is\ available \\ 500, if & stowage\ plan\ is\ not\ available \\ 0, else & \end{cases} \quad (1)$$

$$r_2 = -10 * \phi(5) \quad (2)$$

$$r_3 = -30 * \phi(9) \quad (3)$$

Formula (1) is the reward of availability, (2) is the reward of reshuffling, (3) is the reward of yard crane shifting.

4. Stowage Planning Evaluation Function and Action Evaluation Function

$$v^{\pi}(s) = \sum_{s'} T(s' \mid s,a)[r_{t+1}(s' \mid s,a) + \lambda v^{\pi}(s_{t+1}) \mid s_t = s] \quad (4)$$

$$Q^{\pi}(s,a) = \sum_{s'} T(s' \mid s,a)[r_{t+1}(s' \mid s,a) + \lambda v(s_{t+1})] \quad (5)$$

$$V^{*}(s) = \max_{a} \sum_{s'} T(s' \mid s,a)[R(s' \mid s,a) + \lambda V^{*}(s')] \quad (6)$$

$$Q^{*}(s,a) = \sum_{s'} T(s' \mid s,a)[R(s' \mid s,a) + \lambda V^{*}(s')] \quad (7)$$

π is the stowage strategy or policy,
λ is the discount factor, which represents the influence of next stowage move to this move,
$T(s' \mid s,a)$ is the probability of taking action a to get state s' in state s,
$R(s' \mid s,a)$ is the reward of taking action a to get state s' in state s,
$v(s)$ is expected reward of taking various actions in state s, or expected reward of stowing other containers after state s,
$Q(s,a)$ is the total reward of taking action a in state s,
$V^{*}(s)$ is the maximum reward in state s,
$Q^{*}(s,a)$ is the maximum reward of taking action a in state s.

STOWAGE PLANNING FEATURES

The dimensions of different ship bays are usually different in stowage planning, and reinforcement learning needs a training set with same dimensions. Thus, stowage features are introduced to approximate different stowage states. In this research, 9 features are selected as the feature vector of a stowage state, $\Phi = \{\phi(1), \phi(2), \cdots \phi(8), \phi(9)\}$

$$\phi_i(1) = W_i / W_{\max} \quad (8)$$

$$\phi_i(2) = T_i / T_{\max} \quad (9)$$

$$\phi_i(3) = \begin{cases} (W_j - W_i) / W_{\max}, if \quad T_i > 1 \\ (W_{\max} - W_i) / W_{\max}, if \quad T_i = 1 \end{cases} \tag{10}$$

$$\phi_i(4) = P_i - S_i \tag{11}$$

$$\phi_i(5) = F_i / F_{\max} \tag{12}$$

$$\phi_i(6) = X_i - X_j / X_{\max} \tag{13}$$

$$\phi_i(7) = X_i - X_k / X_{\max} \tag{14}$$

$$\phi_i(8) = (G_i - G_j) + (G_i - G_k) / 2G_{\max} \tag{15}$$

$$\phi_i(9) = \begin{cases} 1, if \quad this\ container\ is\ located\ in \\ \qquad the\ same\ yard\ bay\ with\ the \\ \qquad previous\ one \\ 0, else \end{cases} \tag{16}$$

(8) Represents the normalized weight of selected container.

(9) Represents the normalized tier number of selected slot.

(10) Represents the normalized weight gap between selected container and the container located right under it on the ship.

(11) Represents the potential of selected match of container and slot (or action), which means number of remaining lighter container minus available ship slots above selected slot, or expression of influence of selected action to later stowage planning.

(12) Represents normalized reshuffles caused by this action.

(13) Represents normalized sequential gap between selected container with containers located left of selected.

(14) Represents normalized sequential gap between selected containers with containers located right under selected container.

(15) Represents normalized sum of sequential gap between selected container with containers located left of selected and sequential gap between selected containers with containers located right under selected container.

(16) Represents whether this container locates in the same yard bay with the previous one.

DEEP REINFOREMENT LEARNING ALORITHM FOR STOWAGE PLANNING PROBLEM

Figure 3 shows the framework of reinforcement learning or Q-Learning for stowage plan. In the initial state of learning, the intelligent agent is like a naïve planner, every action the planner take will have a reward to update $Q(s,a)$, and the agent will decide next action for next state depending on updated $Q(s,a)$, this is the iteration of reinforcement learning. Actually, the agent learns from iterations of attempts and rewards to maximize the final reward and make the policy better.

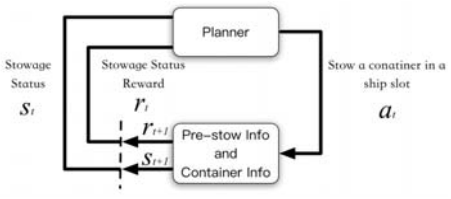

Fig. 3. Framework of Reinforcement Learning

The difference between Deep Q-Learning and Q-Learning is that the look up table is replaced by deep neuron network to update $Q(s,a)$, which enables effectiveness in super large state space scale. And the deep neuron network can be trained with minimizing lost function $L_i(w_i)$ which updates in each iteration.

$$L_i(w_i) = \mathrm{E}[\underbrace{(r + \gamma \max_{a'} Q(s', a'; w_i)}_{Target} - Q(s, a; w_i))^2] \tag{17}$$

s' is the next state, and a' is the next action. The partial in the w_i direction is in (18).

$$\nabla_{w_i} L_i(w_i) = \mathrm{E}_{s,a,r,s'}[(r + \gamma \max_{a'} Q(s', a'; w_i) - Q(s, a; w_i)) \nabla_{w_i} Q(s, a; w_i)] \tag{18}$$

Stochastic Gradient Descent is used to optimize the lost function, and the weight updates after every iteration, which is quite similar to traditional Q-Learning algorithm.

In order to approximate reward for new states that never appeared before, a evaluation function approximation function is introduced to improve generalization ability. Unlike supervised learning, reinforcement learning doesn't have known tags for training, tags are obtained through iterations. While a state and an action is updated, the change of weight for this match can affect other matches, which causes ineffectiveness of previous state and action matches, and then it causes longer training time or even failure of training. Thus, an experience replay method is introduced to prevent ineffectiveness.

Experience replay stores the experience of time t as $(\phi_t, a_t, r_t, \phi_{t+1})$ in experience history queue D, and then D is stochastically sampled as $(\phi_j, a_j, r_j, \phi_{j+1})$ to do mini-batch to update the weight. This ensures every history points are considered when updating a new data point. Experience replay stores all previous states and action in a sequence to minimize objective function when Q-Function updates.

$$L_i(w_i) = \mathrm{E}_{(s,a,r,s') \sim U(D)}[(r + \gamma \max_{b} Q(s', b; w_i) - Q(s, a; w_i))^2] \tag{19}$$

D represents a sequence of previous states and actions, $U(D)$ is a uniform distribution among experience sequence D. Experience replay based upon uniform distribution lowered data dependency to improve learning robustness.

The deep network for the stowage planning problem is designed as follows.

1. Input layer and output layer

For stowage planning problem, the input layer is a matrix of feature vector of stowage samples, the output layer is the approximate Q-value. Thus, the number of nodes in input layer is 9, the number of nodes in output layer is 1.

2. Number of hidden layers

Generally, more hidden layer makes higher precision of approximation, while more hidden layer costs more training and greater probability of over-fitting. In this case, 9 hidden layer is accepted.

3. Number of nodes in hidden layers

To avoid over-fitting and maintain better generalization ability, the number of nodes in hidden layer should be minimized when the precision is assured. Number of nodes in hidden layer is related to number of nodes in input layer, number of nodes in output layer, complexity of learning problem, transition function and sample data. Too few nodes causes poor training performance, and too many nodes causes less system error but may cause over-fitting.

4. Activation function

There are three widely used activation functions, *TanH*, *Sigmoid* and *Relu* (Rectified Linear Units). *Relu* has better training performance especially in attenuation of gradient and network sparsely. Thus, *Relu* is used as the activation function of this research.

$$Relu: f(x) = \max(0, x) \tag{20}$$

The designed deep neuron network for stowage planning problem is shown in Figure 4.

According to deep network design, DQN training algorithm is designed, pseudo code for DQN Algorithm for Stowage Planning is shown in Table 1, and flowchart in Figure 5.

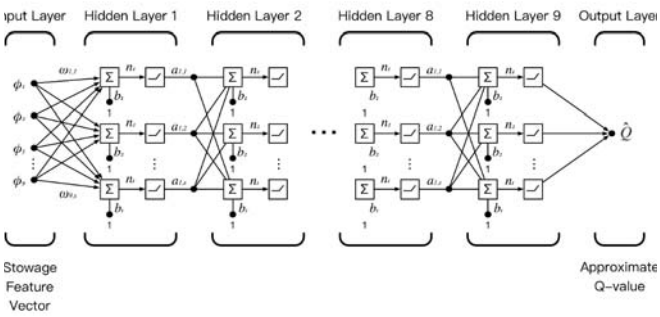

Figure 4. Deep Neuron Network for Stowage Planning Problem

Tab. 1. DQN Algorithm for Stowage Planning

DQN Training Algorithm for Stowage Planning
Initialize experience history queue D with length N Initialize $Q(s,a;w)$ with random weight w_0 For each stowage episode loop: Initialize observation sequence $s_1 = \{x_1\}$ and feature sequence $\phi_1 = \phi(s_1)$ For each step in an episode loop: Select an action to perform in state s with $\varepsilon(softmax) - greedy$ Update reward and extract feature $\phi_1 = \phi(s_1)$ Save experience tuple $(\phi_t, a_t, r_t, \phi_{t+1})$ into experience history queue D Collect samples $(\phi_j, a_j, r_j, \phi_{j+1})$ with size of random sampling mini-batch Transform sample $(\phi_j, a_j, r_j, \phi_{j+1})$ into training tuple (x_k, y_k) $x_k = \phi_j, y_k = r + \lambda \max_{a'} Q_i(\phi_{j+1}, a', w_{i-1})$ Update network weights of training set $\{(x_k, y_k)\}_m$ according to $\nabla_{w_i} L_i(w_i)$ with stochastic gradient descent Loop until end of states s Loop until end of episodes

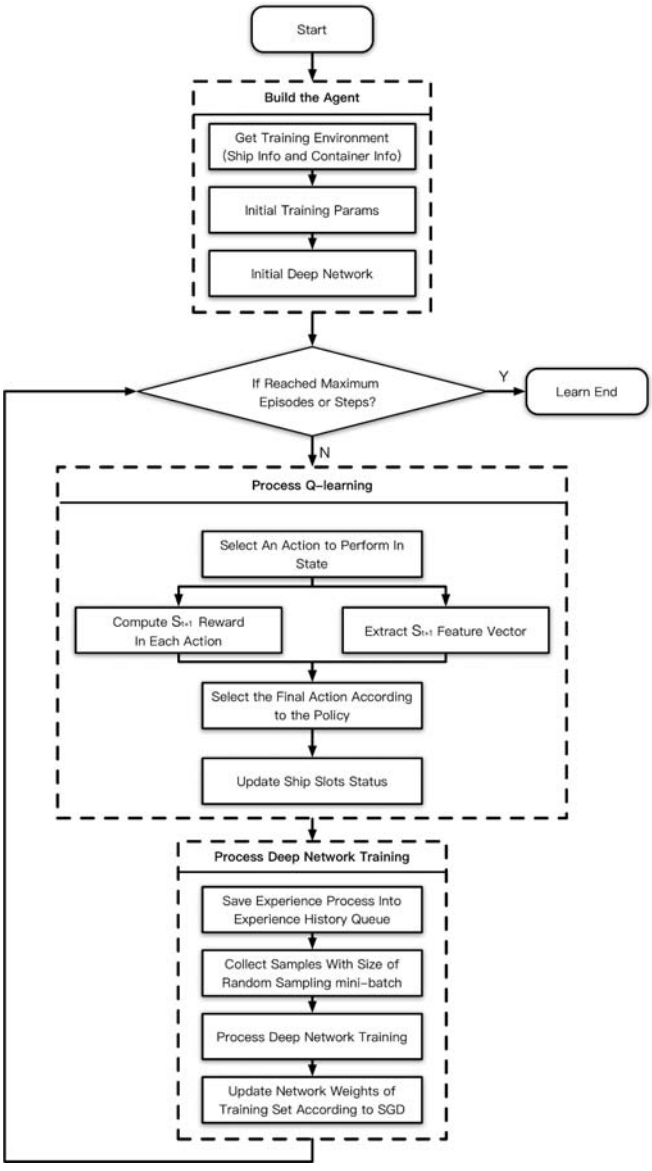

Fig. 5. Flowchart for DQN Algorithm for Stowage Planning

STOWAGE CASE STUDY OF DQN STOWAGE PLANNING

CASE DESCRIPTION

In this case, production data of Ningbo Port is used to verify proposed method. Selected ship bay has 19 slots, 19 corresponding containers locate in 4 yard bays in 2 blocks. Ship bay is shown in Figure 6, this bay has 4 tiers and 5 rows, each weight box is a slot to be stowed. Container distribution in yard is shown in Figure 7. Number inside each box in Figure 7 is the weight of each container.

Parameter setup for stowage planning is shown in Table 2 and parameter setup for DQN learning algorithm is in Table 3. Random exploration rate ε indicates that in the initial state of iterations, the random exploration rate equals to 1 to improve exploration. After each 1% of total iterations, the exploration rate decrease by a step of 0.09 to reach 0.1 when iterations finish. With this descending, the agent can focus on optimized solution gradually to converge while keeping a moderate ability of exploration.

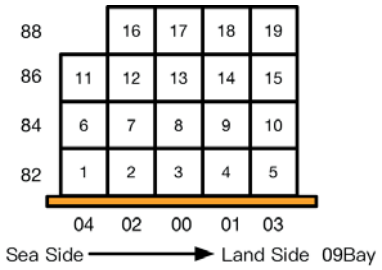

Fig. 6. Ship Bay Layout

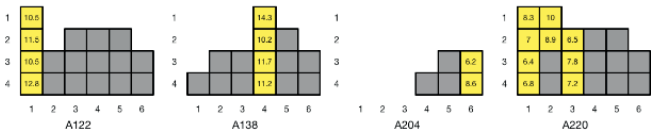

Fig. 7. Container Distribution

Tab. 2. Parameter setup for stowage planning

Heavy-over-light limit factor δ	Reshuffle weight w_1	Yard crane shift weight w_2
0.5 t	3	1

Tab. 3. Parameter Setup for DQN Learning Algorithm

Learning ratio α	Discount factor λ	Random exploration rate ε
1*10-4	0.3	1 to 0.1 with step of -0.09
Random exploration rate update internal	Experience replay depth	Number of nodes in hidden layers
1%	5000	128

STOWAGE RESULT ANALYSIS

The proposed DQN is trained with production data for 200000 iterations, which costs 2 hours and 46 mins. The trained DQN can finish the test case in 0.069 seconds, and the stowage result of the test case is as in Figure 8.

The upper left figure shows the weight distribution of stowage, the upper right figure shows the sequence of stowage. The boxes are filled with different colors to distinguish its original yard bay. In this stowage plan, 1 reshuffle and 3 shifts are needed to finish loading of this ship bay, of which 3 shifts are necessary (because there are 4 yard bays in total). The reshuffle of container with sequence 18 is unnecessary, but it is still a good solution. With all that, the effectiveness of DQN trained with production data is validated.

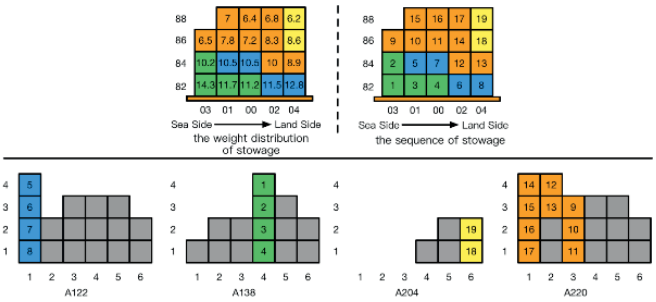

Fig. 8. Stowage result of test case

GENERALIZATION ABILITY ANALYSIS

1. Generalization of Data with Same Size

To verify the generalization of same size data, another stowage case with 19 containers is introduced. This case (case. 2) comes from a different ship of same port. Stowage result has 1 reshuffle and 4 shifts, 4 shifts of witch are all necessary. With comparison with port planners' stowage results, the stowage plan of DQN shows a good performance. Manual plan costs 121 seconds on average, while DQN can complete the calculation in 0.073s. This case study shows a good result in terms of generalization of same size data.

2 Generalization of Data with Different Size

To verify the generalization of different size data, a stowage case with 40 containers is introduced. This case (case. 3) has a big difference with the previous one both in case size and container distribution. Result of DQN of this case shows some heavy-over-light containers, while the weight gaps are all in the heavy-over-light limit. The result has 12 reshuffles and 11 shifts, 4 shifts are unnecessary. For the complexity of this case, port planners show varieties in their plans, with an average of 10.2 reshuffles and 9.6 shifts. Port planners takes 237s to make the plan while DQN costs 0.131s. Thus, DQN shows comparable ability in this case with human competitors with much better time consumption. This case study shows a good result in terms of generalization of different size data.

ROBUSTNESS ANALYSIS

In stowage planning DQN learning, robustness of algorithm refers to whether the training algorithm can get good DQN with various stowage planning cases.

In generalization analysis part, DQN trained with case. 1 is used to plan case. 2 and case. 3. To verify the robustness of proposed algorithm, case. 2 and case. 3 are used as training set to get new DQNs. Planning results of different DQNs are shown below.

Tab. 4. Training parameters and time consumption

Training Set	No. of Containers	Iterations	Training time
Case. 1	19	150000	2 h 46 min
Case. 2	19	150000	2 h 53 min
Case. 3	28	150000	4 h 21 min

Table 4 shows that these three training has same iteration setup, and with same case size, the training time is quite similar.

Tab. 5. Comparison of DQNs' planning results

Training Set	Case. 1			Case. 2			Case. 3		
Test Case	Case. 1	Case. 2	Case. 3	Case. 1	Case. 2	Case. 3	Case. 1	Case. 2	Case. 3
Reshuffles	1	1	12	2	1	12	1	1	10
Shifts	3	4	11	3	4	12	3	5	11
Training Time	0.069	0.073	0.131	0.071	0.081	0.142	0.068	0.073	0.137

As in Table 5, different test cases shows good result with different trained DQNs, and the efficiency of different DQNs are quite similar, which means influence of different training cases and test cases are negligible. Thus, the proposed algorithm has good stability and robustness.

CONCLUSIONS

In this study, a DQN and a learning method for this DQN is proposed to solve ship stowage planning problem, inovations are as follows.

1. Introduces deep learning algorithm to solve planning problem. With DQN, massive calculation and training is done in pre-training stage, while in application the planning problem can be solved in seconds

2. Objectives and constraints of ship stowage planning problem are transformed to feature vectors to extract stowage policies with deep learning algorithm automatically. Policies from data tends to have less bias than designed heuristics in previous studies.

3. Experience replay is introduced in DQN to enforce generalization and robustness of proposed algorithm.

4. Provided reference to solving planning problem in container terminals such as yard storage planning and equipment scheduling.

ACKNOWLEDGEMENTS

This research was supported by the National Natural Science Foundation of China (No.61540045), the Science and Technology Commission of Shanghai Municipality (No.15YF1404900, No.14170501500), Ministry of Education of the PR China (No.20133121110005), Shanghai Municipal Education Commission (No. 14ZZ140), Shanghai Maritime University (No.2014ycx040).

REFERENCES

1. M. Omar, S. S. Supadi. 2012. Integrated models for shipping a vendor's final production batch to a single buyer under linearly decreasing demand for consignment policy. *Sains Malaysiana* 41.3: 367-370.

2. C. Mi, Z. W. Zhang, Y. F. Huang, Y. Shen, 2013. A fast automated vision system for container corner casting recognition. *Journal of Marine Science and Technology-Taiwan*, 24(1): 54-60. DOI: 10.6119/JMST-016-0125-8

3. X. P. Rui, X. T. Yu, J. Lu, et al. 2016. An algorithm for generation of DEMs from contour lines considering geomorphic features. *Earth Sciences Research Journal*, 20(2): G1-G9, 20(2):G1-G9.

4. Y. Shen, 2016. An Anti-Collision Method of Slip Barrel for Automatic Ship Loading in Bulk Terminal. *Polish Maritime Research*, 23(s1).

5. C. Mi, Y. Shen, W. J. Mi, Y. F. Huang, 2015. Ship Identification Algorithm Based on 3D Point Cloud for Automated Ship Loaders. *Journal of Coastal Research*, 2015(SI.73): 28-34. DOI: 10.2112/SI73-006.

6. C. Mi, Z. W. Zhang, X. He, Y. F. Huang, W. J. Mi, 2015. Two-stage classification approach for human detection in camera video in bulk ports, *Polish Maritime Research*, 22(SI.1):163-170. DOI: 10.1515/pomr-2015-0049

7. C. Mi, H. W. Liu, Y. F. Huang, W. J. Mi, Y. Shen, 2016. Fatigue alarm systems for port machine operators. *Asia Life Sciences*, 25(1): 31-41.

8. Yifan S, Ning Z, Weijian M. 2016. Group-Bay Stowage Planning Problem for Container Ship. *Polish Maritime Research*, 23(s1).

9. Mengjue X., Ning Z, Weijian M. 2016. Storage Allocation in Automated Container Terminals: the Upper Level. *Polish Maritime Research*, 23(s1).

10. C. Mi, X. He, H. W. Liu, Y. F. Huang, W. J. Mi, 2014. Research on a Fast Human-Detection Algorithm for Unmanned Surveillance Area in Bulk Ports. *Mathematical Problems in Engineering*. DOI: 10.1155/2014/386764

11. D. S. Todd, P. Sen, 1997. A Multiple Criteria Genetic Algorithm for Containership Loading International Conference on Genetic Algorithms, East Lansing, Mi, Usa, July. DBLP, 674-681.

12. N. Zhao, W. J. Mi, 2008. Robust approach in stowage planning at contianer terminals. *IEEE proceeding of the 4th International Conference on Intelligent Logistic System*, 191-204.

13. A. Moura, J. Oliveira, C. Pimentel, 2013. A Mathematical Model for the Container Stowage and Ship Routing Problem. *Journal of Mathematical Modelling and Algorithms in Operations Research*, 12(3): 217-231.

14. M. Avriel, M. Penn, 1993. Exact and approximate solutions of the container ship stowage problem. *Computers & Industrial Engineering*, 25(1-4):271-274.

15. J. J. Shields, 1984. Containership Stowage: A Computer-Aided Preplanning System. *Marine Technology*, 21(4): 370-383.

16. A. Imai, T. Miki, 1989. A heuristic algorithm with expected utility for an optimal sequence of loading containers into a containerized ship. *Journal of Japan Institute of Navigation*, 80: 117–124 (in Japanese).

17. A. Imai, E. Nishimura, K. Sasaki, S. Papadimitriou, 2001. Solution comparisons of algorithms for the containership loading problem. Proceedings of the International Conference on Shipping: Technology and Environment, available on CD-ROM.

18. A. Imai, E. Nishimura, K. Sasaki, S. Papadimitriou, 2001. Solution comparisons of algorithms for the containership loading problem. Proceedings of the International Conference on Shipping: Technology and Environment, available on CD-ROM.

19. A. Haghani, E. I. Kaisar, 2001. A model for designing container loading plans for containerships. In: 80th Transportation Research Board Annual Meeting, Washington, DC, USA.

20. J. F. Álvarez, 2006. A heuristic for Vessel planning in a reach stacker terminal. *Journal of Maritime Research Jmr*, 3(1): págs. 3-16.

21. K. H. Kim, 1994. Analysis of rehandles of transfer crane in a container yard. *APORS-Conference*, 3: 357-365.

22. K. H. Kim. 1997. Evaluation of the number of rehandles in container yards. Computers & Industrial Engineering, 32: 701–711.

23. K. H. Kim, Y. M. Park, K. R. Ryu, 2000. Deriving decision rules to locate export containers in container yards. *European Journal of Operational Researc*h, 124: 89-101.

24. K. H. Kim, J. S. Kang, K. R. Ryu, 200. A beam search algorithm for the load sequencing of outbound containers in port container terminals. *OR Spectru*m, 26: 93-116.

25. Y. Lee, J. Kang, K. R. Ryu, K. H. Kim, 2005. Optimization of Container Load Sequencing by a Hybrid of Ant Colony Optimization and Tabu Search, *Natural Computation Lecture Notes in Computer Science*, 3611, 1259-1268.

CONTACT WITH THE AUTHORS

Ning Zhao

Logistics Engineering College
Shanghai Maritime University
Shanghai201306
China