J. of Advanced Engineering and Technology
Vol. 10, No. 4 (2017) pp. 453~456

강화학습기법의 동적일정계획에의 적용가능성에 대한 소고

강 장 하†
†조선대학교, 산업공학과

Note : Feasibility Study on the Reinforcement Learning for a Dynamic Scheduling Problem

Jangha Kang†
†*Department of Industrial Engineering, Chosun University, Gwangju, Korea*
(Received : Nov. 20, 2017, Revised : Dec. 15, 2017, Accepted : Dec. 22, 2017)

Abstract : The ability to dynamically reschedule jobs is core of flexible manufacturing system. Reinforcement learning, a machine learning approach undergoing development in various control systems, offers advantages in dynamic environments. This paper presents a feasibility study on the reinforcement learning for a dynamic scheduling problem.

Keyword : Flexible Manufacturing System, Reinforcement Learning, Dispatching Rules, Dynamic Scheduling

1. 서 론

지속적으로 변화하는 작업환경에 맞게 작업자의 개입 없이 실시간으로 작업 일정을 재수립하는 능력은 smart manufacturing 이라 부르는 차세대 생산환경 구축을 위한 핵심 요소이다. 또한 생산환경을 효율적으로 운영하기 위한 더 중요한 것은 납기 지연, 대기시간, 가동률 등 다양한 성능척도에 대한 최적 전략을 반영할 수 있는 능력이다. 예를 들어, 납기 일정차질을 최소화하는 전략, 또는 전체 생산시간을 최소화하는 전략, 대기 중인 생산 중 반제품 재공(Work in Process)을 최소화하는 전략들에 맞게 생산계획을 수립할 수 있는 능력은 효율적인 생산을 위해 필수적이다.

기존 생산 운영시스템들은 동적인 환경에서 주로 기계적으로 단순한 할당기준(dispatching rule)을 사용하였다. 예를 들면, 지연 최소화 (delay minimization)를 위해 EDD Rule(Earliest Due Date first)을 사용하고, 생산시간 최소화 (flow time minimization)를 위해, 최소생산시간(SPT: Shortest Processing Time first)을 사용한다[3]. 그런데, 이러한 할당기준들이 특화된 전략에만 맞게 설계되어 여러 기준을 동시에 고려하기에 너무 단순하다.

만약 생산환경 내의 과거의 작업 이력에서 작업 일정이 운영전략에 미치는 영향을 학습하여, 시스템을 운영할 수 있다면, 그러한 운영시스템은 보다 효율적으로 생산환경을 운영할 수 있을까? 그동안의 기계학습 분야의 많은 연구 결과들은 시스템을 효율적으로 운영할 수 있는 다양한 기법들을 제공한다. 그 중 가장 기대되는 기법으로 강화학습(Reinforcement Learning)이 있다. 강화학습에 대한 상세한 정보는 Sutton[4], Kaelbling [2], 또는 Bertsekas [1]에서 확인할 수 있다.

이 논문에서는 동적일정계획에의 적용가능성을 확인하기 위해, 단일 기계 작업순서 결정 문제를 위한 강화학습 모형을 소개하고자 한다. 2장에서 강화학습 기법을 소개하고, 3장에서 단일기계일정계획을 위한 모형을 소개한다. 전산 테스트 결과를 4장에서 소개하며, 마지막으로 5장에서 결론을 제시하고자 한다.

이 논문은 2016학년도 조선대학교 연구비의 지원을 받아 연구되었음
† Corresponding Author
성 명 : 강 장 하
소 속 : 조선대학교 산업공학과
주 소 : 광주 동구 필문대로 309 조선대학교
전 화 : 062-230-7126
E-mail : janghakang@chosun.ac.kr

2. 강화학습

강화학습은 설정된 목표를 이루기 위해 운영프로그램(agent)이 실행환경(environment)과 정보를 주고받으며 학습하도록 한다. 운영프로그램은 실행환경상의 다양한 상황(state)에 대해 반복적으로 다양한 행동(action)을 선택하여 궁극적으로 목표를 이루기 위한 최적의 행동을 식별한다. 선택된 행동에 따른 결과는 실행환경에 반영되고, 행동이 목표 향상에 공헌하는 정도가 계량화된다. 이 논문에서는 강화학습의 대표적인 기법인 Q-learning을 사용하였다. 다음은 Q-learning에 대한 간단한 소개이다.

2.1 상태파악

운영프로그램은 실행환경에서 얻은 다양한 정보를 해석하여 현재 상태를 파악하고 상태에 맞는 행동을 선택한다. 그리고 선택된 행동이 실행환경에 미치는 영향을 학습한다. 시점 t에 운영프로그램은 실행환경으로부터 현재의 상황을 나타내는 다양한 상태 정보를 얻어 현재 상태 s를 파악한다. 이론적으로, 상태정보는 반드시 Markov 성질을 지녀야 한다. 즉, 실행환경의 다음 시점$(t+1)$의 상태는 운영프로그램이 선택하는 행동 a와 함께 현재의 상태 s에만 영향을 받아야 하며, 과거의 상태나 행동에는 영향 받지 않는다. 실행환경이 Markov 성질을 가지고 있지 않는 경우에 Q-learning 기법을 적용하기 위해서 Markov 성질을 갖도록 상태 s를 정의하는 것이 중요한 작업이다.

2.2 행동학습

현재 상태 s가 파악되면, 운영프로그램은 취할 수 있는 대안 중에서 하나의 행동 a를 선택한다. 이 때, 선택기준은 현재 상태에서 선택한 행동이 장차 시스템의 성능척도에 미치는 영향을 계량적으로 나타내는 수치 $Q_{s,a}$에 기반을 둔다. 운영프로그램은 처음에 이 수치에 대한 정보가 없이 시작하지만 학습의 과정에서 각 상태에서 다양한 행동을 선택하고 그 결과를 관찰하는 학습과정을 통해 $Q_{s,a}$을 갱신한다.

2.3 실행환경에의 영향

현재 상태 s에서 운영프로그램이 행동 a를 선택하면, 그 결과로 실행환경은 새로운 상태 s'로 전이하며 그 과정에서 성능척도에 미치는 영향($r_{s,a}$; Reward, 소득 또는 부담)을 관찰할 수 있다. 이렇게 얻어지는 영향을 누적하여 $Q_{s,a}$을 갱신한다. 일련의 행동을 선택하는, 운영프로그램의 운영전략(policy)의 목표는 시간의 흐름에 따라 누적되는 소득의 총합을 최대화하거나 부담을 최소화하는 것이다. 끊임없이 지속되는 시스템의 경우 얻어지는 영향의 총합이 무한하게 증가 또는 감소할 수 있다. 이런 문제를 해결하기 위해서, 매 시점 총 $Q_{s,a}$을 갱신할 때 과거 시점의 영향에 할인율을 반영하여, 최근 시점의 영향의 중요도를 높이고 먼 과거시점의 중요도를 낮추는 방법을 사용할 수 있다.

2.4 실행환경에의 영향

직전 시점 $t-1$의 상태 s에서 행동 a를 선택하여 실행환경의 상태가 s'로 전이하고 그 과정에서 영향 $r_{s,a}$를 얻는 경우, 다음 시점의 Q-value는 다음과 같은 재귀적인 학습 규칙에 따라 갱신한다.

$$Q_{s,a}^{t} = Q_{s,a}^{t-1} + \alpha_{s,a} \cdot \left\{ r_{s,a} + \gamma_t \cdot MAX_{a'}[Q_{s',a'}^{t-1}] - Q_{s,a}^{t-1} \right\}$$

여기서, $\alpha_{s,a} \in [0,1]$는 학습율, $\gamma_t \in [0,1]$는 할인율, $MAX_{a'}[Q_{s',a'}^{t-1}]$는 다음 시점의 상태 s'에서 선택할 수 있는 최적의 행동에 따른 Q-value, $Q_{s,a}^{t-1}$는 상태 s에서 행동 a를 선택했을 때의 Q-value 이다.

Tsitsiklis[5]는 Q-value가 수렴하기 위해서 학습률은 다음의 조건을 만족해야 한다는 것을 증명하였다.

$$\sum_{t=1}^{\infty} \gamma_t = \infty, \quad \sum_{t=1}^{\infty} \gamma_t^2 < \infty .$$

일반적으로 학습율을 $\gamma_t = \frac{1}{t}$로 한다.

3. 일정계획 모형

3.1 대상 시스템

이 논문에서는 강화학습의 적용가능성을 확인하기 위하여 가장 간단한 형태의 일정계획문제인 단일기계일정계획문제를 분석 대상으로 한다. Figure 1은 단일기계일정계획문제의 예시이다. 공작기계는 한 시점에 하나의 작업을 제품별로 정해진 시간동안 가공한다. 기계가 가공하는 동안 도착하는 작업들은 제품별로 구분된 대기열에 나뉘어 대기한다. 신규 작업들은 지속적으로 지수분포를 따르는 임의의 시간간격마다 하나씩 납기가 지정되어 도착한다. 또한 신규 작업은 각 제품별로 균등하게 배분된다고 가정한다.

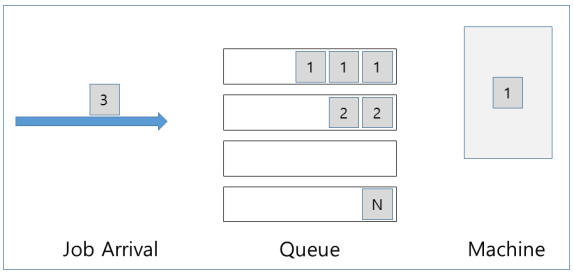

Figure 1. Illustration of one machine job scheduling

평균 납기 지연의 최소화, 대기열의 길이의 최소화,

그리고 평균 생산시간의 최소화를 생산시스템 운영 목표로 한다.

운영프로그램은 공작기계가 하나의 작업을 완료한 시점에 다음 작업을 지정한다. 직전 작업과 다음 작업이 동일한 제품이 아닐 경우, 작업 설정이 변경되어야 하며 이 때 설정 변경 시간이 소요된다.

설정 변경 시간을 무시할 수 있는 경우, EDD (Earliest Due Date first) rule은 최대지연을 최소화 한다고 알려져 있으며, 설정 변경 시간이 고려된 경우, SST (Shortest Setup Time first) rule이 주로 사용된다. 자세한 내용은 Pinedo[3]을 참조하라.

3.2 Q-learning

실행환경이 운영프로그램에게 제공하는 상태 정보에는 공작기계가 직전에 가공한 제품, 각 제품별 대기열의 길이뿐만 아니라 시스템 내의 각 작업들의 납기까지의 잔여일정이 포함된다. 그러나 위 항목들을 모두 고려하여 상태를 정의할 경우, 정의되는 실행상태는 무한하다. 이에, 다음 정보만을 이용하여 간략하게 상태를 정의한다.

a) 직전 가공 제품,
b) 각 제품별 대기 작업의 존재 여부,
c) 각 제품별 대기 중인 납기지연 작업의 존재 여부.

위 정보만을 이용하여 상태를 정의할 경우, 제품의 개수가 n일 때 전체 상태의 개수는 $n \times 3^n$이 된다. 각 상태에서 선택 가능한 행동들은 대기 작업이 존재하는 제품으로 한정된다. 운영프로그램이 상태 s에서 행동 a를 선택한 후, 납기 지연이 존재하는 제품 개수의 감소량을 $r_{s,a}$(소득, Reward)로 정의한다.

확률적으로 변하는 시스템을 대상으로 할 때 Q-learning 과정에서 Q-value가 수렴하기 위해서는 운영프로그램이 각각의 상태들을 충분히 많이 방문하여야 한다. 각 상태들에 대해서 충분하게 다양한 행동들을 선택해서 학습하지 않고 현재의 Q-value에만 의존해서 다음 행동을 선택하는 경우에 전체 최적 전략을 찾는 것은 기대할 수 없다. 이러한 문제를 해결하기 위해서 ϵ-greedy policy가 사용된다. ϵ-greedy policy란 학습의 과정에서 행동을 선택할 때 $1-\epsilon$확률로 가장 큰 Q-value($Q_{s,a}$)를 갖는 action을 선택하고 ϵ의 확률로 실행 가능한 임의의 행동을 선택하여 현재 Q-value가 낮은 행동들도 충분히 실행하여, 학습할 수 있도록 한다.

4. 실험 및 결과 분석

4.1 실험 상세

할당 기법과 강화학습 기법에 의한 일정계획을 비교하기 위해 전산 프로그램으로 모의 실행환경을 구현하였다. 공작기계가 하나의 작업을 완료하는 시점에 실행환경은 운영프로그램에 정보를 전달한다. 운영프로그램은 전달된 정보를 바탕으로 다음 작업을 선정하여 실행환경이 다음 작업을 진행하도록 한다.

할당 기법으로는 SST (Shortest Setup Time first) rule을 사용하였다.

Q-learning에서는 초기 Q-value를 다음의 2가지 기준으로 생성하였다.

a) zero 기준: 모든 (s,a)에 대해서 0 값을 할당
b) negative setup time 기준: 모든 (s,a)에 대해서 $-ST(a',a)$를 할당.

여기서 a'는 직전 생산 제품이고 $ST(a',a)$는 제품 a'생산 후 제품 a 생산 시 발생하는 설정변경 시간을 의미한다. 일반적인 Q-learning은 zero 기준으로 초기 Q-value를 생성한다. 이 논문에서는 학습시간의 단축을 위해 negative setup time을 사용하여 그 결과를 분석하였다.

4.2 결과 분석

각 학습(episode)은 시스템 내에 작업이 없는 상태에서 시작하여 10,000시간 작동하며 주요 성능치의 평균값을 기록하였다. Q-learning은 ϵ이 0.1인 ϵ-greedy policy를 사용하였으며, 총 1,000개의 문제(episode)에 대해 학습하였고, 학습 과정에서 매 10번의 문제(episode)마다 동일한 테스트 문제(episode)를 풀어 Q-value의 수렴성을 확인하였다. 이 작업은 2가지 기준의 초기 Q-value에 대해 동일하게 수행하였으며, 그 결과는 Figure 2와 같다.

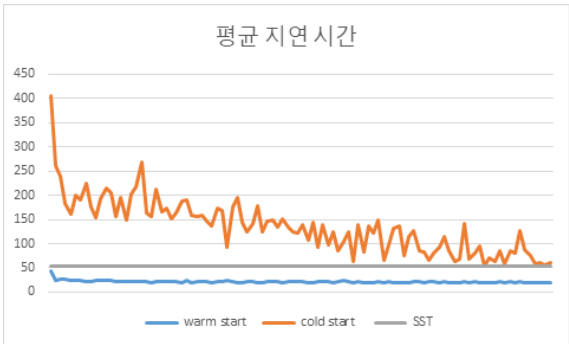

Figure 2. 결과 분석: 수렴

Figure 2에서 "Cold Start"항목은 zero 기준으로 초기 Q-value를 생성한 경우의 결과이고 "Warm Start" 항목은 negative setup time 기준으로 초기 Q-value를 생성한 경우의 결과이다. "SST"는 SST rule을 적용한 결과이다.

zero 기준으로 초기 Q-value를 생성한 경우, 꾸준하게 평균 지연시간이 감소하지만 1,000번의 학습 동안 SST rule에 따른 할당보다 나은 해를 얻지 못하였다. 그러나 negative setup time 기준으로 초기 Q-value를 생성한 경우, 처음부터 SST rule에 따른 할당보다 나은 해를 제공하고 대략 200번째 학습 이후에는 안정적으로 SST rule에 따른 할당대비 절반 이하

의 지연을 보여주고 있다.

Negative setup time 기준으로 초기 Q-value를 생성하고 이후 1,000번의 학습을 진행한 이후 SST rule에 따른 할당과 그 결과를 비교하기 위해 10개의 문제에 대해 계획을 각각 수립하였다. Figure 3은 두 방법으로 얻은 해의 평균 지연시간을 비교한 결과이고, Figure 4는 평균 대기열의 길이를 비교한 결과이다. 두 가지 기준 모두 SST rule에 의한 할당은 안정적이지 못한 결과를 보여주고 있다. 반면에 Q-learning 방법은 10개 문제 모두에 대해서 일정한 성능치를 보여주고 있다. 특히 평균지연시간, 평균 대기열 두 가지 기준에서 SST rule에 의한 할당보다 월등한 결과를 보여주고 있다.

Figure 3. 결과 분석: 평균지연시간

Figure 4. 결과 분석: 평균 대기열의 길이

5. 결론

강화학습은 실시간 대응이 중요한 다양한 환경에서 그 가능성이 연구되고 있으며, 이미 많은 성공사례를 보여주고 있다. 이에 이 논문에서는 실시간 대응이 필요한 동적일정계획 문제에 대해 적용가능성을 확인해 보고자 동적일정계획문제에 대해 Q-learning 모형을 개발, 구현하여 기존의 할당 기법과 그 성능을 비교 평가해 보았다. 가장 간단한 생산환경인 단일기계일정계획문제를 분석 대상으로 하여, 상태, 행동, 소득 등 Q-learning 모형의 기본 구조를 설계하고, 학습을 진행하였다. 그 결과 기존에 널리 사용하던 할당 기법을 능가하는 성능을 확인할 수 있었다. 이러한 결과는 동적일정계획문제에 대한 강화학습기법의 사용가능성을 보여준다.

이 논문에서는 가장 간단한 환경에서의 적용가능성을 확인하였다. 따라서 향후의 연구에서는 보다 복잡한 제조 시스템에 사용가능한 강화학습 모형을 개발하는 것이 중요 과제가 될 것이다.

참고문헌

1. Bertsekas, D. P., and Tsitsiklis, J. N. Neuro-dynamic programming, Athena Scientific (1996).
2. Kaelbling, L. P., Littman, M. L., and Moore, A. W. “Reinforcement learning: A survey.“ J. Artif. Intell. Res., **4**, 237-285 (1996).
3. Pinedo, M., Scheduling: Theory, Algorithms, and Systems, Prentice Hall (1995).
4. Sutton, R. S., and Barto, A. G., Reinforcement learning-An introduction, MIT Press (1998).
5. Tsitsiklis, J., “Asynchronous stochasitc approximation and Q-learning,” Machine Learning, **16**(3), 185-202 (1994).