Journal of Convergence for Information Technology
Vol. 9. No. 12, pp. 124-132, 2019
e-ISSN 2586-4440
DOI : https://doi.org/10.22156/CS4SMB.2019.9.12.124

병렬설비를 위한 주기적 일정계획

주운기
선문대학교 산업경영공학과 교수

Periodic Scheduling Problem on Parallel Machines

Un Gi Joo
Professor, Department of Industrial and Management Engineering, Sun Moon University

요 약 일정계획은 오프라인(offline) 일정계획과 온라인(online) 일정계획으로 구분할 수 있고, 본 논문은 온라인 상황에서 병렬설비의 주기적 일정계획 수립문제를 다룬다. 도착시간(ready time)이 다른 여러 작업들에 대해 makespan을 최소화하기 위한 작업 일정계획 알고리즘 개발이 목적이다. 이를 위해 각 설비에서의 작업처리 순서는 ERD(Earliest Ready Date) 규칙에 따른 순서가 최적임을 밝혔다. 각 설비 별 배정 작업도 결정해야하는 병렬설비 문제를 위해서는 혼합정수계획모형(MIP)을 이용하는 알고리즘을 제시하였다. 개발한 알고리즘의 유용성과 성능분석을 위해 수치 예를 활용하여 오프라인 일정계획과 비교하였다. 비교분석 결과, 오프라인 일정계획에 비해 매우 빠른 시간에 일정계획을 수립할 수 있음을 보였고, 주기시간의 감소를 통한 makespan의 단축 가능성을 보였다. 본 논문의 주기적 일정계획 방법은 계획수립을 위한 시간이 매우 작으므로, 설비 및 작업의 수가 많은 온라인 환경에서도 활용할 수 있다. 더불어서 스마트공장이나 블록체인 플랫폼에서의 작업일정 수행을 위해 활용될 수 있을 것으로 기대한다.

주제어 : 주기적 일정계획, Makespan, ERD, 혼합정수계획모형, 온라인, 블록체인

Abstract Scheduling problems can be classified into offline and online ones. This paper considers an online scheduling problem to minimize makespan on the identical parallel machines. For dynamically arrived jobs with their ready times, we show that the sequencing order according to the ERD (Earliest Ready Date) rule is optimal to minimize makespan. This paper suggests an algorithm by using the MIP(Mixed Integer Programming) formulation periodically to find a good periodic schedule and evaluates the required computational time and resulted makespan of the algorithm. The comparition with an offline scheduling shows our algorithm makes the schedule very fast and the makespan can be reduced as the period time reduction, so we can conclude that our algorithm is useful for scheduling the jobs under online environment even though the number of jobs and machines is large. We expect that the algorithm is invaluable one to find good schedules for the smart factory and online scheduler using the blockchain mechanism.

Key Words : Periodic Scheduling, Makespan, ERD, MIP formulation, Online, Blockchain

*Corresponding Author : Un Gi Joo(ugjoo@sunmoon.ac.kr)

Received October 28, 2019
Revised December 5, 2019
Accepted December 20, 2019
Published December 28, 2019

1. 서론

고객 수요의 다변화 및 개인화생산(mass personalization)에 대한 요구가 증가함에 따라 일정계획 수립방법 개발의 필요성은 증대하고 있다. 본 논문은 각각 도달시간(ready time)을 가진 여러 작업들에 대해, 동일한 성능의 병렬(parallel)설비에서 작업을 수행하기 위한 주기적(periodic) 일정계획 수립방안을 제시한다. 주기적 일정계획은 온라인(online) 작업 환경이나 블록체인(blockchain) 플랫폼에서 작업일정 수행을 위해 필요한 일정계획 방안이다.

1.1 연구 배경 및 필요성

일정계획은 일정계획 수립 시점에 처리할 작업들에 대한 정보를 정확히 알고 있는 지 여부에 따라 온라인 일정계획과 오프라인(offline) 일정계획으로 구분할 수 있다.

온라인 일정계획은 일정계획 시점에 일정계획 수립 대상 작업들에 대한 정보를 정확히 모르는 상황에서의 일정계획으로, 작업정보는 작업들의 처리가 요청된 이후에 확정된다. 이들 작업들을 위해서는 현재 시점에서 일정계획을 수립한 후, 일정계획 대상 작업에 대한 정보 정확도가 높은 시점에 다시 일정계획을 수립(re-scheduling)하는 재일정계획 수립 방법을 이용한다. [1, 2]. 그러나 재일정계획 방식의 온라인 일정계획은 불확실한 정보를 이용하여 일정계획을 수립하므로, 일정계획 효율감소와 하나의 작업에 대한 일정계획을 여러 번 수립함에 따른 계획수립 시간증가 문제를 해결할 필요가 있다. 이러한 작업정보의 불확실성 문제와 재일정계획 수립에 따른 문제는 주기적 일정계획(cycle scheduling)으로 해결할 수 있다.

주기적 일정계획은 시간에 따라 도착(작업처리 요청)하는 작업들을 일정 시간(주기시간)마다 모아서, 작업정보를 알고 있는 작업들에 대해서만 일정 계획을 수립한다. 각 주기시간에는 동일한 작업들이 순환(cycle)적으로 처리되거나 서로 다른 작업들이 주기적(periodic)으로 처리될 수 있다. 각 주기시간에 서로 다른 작업들을 처리하는 주기적(periodic) 일정계획 문제는 개인화생산 체계에서 자주 발생한다. 이를 위해서는 각 주기시간 별 일정을 각각 수립해야 하므로, 순환적인 일정계획에 비해 계산복잡도가 크게 증가하는 문제가 있다. 따라서 주기적 일정계획 수립을 위해서는 계산시간의 과대 문제를 해결할 수 있는 연구가 필요하다.

계산시간의 과대 문제는 다중설비를 이용하여 완화할 수 있다. 병렬구조의 다중설비에서는 전체 작업에 대한 일정 계획을 중앙의 하나의 일정계획 수립자(scheduler) 대신 여러 곳에서 나눠서 일정계획을 수립해도 된다. 이와 같은 분산일정 계획은 클라우드 컴퓨팅 환경에서 많이 활용되고 있다 [3,4]. 그러나 클라우드 컴퓨팅에서의 병렬 구조 설비(자원)에 대한 일정수립 연구에서도 온라인 환경을 위해서는 주기적 일정계획에 대한 연구는 필요한 상태이다.

온라인에서 처리를 요구하는 작업들에 대해 다중설비를 이용하여 주기적으로 작업을 처리하는 것은 블록체인(blockchain)을 이용하는 서비스에서 볼 수 있다. 블록체인은 주기적으로 생성한 블록들을 관련자 모두가 체인형태로 분산 저장하는 기술로, 하나의 블록은 여러 개의 거래(처리) 내역들의 정보를 포함하고 있다. 블록은 비트코인(bitcoin) 경우는 10분, 이더리움(Ethereum) 경우는 약 14초마다 하나씩 생성하고, 이미 생성된 블록은 관련자 모두에게 공개되고 체인화되므로, 생성된 블록 내용을 수정하기는 어렵다. 또한 생성된 블록 간 순서는 해쉬코드를 이용하여 유지되므로 모든 거래(처리) 순서도 수정되지 않는 구조를 가지고 있다. 따라서 블록체인의 공유를 통한 투명성, 분산처리, 다수결 기반의 블록생성을 통한 무결성 등의 장점을 활용하기 위해 금융거래, 공공서비스, 물류(공급망 관리)분야 등에 적용되고 있다 [5]. 블록체인 기술의 적용 분야가 다양해지고 있기는 하지만, 공급망 관리 외의 생산계획 및 통제를 위해 적용한 사례는 찾기가 어렵다. 생산 계획 및 통제 분야에서 블록체인을 이용하기 위해서는 블록체인의 주기적 블록 생성, 생성된 블록의 수정 불가능, 구성원들의 분산화된 블록 생성 구조를 갖는 작업 일정계획 수립 방안이 필요하다.

1.2 선행연구

온라인 환경에서는 작업 정보의 불확실성 문제를 해결하고, 하나의 작업에 대한 여러 번의 일정계획을 수립에 따른 일정변경과 계획수립 시간 (준비시간)의 증가 문제를 개선하는 방안이 필요하다. 이를 해결하기 위한 기존의 주기적 일정계획(cycle scheduling)에 대

한 연구는 전체 작업을 일정크기의 묶음(batch)단위로 반복 생산하는 상황에서, 하나의 배치를 생산하는데 필요한 최대시간을 주기시간(cycle time)으로 관리하고, 주기시간 내에 생산 가능한 작업 조합 중에서 목적에 적합한 조합을 찾는 연구가 주로 진행되었다 [6]. 주기적 일정계획은 작업정보의 불확실성 문제를 해결할 수 있고, 재일정계획 수립에 따른 작업 별 일정변경 문제가 발생하지 않는다는 장점이 있다. 그러나 기존의 주기적 일정계획에서는 하나의 배치 생산시간인 주기시간 별 작업일정이 동일하다는 제약이 있다. 논문 [7]이 다룬 유지보수를 고려한 단일(single)설비 일정계획은 각 주기별 구성 작업이 다른 일정을 수립하지만, 작업 가능량은 주기기간 내에 처리할 수 있는 양이어야 하는 제약이 있다. 따라서 온라인 환경에서 각 주기별 서로 다른 작업을 처리하고, 주기 별 작업시간 제약이 완화된 다중(병렬)설비 일정수립을 위해서는 연구가 필요한 상태이다.

일정계획은 최적화 대상인 척도에 따라서 최적 일정계획의 특성 변화가 매우 크다. 예로, 단일설비에서의 makespan 최소화를 위한 일정계획은 용이하게 수립할 수 있지만, 병렬 구조의 다중설비에서는 모든 작업이 시점 0 에 모두 도달해 있는 (도달시간이 모두 0인) 작업들에 대해서도 NP-complete 임이 알려져 있다 [8]. 따라서 이를 위한 해법으로 발견적 해법(heuristic algorithm) 개발을 주로 하고 있는데, 논문 [9]는 모든 작업들에 대한 정보를 사전에 알고 있는 상황(오프라인)에서 이용가능한 발견적 해법을 제시하였지만, 주기적 일정계획(cycle/ periodic scheduling)을 다룬 연구는 찾기 어렵다.

다중설비를 이용하는 분산일정 계획은 클라우드 컴퓨팅 환경에서 많이 활용되고 있고, 일정계획 척도로는 makesapn이 가장 많이 이용되고 있다 [3]. 특히 클라우드를 구성하는 클라이언트 수가 많은 경우에는 일정계획의 성능이 전체성과에 중요한 역할을 한다는 것을 밝혔다[4]. 그러나 클라우드 컴퓨팅에서의 병렬 구조 설비(자원)에 대한 일정수립 연구에서도 주기적 일정계획에 대한 연구는 필요한 상태이다. 분산처리는 블록체인에 대한 연구에서도 찾을 수 있다. 논문 [10]과 [11]이 IoT(Internet Of Things)의 보안 취약성 문제 해결을 위한 방안으로 블록체인 기술을 활용할 수 있음을 밝혔고, 논문 [12]는 제조 공급망에서의 제조 로트의 정보와 흐름 및 거래 이력 추적을 위해 공공(public) 블록체인을 사용할 수 있음을 보였다. 또한, 논문 [13]은 스마트 공장의 IoT 등의 자원접속 권한 제어를 위해 블록체인 기술을 이용할 수 있음을 보였다. 이와 같이 블록체인 기술의 적용 분야가 다양해지고 있기는 하지만, 공급망 관리 외의 생산계획 및 통제를 위해 적용한 사례는 찾기가 어렵다. 생산 계획 및 통제 분야에서 블록체인을 이용하기 위해서는 블록체인의 주기적 블록 생성, 생성된 블록의 수정 불가능, 구성원들의 분산화된 블록 생성 구조를 갖는 작업 일정계획 수립 방안이 필요하다.

1.3 연구 목적

본 연구는 병렬설비에서의 주기적 일정계획을 수립하는 방법을 개발하는 것을 목적으로 한다. 각각의 도착시간에 작업처리 요청된 여러 작업들을 일정한 주기시간마다 모아서 makespan 최소화를 위한 작업일정을 수립하는 알고리즘을 제시한다.

기존의 주기적 일정계획의 제약인 주기 별 동일한 작업일정 및 처리될 수 있는 총 작업시간 제한 문제를 해결하기 위해, 본 논문은 각 주기별 구성 작업이 모두 다른 경우에도 적용할 수 있고, 하나의 주기 내에서 처리되는 총 작업시간에 대한 제약이 없는 일정계획을 수립한다. 이러한 주기적 일정계획 수립 알고리즘은 기존의 병렬설비에서의 makespan 최소화 연구에서는 찾기 어렵다. 본 연구에서 제시한 알고리즘은 일정계획을 주기적으로 수립하며, 이전 주기에서 수립한 일정 계획은 수정이 불가한 상황을 다룬다. 기존의 온라인 일정계획은 한번 수립한 일정계획에 대해 현 상태의 정보를 반영하도록 재계획을 수립하지만, 본 논문에서는 한번 수립한 일정계획은 변경을 하지 않는다는 차이가 있다. 본 논문에서 제시하는 알고리즘의 주기적인 일정계획 수립, 수립된 일정의 수정 불가성, 병렬설비에서 분산적으로 일정계획 수립하는 특징은 온라인 작업환경이나 블록체인 플랫폼에서의 작업일정 수립에도 활용할 수 있을 것으로 기대한다. 본 논문의 제 2장에서는 본 논문에서 다루는 문제를 자세히 기술한다. 제 3장에는 일정계획 수립을 위한 알고리즘을 제시하고, 제시한 알고리즘의 성능분석에 따른 알고리즘 활용 방안을 제안한다. 마지막으로 제 4장에 논문의 의의와 시사점 및 추후 연구 과제를 기술하였다.

2. 작업 일정계획 모델

주기적 일정계획을 위한 주기시간을 T라고 할 때, 주기시간 T 동안에 도착한 n개의 작업을 M개의 동일한 병렬설비를 이용하여 처리하고자 한다. 여기서, 모든 설비는 고장이 나지 않고, 한 번에 하나의 작업만 처리 가능하며, 한번 처리를 시작한 작업은 그 작업을 모두 마칠 때까지 다른 작업을 못한다고 가정한다. 각 작업 j는 r_j의 시점에 도착하고, t_j의 처리시간을 필요로 한다. 각 설비 i에서의 작업처리 시작이 가능한 가장 빠른 시점을 기호 C_{i0}로 표시하기로 하고, 일정계획 수립을 통해 결정해야 하는 결정 변수로, C_{ik}를 설비 i에서 k번째로 처리되는 작업의 완료시간(completion time)이라고 하자. 그리고 X_{ijk}를 설비 i에서 작업 j가 k번째로 처리 될 때에만 1의 값을 가지는 0-1 정수 변수라고 하자, i=1,2,...,M; j=1,2,..., n; k=1,2,...n. 전체 작업 일정계획의 makespan을 $C_{\max}$라고 할 때, 현재 주기의 T 시간동안에 작업 처리 요구된 작업들의 makespan을 최소화하는 작업 일정계획을 찾는 본 논문의 연구문제는 다음의 선형 0-1 혼합정수계획(MIP)으로 모형화할 수 있다.

$$\text{Min } C_{\max} \quad (1)$$

$$\text{s.t. } C_{\max} \geq C_{ik},\ i=1,2,...,M;\ k=1,2,...,n \quad (2)$$

$$C_{ik} \geq \sum_{j=1}^{n} X_{ijk} r_j + \sum_{j=1}^{n} X_{ijk} t_j,\ i=1,2,...,M;\ k=1,2,...,n \quad (3)$$

$$C_{ik} \geq C_{i,k-1} + \sum_{j=1}^{n} X_{ijk} t_j,\ i=1,2,...,M;\ k=1,...,n \quad (4)$$

$$\sum_{j=1}^{n} X_{ijk} \leq \sum_{j=1}^{n} X_{ij,k-1},\ i=1,2,...,M;\ k=2,...,n \quad (5)$$

$$\sum_{j=1}^{n} X_{ijk} \leq 1,\ i=1,2,...,M;k=1,2,...,n \quad (6)$$

$$\sum_{i=1}^{M}\sum_{k=1}^{n} X_{ijk} = 1,\ j=1,2,...,n \quad (7)$$

$$X_{ijk} = 0 \text{ 또는 } 1,\ i=1,2,...,M;\ j=1,...,n;\ k=1,...,n \quad (8)$$

위의 MIP 모형의 제약식의 총수는 5nM + n 개이고, 결정변수의 개수는 n^2M 개의 0-1변수와 nM+1개의 실수 변수로 구성된다. 따라서 최적해를 위한 계산시간에 작업 개수 n의 영향이 큰 모형이다.

3. 주기적 일정계획 수립 알고리즘

최적 일정계획의 특성을 분석하기 위해 우선 설비가 하나인 경우의 분석을 하였다. 설비가 1개인 단일 설비 문제의 경우, 다음과 같이 ERD(Earliest Ready Date) 규칙의 일정 수립을 통해 최적의 일정계획을 수립할 수 있다.

성질 1. 단일설비의 경우, ERD 규칙에 따라 작업 순서를 결정하면, makespan이 최소화된다.

ERD 규칙은 도착시간이 빠를수록 빨리 작업처리를 시작하는 규칙을 말한다. 즉, 주어진 n개의 작업에 대해 $r_1 \leq r_2 \leq \cdots \leq r_n$ 의 관계가 만족되도록 작업처리 순서를 결정하는 것이 ERD 규칙에 따른 작업 순서이다. 이러한 작업 순서의 최적성 (성질 1)은 인접한 두 작업의 작업 순서 변경에 따른 작업 완료 시간의 증감을 이용하여 쉽게 증명할 수 있으므로 자세한 증명은 생략한다.

각 설비 별 수행할 작업들 배정이 주어진 경우에는, 성질 1에 의해 각 설비에 배정된 작업들의 순서는 ERD가 최적이다. 그러나 병렬설비 일정계획 문제는 설비별 작업 배정문제 해결도 필요하므로, 모든 작업의 도착시간이 0이라 해도 NP-complete이고, 처리할 작업의 수와 병렬설비의 개수가 증가할수록 최적해를 찾는데 걸리는 시간은 급격하게 증가한다[8].

3.1 일정계획 수립 알고리즘

성질 1을 이용하기 위해, 처리해야 하는 n개의 작업을 ERD 규칙에 따라 재배열하였다고 가정하는데, 이는 온라인 일정계획 환경에서는 자연스런 것이다. 일정계획 알고리즘을 구성하기위해 주기 p의 주기시간을 T_p로 정의하자, p=1,2,...,P.

단계 0: p=1라 하고, T_0=0으로 둔다.

단계 1: $\sum_{t=0}^{p-1} T_t$~ $\sum_{t=1}^{p} T_t$ 동안 도착한 작업들에 대해, 식 (1)~(8)의 MIP 문제를 풀어서 각 설비 별 makespan을 구한다.

단계 2: MIP 모형에서 식 (4)의 C_{i0} 값을 단계 1의 makespan 값과 $\sum_{t=1}^{p+1} T_t$ 중 큰 값으로 두고, p=p+1로 하여 더 이상 처리할 작업이 없을 때까지 단계 1을 반복한다: C_{i0} = Max{단계 1의 설비 i의 makespan, $\sum_{t=1}^{p+1} T_t$}.

단계 1은 전체 작업에 대한 일정수립 문제를 여러 연속된 주기별 문제로 나눠서 해결한다. 각 주기별 문제 간 연계는 단계 2와 같이 주기시간과 식 (4)의 C_{i0}를 이용하는데, C_{i0}를 직전 주기시간 문제의 makespan 결과 값을 이용하여 설정한다.

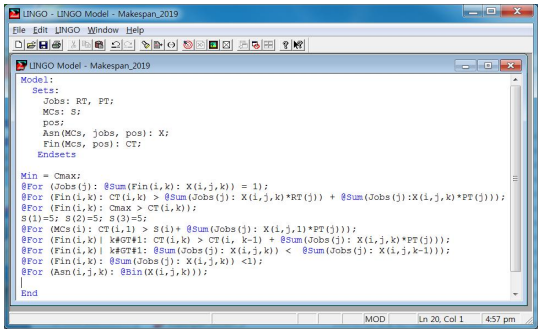

Fig. 1. LINGO for the MIP formulation

3.2 알고리즘 사용 예

3.1장의 알고리즘의 유용성을 보이기 위해 n=16, M=3인 수치 예를 사용하였다. 임의의 수치 예를 위해 난수(random number)를 이용하는 별도의 소프트웨어를 C++로 개발하였는데, 이를 위한 난수 범위는 다음과 같다. 도착시간이 범위 0~20을 가지는 일양분포(uniform distribution) U[0, 20]을 갖고, 처리 시간은 범위 1~10의 일양분포 U[1, 10]을 갖는 16개의 작업을 가지는 문제 하나를 Table 1과 같이 임의 생성하였다. 이에 대해 단계 1의 MIP 문제풀이가 필요하고, 식 (1)~(8)의 MIP 문제를 위해 Fig. 1의 LINGO 모형을 개발하여 Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, Windows 7 PC에서 수행하였다[14].

본 논문에서 제시하는 알고리즘은 주기시간 T_p에 대해, $\sum_{t=0}^{p-1} T_t < r_j \leq \sum_{t=1}^{p} T_t$ 관계의 도착시간을 가지는 작업들을 대상으로 주기 별로 일정계획을 수립한다. 주기시간 T_p가 주기 p별로 달라도 적용 상에 문제가 없지만, 편의를 위해 주기에 관계없이 주기시간이 모두 T=5인 경우를 고려한다. 먼저 (0, 5] 기간에 도착한 Table 1의 처음 6개의 작업에 대해 시점 5에 일정계획을 수립한다. 작업 1~6에 대한 makespan 최소화를 위한 일정을 식 (1)~(8)의 MIP 문제를 풀어서 Fig. 2 (a)에서 보는 바와 같이 수립하는데, 0~T 동안의 초기유휴시간(initial idle time)이 발생하고 이때의 makespan은 설비 1 및 설비 2에서의 작업완료 시점인 14로, 다음 주기 시작 시점인 2T=10을 초과한다. 다음 주기 일정계획 수립 시점인 10에는 (5, 10]기간에 도착한 (r_7, t_7) = (6,3)과 (r_8, t_8)= (9,2)의 두 작업 7과 작업 8에 대해, 설비 별 작업 처리 시작 가능한 가장 빠른 시점을 각각 C_{10} = max {2T, 14} = 14, C_{20} = max {2T, 14} = 14, C_{30} = max {2T, 11} = 11로 설정하여 MIP 문제를 풀면, Fig. 2(b)와 같은 일정계획을 얻을 수 있다. 설비 별 작업처리 가능한 가장 빠른 시점을 C_{10} = 14, C_{20} = 14, C_{30} = 11로 설정함으로써 과거 주기에서 수립한 일정계획은 수정하지 않는다. 유사하게, 시점 3T= 15에 (10, 15] 동안 도착한 Table 1의 (r_9, t_9) = (13,4)와 (r_{10}, t_{10})= (15,1)의 두 작업 9와 10에 대해 C_{10} = max{3T, 14} = 15, C_{20} = max{3T, 14} = 15, C_{30} = max{3T, 16} = 16로 설정하여 makespan 최소화를 위한 일정계획을 수립하면, Fig. 2(c)와 같다. 여기에서 보는 바와 같이, 작업 9는 도착시간이 13이지만 일정 수립시점이 3T=15이므로, C_{10} = 15가 되어 1만큼의 중간유휴시간이 발생한다. 마지막으로 4T=20 시점에 (15, 20] 동안 도착한 작업 11~16에 대해, C_{10} = C_{20} = C_{30} = 20로 하여 MIP 문제를 풀면, Fig. 2(d)와 같은 일정이 생성된다. 따라서 Table 1의 도착시간이 다른 16개의 작업에 대해 일정계획 수립 주기를 T=5로 하는 경우, Fig. 2(d)와 같은 makaspan이 31인 일정을 수립할 수 있다.

Table 1. Numerical Example for Parallel Machines

	$(r_j,\ t_j)$ of jobs
jobs arrived in (0,5]	(1,1), (1,1), (2,5), (3,8), (5,4), (5,5)
jobs arrived in (5,10]	(6,3), (9,2)
jobs arrived in (10,15]	(13,4), (15,1)
jobs arrived in (15,20]	(16,4), (17,5), (17,8) (17,7), (19,6), (20,3)

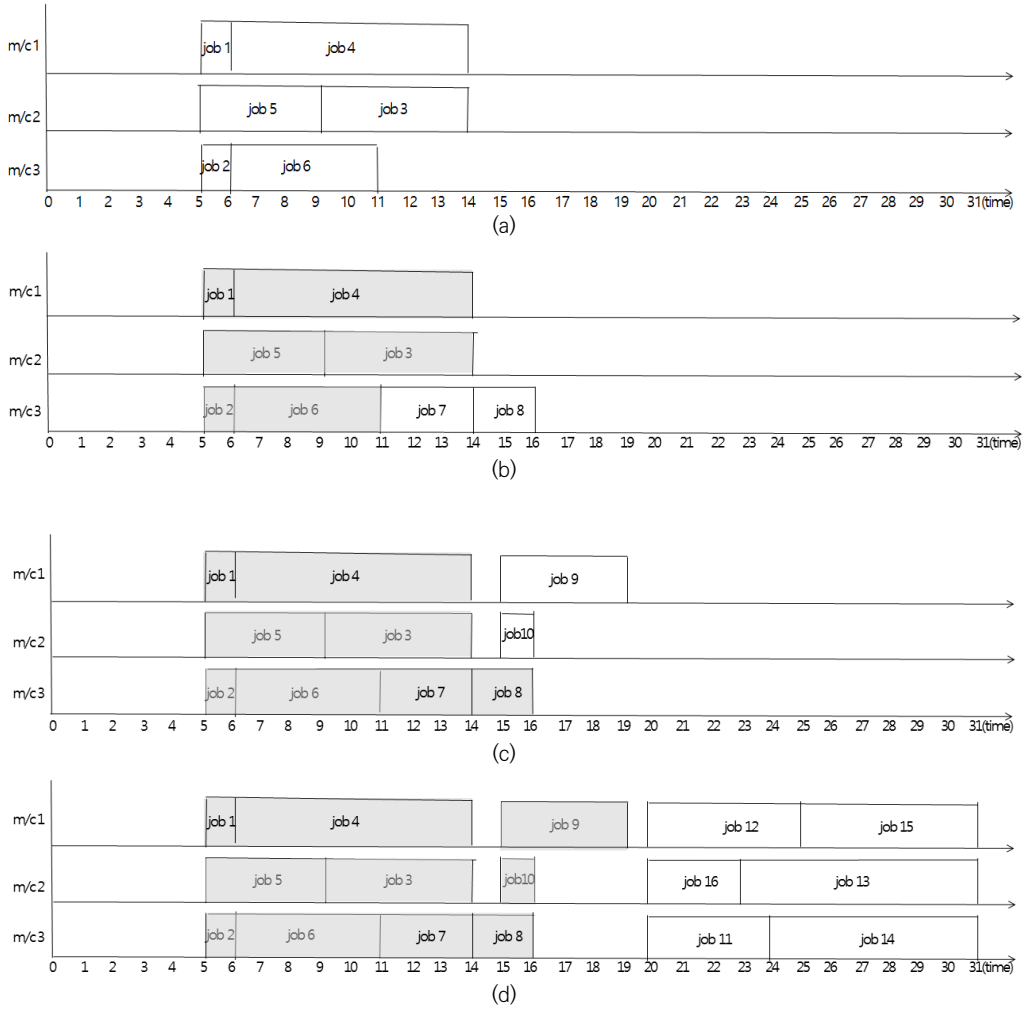

Fig. 2. Optimal Schedule under T=5

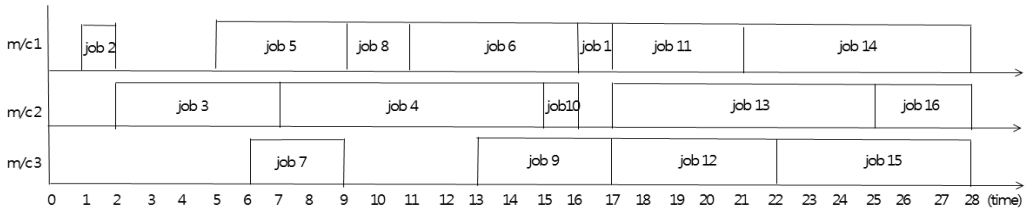

Fig. 3. Optimal Offline Schedule

3.3 알고리즘 활용 방안

3.3.1 알고리즘 성능분석

기존의 온라인 주기적 일정계획 알고리즘은 주기별 작업 일정이 동일하고 주기 내 작업처리 가능량이 제한이 있어서 본 논문의 알고리즘과 직접 비교가 어렵다. 따라서 제 3.1장에서 제시한 알고리즘의 성능을 기존의 오프라인(offline) 일정계획 수립 방법과 비교 하였다. 오프라인 일정계획은 작업에 대한 정보를 시점 0에 모두 알고 있는 경우의 일정계획이므로, 주기적 일정계획에서 주기시간이 0, T=0, 인 특수한 경우로 볼 수 있다.

Table 1의 문제에 대한 최적 오프라인 일정계획은 Fig. 3과 같이 makespan=28 인 일정이다. 그러나 이를 위해서는, Table 2에서 보는 바와 같이, 1시간 41분 3초 (6,063초)의 계산 시간이 소요되었으므로, 수립한 일정계획은 1시간 41분 3초 (준비시간) 이후에나 사용 가능한 일정계획이다.

주기적 일정계획은 Fig. 2(a)와 같이 T 만큼의 초기 유휴기간이 발생하고, 주기 p에서 작업처리 가능한 시점이 단계 2와 같이 max{주기 p-1에서의 makespan, p*T}로 결정되므로, Fig. 2(c)와 (d)에서 보듯이 중간유휴시간이 발생할 수 있다. 이러한 유휴시간이 makespan을 길어지게 할 수 있으므로, 유휴시간 감소를 위해 짧은 주기 시간 설정을 고려할 수 있다. Table 2는 Table 1의 문제에 대한 기존의 오프라인 일정계획과 T 값에 따른 본 논문의 주기적 일정계획의 계산시간과 makespan 값을 보여 준다. T=5일 때의 계산시간은 Table 1의 첫 번째 주기인 0~5 기간에 도착한 6개의 작업에 대해서만 1초(sec.)가 소요되었고, 나머지 주기에서는 LINGO로는 계산시간 측정이 안 되어서 전체 계산에 1초의 시간이 걸렸다. 반면에 오프라인 일정계획을 위해서는 상대적으로 매우 긴 6,063초가 소요되었다. T가 작아지면 일정계획 수립 빈도가 증가하므로 한 번에 다룬 작업의 수가 감소하면서 makespan 값이 단축될 수 있는데, T=1, 3인 경우에는 계산시간 측정이 안 되었고, T=1일 때 makespan 값이 30으로 감소되었다.

Table 2. Evaluation the Algorithms

	offline scheduling	Ours		
		T=1	T=3	T=5
computation time (sec.)	6,063	0	0	1
makespan	28	30	31	31

알고리즘 사용 방법 설명을 위해 제 3.2장에서 난수를 이용하여 임의로 발생한 하나의 수치 예를 이용하였지만, 본 논문에서 다루는 문제는 도착시간 및 작업시간의 수치 값(instance)에 관계없이 NP-complete 임이 증명된 문제이므로, 다른 수치 예에 대해서도 계산시간과 makepan 성능이 유사하게 나올 것으로 판단된다. 따라서 본 논문에서 제시하는 일정계획 수립 방안의 장점은 다음과 같다고 할 수 있다. 첫째, 온라인 환경에서의 재일정계획 수립이 필요 없다. 처리할 일부의 작업에 대한 정보를 확인 후 일정계획을 수립하고, 수립한 일정계획은 수정 없이 실행하여 온라인 환경에서의 일정변경과 계획수립 시간을 감소할 수 있다. 둘째로, 전체 문제를 여러 개의 작은 문제로 나눠서 해결하는 방법이므로, NP-complete 문제의 계산시간 과대 문제를 해결할 수 있다. 따라서 처리할 작업이나 설비의 수가 많은 경우에도 이용 가능한 방법이다. 셋째로, 주기시간의 감소는 makespan 의 개선을 유도하지만, 이로 인한 전체 계산시간 증가는 크지 않은 기법이다. 따라서 작업의 도착시간 특성에 따라 적절한 주기시간을 설정하여 일정계획을 수립할 수 있다.

3.3.2 블록체인 연관성 검토

본 논문에서 제시한 작업일정 계획 알고리즘을 Table 3과 같이 블록체인의 주기적 블록 생성, 생성된 블록의 수정 불가능, 구성원들의 분산화된 블록 생성 개념과 비교할 수 있다.

Table 3. Comparison to Blockchain

Blockchain	Ours
transactions	jobs
block	sub-schedule
block time	period time
blockchain	schedule
decentralized computers	parallel machines

블록체인은 PTP(Peer-To-Peer) 구조의 분산 컴퓨팅 환경에서 참여자간 거래(transaction) 요청들에 대해, 참여자들이 스스로 해당 거래의 이상 유무 확인을 하여 공유하는 방식이다. 여기서 거래의 확인은 채굴시간을 포함한 블록시간(block time) 동안 확인 요청한 거래들을 대상으로 수행된다. 전자화폐의 경우, 이들 거래를 확인하여 하나의 블록(block)으로 생성한 후, 체인화(blockchain) 하는 참여자가 블록생성의 보상으로 수수료나 전자화폐를 받는다. 본 논문의 알고리즘은 주

기시간(채굴시간) 동안 처리를 요청한 작업들을 대상으로 일정계획(블록)을 수립한다. 여기서, 일정계획 수립을 위해 병렬설비(분산제어 시스템)가 각자 작업 일정계획을 수립할 수 있고, 이 중 하나의 일정계획을 선택하여 이를 일정계획(sub-schedule)로 확정한다. 현재 주기 작업들을 위해 확정된 일정계획(sub-schedule)은 직전 주기까지 수립한 일정계획(schedule)에 Fig. 2와 같이 덧 붙여서, 현재 주기까지의 일정계획을 수립(블록체인 생성)하고, 이를 다른 설비와 공유하여 임의의 일정수정을 방지한다. 이러한 정보공유는 작업처리를 위한 모든 설비가 자신의 설비 외의 다른 설비에서의 작업 진행 상태 및 작업 부하정도 정보를 알 수 있게 하고, 스마트공장(smart factory)을 위한 기반으로 활용하는 경우, 온라인 환경에서 관리비용 감소와 빠른 일정수립을 할 수 있다. 더구나, 블록체인의 스마트계약(smart contract) 기능을 도입하여 제조 일정수립에 이용하면, 수립한 일정에 따라 작업 처리한 제품에 대한 대금 지연지급이나 대금 지급조건 변경과 같은 국내 중소 제조업의 어려움 해소에도 도움이 될 수 있다.

4. 결론

본 논문은 각각 도착시간에 따라 작업처리를 요청하는 다수의 작업들을 병렬설비를 이용하여 처리하는 경우, makespan을 최소화하는 주기적 작업 일정 수립 문제를 다루었다.

각 설비 별로 배정된 작업은 ERD 규칙의 작업 순서가 최적임을 밝혔다. MIP 모형을 이용하여 주기 별 makespan 최소화를 위한 일정계획을 수립하고, 이들 주기적 일정계획을 이용하여 전체 일정계획을 수립하는 알고리즘을 제안하였다. 본 논문에서 제시한 알고리즘은 재일정계획 수립이 필요 없으므로, 온라인 환경에서의 일정변경을 안하고 계획수립 시간도 감소할 수 있는 장점이 있다. 두 번째 장점으로, 전체 문제를 여러 작은 문제로 나눠서 해결하므로, NP-complete 문제의 계산시간 과대 문제를 완화할 수 있는 방안이다.

본 논문에서 제시한 알고리즘은 도착시간 및 작업시간을 미리 알 수 없는 온라인 환경에서도 활용할 수 있고, 블록체인 기술을 생산계획 및 통제에 활용하는 방안을 제시했다는 데에 의의가 있다. 일정계획은 작업의 특성, 설비의 구성 형태 및 척도에 따라 최적해의 특성 변화가 큰데, 이는 본 연구 문제에 대한 알고리즘의 계산시간 과대의 한계와 작업정보를 작업주문을 받기 전에 미리 알 수 없는 어려움에 따른 기존의 오프라인 알고리즘 적용 상 문제에서도 확인할 수 있다. 고객 수요의 다변화로 다품종 소량생산방식에 경쟁력을 갖추는 것이 중요해지고 있다. 본 논문은 작업 특성과 다양한 생산방식에 맞춘 적절한 알고리즘 개발의 중요성을 시사한다. 본 논문의 알고리즘 활용을 통해 분산 일정계획 수립 및 설비 간 정보 공유를 통한 스마트공장에서의 온라인 작업일정 수행과 블록체인 플랫폼에서의 관리비용 감소와 계획수립시간 감소, 그리고 스마트계약의 이점을 기대한다.

주기적 일정계획 방안은 주기시간 만큼의 초기유휴시간이 존재하는 일정계획이 수립된다는 단점이 있다. 그러나 이러한 주기시간은 온라인 환경에서의 재일정계획 수립의 필요성을 제거한다. 더구나 짧은 주기시간 설정을 통해 makespan 개선을 할 수 있다. 제시한 알고리즘의 활용 가능성을 보이기 위해 계산시간 및 makespan 측면에서 성능을 비교분석하였지만, 하나의 예제를 기반으로 한 것이라는 한계가 있다. 따라서 최적 오프라인 일정계획과의 평균성능 분석 등 추가적인 성능 비교분석이 필요하다. 그러나 오프라인 알고리즘과의 성능 분석은 계산시간 과대 문제 때문에 작업 및 설비 수가 작은 경우로 한정해야 한다는 한계가 있다. 따라서 PSO(Particle Swarm Optimization) 알고리즘과 같은 메타휴리스틱(meta-heuristic) 알고리즘의 개발도 필요하다. 그리고 흐름생산 공정(flow-shop)이 포함된 환경에서의 일정계획 방안이나 makespan 외의 다른 척도를 위한 일정계획 수립 알고리즘 개발도 필요한 상태이다.

REFERENCES

[1] S. Albers & M. Hellwig. (2012). Semi-online Scheduling Revisited. *Theoretical Computer Science, 443,* 1-9.

[2] D. Gupta & C. T. Maravelias. (2019), Online Scheduling: Understanding the Impact of Uncertainty. *IFAC PapersOnLine, 52-1,* 727-732.

[3] L. F. Bittencourt, A. Goldman, E. R. M. Madeira, N. L. S. da Fonseca & R. Sakellariou. (2018), Scheduling in Distributed Systems: a Cloud Computing Perspective. *Computer Science Review, 30,* 31-54.

[4] A. R. Arunarani, D. Manjula & V. Sugumaran. (2019). Task Scheduling Techniques in Cloud Computing: a Literature Survey. *Future Generation Computer Systems, 91,* 407-415.

[5] L. Yang. (2019). The Blockchain: State-of-the-art and Research Challenges. *Journal of Industrial Information Integration.*
DOI : 10.1016/j.jii.2019.04.002.

[6] E. Levner, V. Kats, D. A. L. de Pablo & T. C. E. Cheng. (2010), Complexity of Cyclic Scheduling Problems: a State-of-the-art Survey. *Computer and Industrial Engineering, 59,* 352-361.

[7] P. Perez-Gonzalez & J. M. Framinan. (2018). Single Machine Scheduling with Periodic Machine Availability. *Computer and Industrial Engineering, 123,* 180-188.

[8] A. H. G. Rinnooy Kan. (1976), *Machine Scheduling Problems - Classification, Complexity and Computations*, Martinus Nijhoff.

[9] U. G. Joo. (2018), Makespan Minimization Scheduling Problem with Energy-efficient Turning On/Off Mechanism. *Journal of the Korean Institute of Industrial Engineers, 41(5),* 1-8.

[10] A. Reyna, C. Martin, J. Chen, E. Soler & M. Diaz. (2018). On Blockchain and Its Integration with IoT. Challenges and Opportunities, *Future Generation Computer Systems*, *88,* 173-190.

[11] I. Makhdoom, M. Abolhasan, H. Abbas & W. Ni. (2019). Blockchain's Adoption in IoT: the Challenges, and a way Forward. *Journal of Network and Computer Applications*, *125,* 251-279.

[12] J. H. Lee & H. K. Nam. (2017), A Implementation of Blockchain based Manufacturing Supply Chain Tracking System. *Journal of Korea Safety Management Science, 19(4),* 183-188.

[13] Y. J. Lee & S. H. Lee. (2018), Efficient RBAC based on Block Chain for Entities in Smart Factory. *Journal of the Korea Convergence Society*, *9(7),* 69-75.

[14] LINDO Systems Inc. (2019). *LINGO: The Modeling Language and Optimizer.*
http://lindo.com/index.php/help/user-manuals, accessed in 2019.

주 운 기(Un Gl Joo) [정회원]

· 1992년 8월 : 한국과학기술원 산업공학과 (공학박사)
· 1997년 2월 : 한국전자통신연구원 선임연구원
· 2012년 8월 : 미국 오레곤주립대학교 Courtesy Faculty
· 현재 : 선문대학교 산업경영공학과 교수
· 관심분야 : 일정계획, 생산정보 시스템, 통신 시스템
· E-Mail: ugjoo@sunmoon.ac.kr