# Single Vehicle Job Scheduling Using Learning Algorithm

Yongkyun Na, 005504660, cutepro@stanford.edu

## 1. Introduction & Motivation

In the manufacturing line, there is a material handling system which transfers an item from one machine to other machine. One of the simplest material handling systems is RGV(Rail Guided Vehicle) system. In this system, RGV moves only backward and forward on the guided rail and it executes the job which consists of two location information; 'From location', 'To location'. When the RGV gets a job, it moves to the 'From location' first and pickup the item from the machine and moves to the 'To location' and deposit the item to other machine. This is one job cycle.

Given N jobs in RGV system, the goal of this problem is finding the best candidate for the next job among N jobs in the real-time. Since the total processing time depends on the number of jobs and the computation of this problem is N factorial, this is NP-hard problem. The RGV has to decide the next job in the real time but generally, NP-hard problem is very difficult to find a solution in the real-time when N is large.

There are several approaches for NP-hard problem using heuristic method but I would like to solve this problem using machine learning algorithm. The idea for this problem is the relationship between previous optimal solution and new optimal solution. Intuitively, when a new job comes, we can recognize that the new optimal sequence does not fully change. So, there is unknown relation mechanism which is difficult to build a model mathematically. Thus, I will train this relation using learning algorithm and get a close optimal solution in the real-time from the learning module.

## 2. Problem Define

First, I defined a simple model for this problem. RGV has only one buffer. Equipment has input port and output port. Total length is 90m and equipment interval is

30 m as shown in Fig1. For simulation, I will fix the RGV velocity as 2.5 m/s, unloading[1] or loading[2] time is 10 seconds. In this case, new job[3] occurs starting from any equipment with fixed job cycle[4].

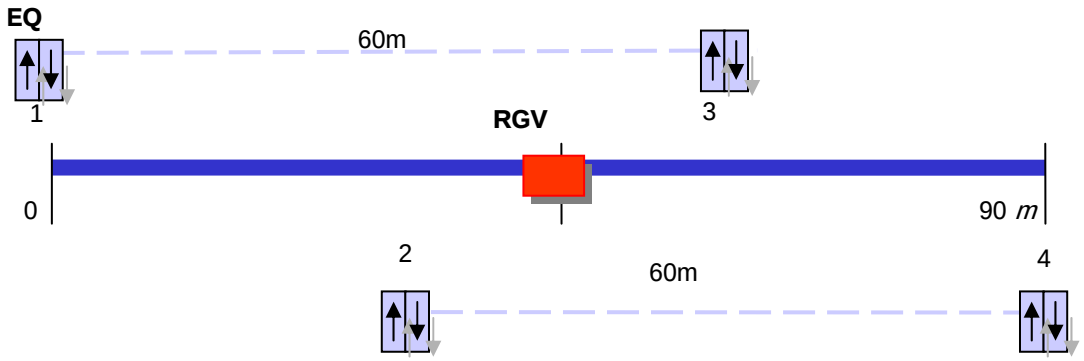


Fig.1 System Layout

When the RGV completes a job at certain time, it has to decide the next job among N jobs. Thus, we need to find out the best candidate for the next job while minimizing the total delivery time.

There are 4 equipments in the layout. The number of total possible job types is $4^2$=16 because the job consists of two locations and it also allows self-location transferring. The RGV can not have the same job type at the same time, so the maximum number of the jobs that RGV could hold is 16. Thus, we have to consider 16! =20922789888000 cases for the worst, which is impossible to compute in the real-time.

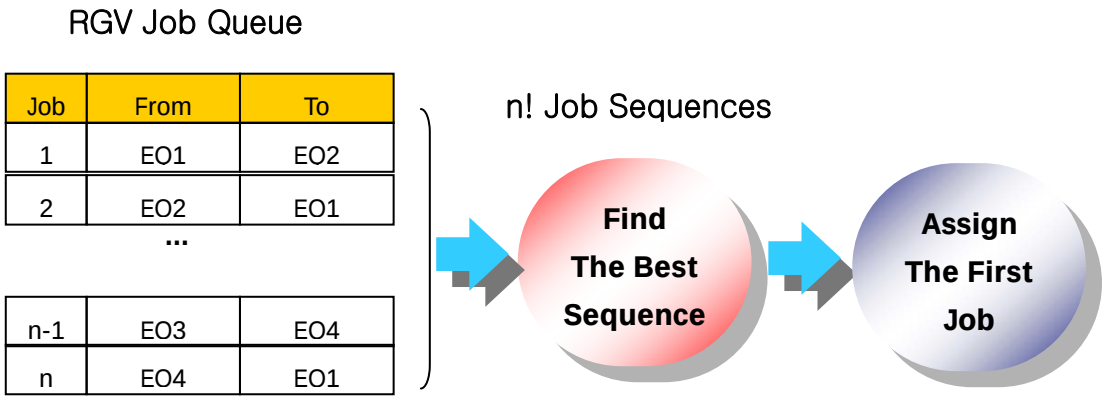


Fig2. Problem Difficulty

[1] unloading means moving transfer unit, which is a carrier or wafer, from equipment to RGV
[2] loading means moving transfer unit from RGV to equipment
[3] job means each transfer command
[4] job cycle is job occurrence interval which is 30, 25, 20,… seconds

## 3. How to solve this problem?

To solve this problem, I used the learning algorithm. The idea of this problem is the fact that the maximum number of the job types is fixed when the number of equipment is given. Thus, if we train the learning module with all possible jobs that the RGV could have, then it is possible to compute the optimal job using this module in the real-time.

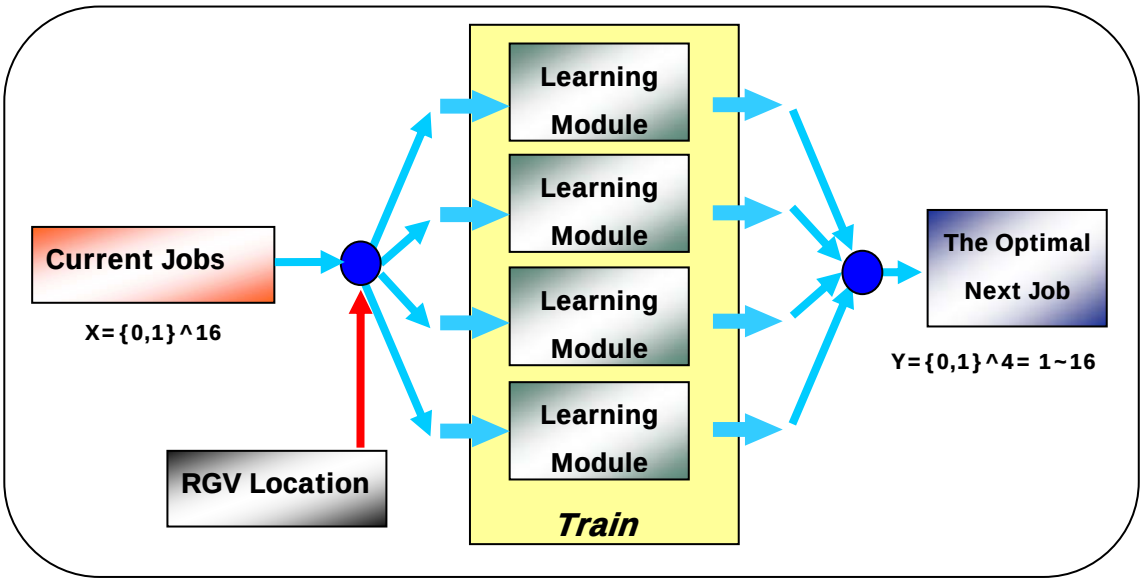


Fig3. Modular Learning Module

Even though the RGV has the same jobs in different time, the optimal next job depends on the RGV current location. Therefore, to get better learning results, I build a modular learning module. For each RGV location, we will have different learning module. In other words, since we have four equipments and RGV only stops in front of those equipments, we will have four learning modules for this layout.

To collect training data, I implemented simple DP(Dynamic Programming) algorithm , which has almost optimal sequence but not real-time. Then, I generated the training data using this algorithm for the generated sample job scenario.

The input vector $X=[0,1]^{16}$ represents the jobs that the RGV has to consider. Each bit indicates n-th job type existence. For instance, [1010000000001] means that the RGV has three jobs (type1, type3 and type16). And the output vector $Y=[0,1]^4$ represents the next job that the RGV has to take. For example, [0011] means that the RGV will work type3 job in the next.

For learning module, I tried neural network and k-nearest neighbor classification algorithm. Since the generalization error of k-nearest neighbor classification was less than neural network, I used k-nearest neighbor classification for this learning module.

## 4. Data Analysis

For the real-time algorithm comparison, I implemented FCFS (First Come First Serve), NJA (Nearest Job Assign) algorithms. FCFS is an algorithm which assigns the oldest job to RGV. This algorithm is easy to implement and normally used when the number of jobs is very small. NJA is an algorithm which assigns the nearest job from the current RGV location. Since this algorithm only needs the comparison of the current jobs, it is fast and efficient in simple layout system.

As a performance criterion, I used 'Mean Delivery Time'. The delivery time is the total time until RGV has completed a job since a job occurred.

**Delivery Time = Waiting + Total Moving + Loading + Unloading**

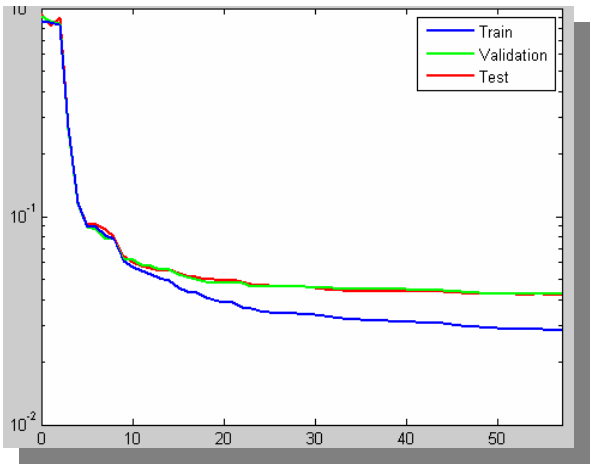


Fig3. Training Error

| Unit[sec] | FCFS | NJA | LA | DP |
|---|---|---|---|---|
| Training (100Jobs) | 1025.7 | 426.8 | 418.8 | 418.2 |
| Training (500Jobs) | 4864.4 | 1648.1 | 1645.2 | 1717.4 |
| Generalization (100Jobs) | 1101.5 | 465.8 | 509.6 | 446.5 |
| Generalization (500Jobs) | 5204.3 | 1636.7 | 1854.6 | 1624.4 |

Table.1. Algorithm Comparison

As we can see in the Table.1, for training scenario, LA (Learning Algorithm) has the best result among the real-time algorithms. The result of LA is very close to or even better than the result of DP which is not real-time algorithm. On the other hands, for

other scenarios, the NJA has better result than LA. I will discuss this more specifically in conclusion part.

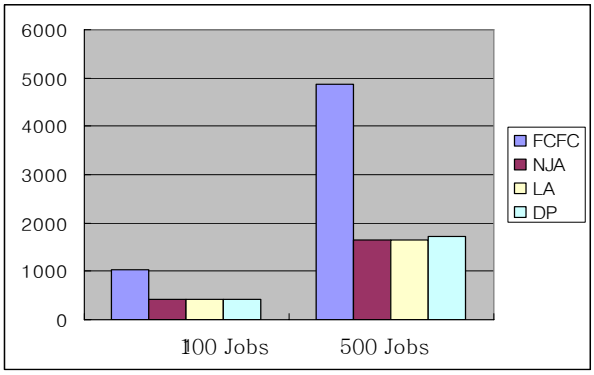

Fig4. Training Scenario Case

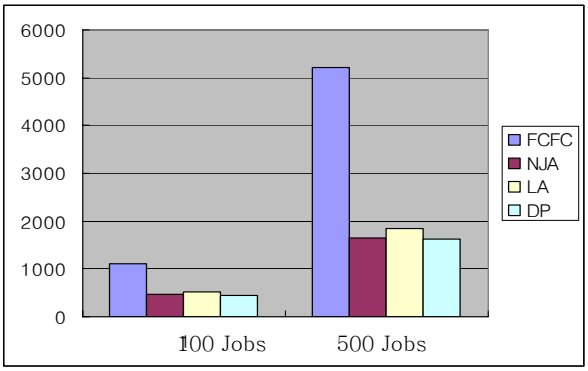

Fig5. Generalization Scenario Case

## 5. Conclusion & Future work

First of all, this learning model had large generalization error, which is 16.7%. The reason is because I used small training examples. However, this is necessary for less computation in large search space. In addition, the purpose of this algorithm is not reducing generalization error but is to have real-time computation and close optimal solution. In other words, it is ok if the performance is better than other real-time algorithm.

Despite the above argument, the result of proposed 'Learning Algorithm' method was not better than 'Nearest Job Assign' method. Finally, it turns out that 'Nearest Job Assign' method is almost optimal for this simple layout. Thus, for future work, we need to extend and apply this method to more complicated layout such as multi-vehicle, multi-buffer and more equipment. Then, the performance of the proposed algorithm will be better than NJA.

For future work, another possible area to apply learning algorithm for this problem is the next job prediction. Actually, there are several frequent job types in the real system. So, if we predict the next job patterns by learning algorithm, in the view of the global optimal, I can get better solution. Concerning the training for future job prediction, we can apply reinforcement learning through the feedback from the real result as time goes by.