Journal of The Korea Society of Computer and Information
Vol. 22 No. 10, pp. 101-107, October 2017
www.ksci.re.kr
https://doi.org/10.9708/jksci.2017.22.10.101

Parallel Machine Scheduling Considering the Moving Time of Multiple Servers

Kyun-Rak Chong*

Abstract

In this paper, we study the problem of parallel machine scheduling considering the moving time of multiple servers. The parallel machine scheduling is to assign jobs to parallel machines so that the total completion time(makespan) is minimized. Each job has a setup phase, a processing phase and a removal phase. A processing phase is performed by a parallel machine alone while a setup phase and a removal phase are performed by both a server and a parallel machine simultaneously. A server is needed to move to a parallel machine for a setup phase and a removal phase. But previous researches have been done under the assumption that the server moving time is zero. In this study we have proposed an efficient algorithm for the problem of parallel machine scheduling considering multiple server moving time. We also have investigated experimentally how the number of servers and the server moving time affect the total completion time.

▸Keyword: parallel machine scheduling, total completion time, multiple server, server moving time

I. Introduction

병렬 머신 스케줄링은 작업(job)들의 총 완료 시간(total completion time, makespan)이 최소가 되게 작업들을 병렬 머신에 할당하는 문제인데, 다양한 제조 시스템 분야에서 사용되고 있다[1]. 병렬 머신 스케줄링 문제 오래전부터 활발하게 연구되었는데 병렬 머신의 특성과 제약 조건, 서버의 존재 유무, 작업의 특성, 목적 함수 등에 따라 다양한 문제들이 제시되었다. 병렬 머신은 특성에 따라 동일(identical), 균등(uniform), 무관(unrelated) 머신으로 분류된다[2]. 동일 머신은 모든 머신의 속도가 같고, 각 머신은 모든 작업을 같은 속도로 처리한다. 균등 머신은 각 머신의 속도는 다를 수 있으나 하나의 머신은 모든 작업을 같은 속도로 처리한다. 무관 머신은 각 머신의 속도가 같지 않고, 작업에 따라 머신의 처리 속도가 다르다. 또 서버의 존재 유무에 따라 서버를 사용하지 않는 경우, 단일 서버를 사용하는 경우와 다중 서버를 사용하는 경우가 있다. 작업의 특성으로는 처리(processing) 과정만 있는 경우, 준비(setup) 과정 또는 제거(removal) 과정이 있는 경우가 있고, 작업들이 독립적으로 처리될 수 있는 지 아니면 작업 간에 선행 제약 조건이 있는 지, 작업이 시작되면 중지 했다가 나중에 다시 처리할 수 있는 지 등이 있다. 목적 함수로는 총 완료 시간이 주로 사용되었는데 이 외에도, 총 지연(tardiness) 시간, 작업의 조기 완료 (early completion) 시간 등이 사용되었다.

병렬 머신 스케줄링을 응용하는 예로 강철 코일의 질을 개선하기 위한 가열냉각 공정을 보면[3, 4], 고정된 받침대에 강철 코일들이 놓여 있는데 먼저 이동식 기중기가 강철 코일을 사용가능한 용광로에 로드한다. 그러면 바로 용광로 안에서 가열이 시작되고 가열이 끝나면 이동식 기중기가 강철 코일을 용광로에서 언로드 한다. 이 예에서 서버는 이동식 기중기, 병렬 머신은 용광로, 강철 코일들은 작업이 된다. 그러면 셋업은 이동식 기중기가 코일을 용광로에 로딩하는 과정, 처리는 용광로가 가열하는 과정, 제거는 이동식 기중기가 코일을 용광로에서 언로딩하는 과정이 된다.

그 동안 단일 서버와 다중 서버를 사용하는 많은 병렬 머신 스케줄링 연구들이 수행되었는데 기존의 연구들은 모두 서버의 이동 시간을 고려하지 않고 있다. 그러나 앞의 예에서 보면 이

• First Author: Kyun-Rak Chong, Corresponding Author: Kyun-Rak Chong
*Kyun-Rak Chong (chong@cs.hongik.ac.kr), Dept. of Computer Engineering, Hongik University
• Received: 2017. 08. 01, Revised: 2017. 09. 05, Accepted: 2017. 10. 11.
• This work was supported by 2015 Hongik University Research Fund.

동식 기중기가 시작 위치에서 용광로로 이동할 때와 반대로 용광로에서 시작 위치로 이동할 때는 이동 시간이 필요하다.

본 연구에서는 다중 서버와 동일 병렬 머신들을 사용하고, 서버의 이동 시간과 작업의 준비 시간, 처리 시간과 제거 시간을 모두 고려하는 병렬 머신 스케줄링 문제를 위한 효율적인 알고리즘을 제안하였다. 또 서버 부하를 고려하여 다양한 데이터를 생성하고 서버의 수와 서버 이동 시간에 따라 작업들이 완료되는 최종 시간이 어떻게 변하는지 실험을 통해 분석하였다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 살펴보고, 3장에서는 서버 이동 시간을 고려하는 병렬 머신 스케줄링 문제를 정의하고 제안된 알고리즘에 대해 기술한다. 4장에서는 실험 결과에 대해 분석하고, 5장에서 결론을 맺는다.

II. Related Works

병렬 머신 스케줄링 문제는 오래전부터 목적 함수, 서버의 존재 유무, 병렬 머신과 작업의 특성 등에 따라 다양한 문제들이 연구되었다.

단일 서버를 사용하는 병렬 머신 스케줄링 문제는 대부분 이론적인 연구에 초점이 맞춰져 있어서, 고정된 수의 병렬 머신을 사용하거나 작업 시간도 특수한 값을 사용하는 연구가 주를 이루고 있다.

먼저 두 대의 병렬 머신을 사용하는 대표적인 연구를 보면, [5]에서는 각 작업이 준비 시간과 처리 시간만을 가질 때 총 완료 시간을 최소로 하는 스케줄링 문제가 강한 NP-하드임을 보이고 정수프로그래밍을 사용하여 이 문제를 해결하는 방법을 제시하였다. 또 특수한 경우로 모든 작업에 대해 처리 시간이 준비 시간보다 작거나 같을 때와 준비 시간과 처리 시간의 합이 모두 같을 경우를 위한 다항 시간 최적 알고리즘을 제안하였다. [6]에서는 처리 시간이 모두 같고 준비 시간이 모두 처리 시간 보다 작거나 같은 경우와 준비 시간이 모두 같고 각 작업의 처리 시간이 다른 작업의 준비 시간과 처리 시간의 합보다 작거나 같은 경우도 NP-완비임을 보이고 휴리스틱들을 제안하였다. [7]에서는 [5]의 연구를 확장하여 준비 시간과 처리 시간이 일반적인 경우에 대해 그리디 알고리즘, 유전자 알고리즘 등을 제안하고 실험적으로 분석하였다. [8]에서는 혼합 정수 프로그래밍을 사용하여 문제를 해결하는 방법을 제시하였고, 근사해를 구하기 위해 분기와 평가(branch and price) 방법과 그리디 방법을 사용하는 휴리스틱을 제안하고 실험적으로 비교하였다. 특수한 작업 시간을 사용하는 연구로는 [9]에서 작업의 준비 시간이 모두 1일 때, 스케줄링 문제가 NP-하드임을 보였고 의사 다항(pseudo polynomial) 시간 알고리즘을 제안하였다. [10]에서는 준비 시간이 상수인 경우에는 스케줄링 문제가 NP-하드임을 보였고, 처리 시간이 모두 1일 때는 다항 시간 알고리즘을 제안하였다. 처리 시간이 상수일 때는 [11]에서 스케줄링 문제가 NP-하드임을 보였다. [12]에서는 준비 시간과 제거 시간이 모두 1이고 각 작업의 처리가 끝나면 바로 제거되는 경우를 위한 휴리스틱을 제안하였다.

단일 서버를 사용하면서 여러 대의 병렬 머신을 사용하는 스케줄링 문제에 대한 연구를 보면 [3]에서는 작업의 준비 시간, 처리 시간, 제거 시간을 모두 고려한 스케줄링 문제에서 LPT 휴리스틱을 제안하였다. [13]에서는 [3]에서와 같은 조건에서 각 작업들을 병렬 머신 수만큼의 부분집합으로 분할하고 각 분분집합마다 한 대의 병렬 머신을 배정하는 다항 시간 근사 알고리즘을 제안하였다.

다중 서버를 사용하는 병렬 머신 스케줄링 문제는 단일 서버를 사용하는 경우에 비해 상대적으로 적은 수의 연구만 진행되었다. [14]에서는 다수의 언로딩 서버를 사용하고 각 작업은 처리 시간과 제거 시간만을 고려하는데 제거 시간은 모든 작업이 같다고 가정하였다. 최소 처리 시간을 갖는 작업부터 스케줄하면 최악의 경우 성능 비율이 2임을 보였고, 이 문제를 해결하기 위한 휴리스틱 알고리즘들을 제안하였다. [15]에서는 각 작업의 준비 시간과 처리 시간을 고려하였는데 고정된 수의 병렬 머신과 서버가 주어졌을 때 총 완료 시간을 최소로 하는 스케줄링 문제가 NP-하드임을 보였다. 특수한 경우로 모든 작업의 처리 시간과 준비 시간이 같을 때 문제 해결을 위한 다항 시간 알고리즘을 제안하였고, 모든 작업의 준비 시간이 1이고 서버의 수가 병렬 머신의 수보다 한 대 적을 때는 의사 다항 시간에 해결됨을 보였다. [4]에서는 준비 시간, 처리 시간, 제거 시간을 모두 고려한 스케줄링 문제를 위한 효율적인 휴리스틱을 제안하고 서버의 수와 병렬머신의 수를 증가시키면서 총 완료 시간이 단축되는 비율을 서버 부하(load)가 다른 다양한 데이터를 사용하여 분석하였다.

최근에는 작업이나 병렬 머신에 제약 조건이 있는 연구들이 많이 진행되고 있다.[16]에서는 균등 병렬 머신에서 각 작업들 사이에 선행 제약 조건이 있을 때 총 완료 시간을 최소로 하는 연구가 진행되었고, [17]에서는 서로 무관한 병렬 머신을 사용하고 각 작업들 사이에 선행 제약 조건이 있을 때 총 완료 시간을 최소로 하는 연구가 진행되었다.

비가동(unavailability) 제약 조건을 가는 병렬 머신 스케줄링 문제로는 [18]에서 두 대의 병렬 머신에서 각 작업들 사이에 선행 제약 조건이 있을 때 총 완료 시간을 최소로 하는 병렬 머신 스케줄링이 연구되었고, [19]에서 세 대의 병렬 머신에서 혼합 0/1 프로그래밍이 사용하여 총 완료 시간을 최소로 하는 병렬 머신 스케줄링이 연구되었다. [20]에서 도구 교환이나 주기적 유지보수 같은 비가동 제약 조건을 고려한 두 대의 병렬 머신에서 총 완료 시간을 최소로 하는 병렬 머신 스케줄링이 연구되었다. 문제의 크기가 작은 경우에는 혼합 0-1 프로그래밍이 사용되었고, 문제의 크기가 큰 경우에는 LPT 방법을 비롯한 여러 가지 휴리스틱을 개발하고 실험을 통해 비교분석하였다.

무관한 병렬 머신을 사용하는 연구로는 총 완료 시간을 최소로 하는 스케줄링 문제가 [21]에서 연구되었고, 총 지연 시간을 최소로 하는 문제를 해결하기 위해 타부 탐색이 [1]에서

제안되었고, [22]에서는 총 가중 지연 시간을 최소로 하는 문제가 연구되었다. 또 [23]에서 총 완료 시간과 조기 완료와 지연 페널티를 동시에 고려하는 방법이 연구되었다.

III. Proposed Method

1. Problem Definition

병렬 머신 스케줄링 문제는 다음과 같이 정의 된다. n 개의 작업이 있을 때 각 작업은 준비 과정, 처리 과정과 제거 과정을 거치게 된다. 준비 과정과 제거 과정은 서버와 해당 병렬 머신이 동시에 사용되며, 처리 과정은 해당 병렬 머신만 사용된다. 모든 서버와 작업들은 초기에 시작 위치에 위치하여 있다. 어떤 작업을 처리하기 위해서는 서버는 시작 위치에서 그 작업을 할당된 병렬 머신으로 이동시킨다. 그리고 병렬 머신과 함께 준비 과정을 수행한다. 준비 과정이 완료되면 해당 병렬 머신에서 작업의 처리가 바로 시작되고, 서버는 다른 일을 할 수 있다. 작업의 처리가 끝나면 제거 과정을 시행하게 되는데 제거 과정은 바로 시행할 필요는 없고 나중에 서버가 시간이 있을 때 시행하면 된다. 이 문제의 목적은 주어진 조건을 모두 만족하면서 모든 작업들의 총 완료 시간을 최소로 하는 스케줄을 찾는 것이다.

본 연구에서는 다음을 가정한다.

1. 서버와 병렬 머신의 개수, 작업의 개수와 작업 시간들은 처음부터 고정되어 있고 중간에 변하지 않는다.

2. 작업은 한 개의 병렬 머신에만 할당될 수 있으며, 서버와 병렬 머신은 동시에 두 개 이상의 작업을 처리할 수 없다.

3. 작업의 준비, 처리, 제거 과정들은 모두 한번 시작되면 완료될 때까지 실행되어야 하며 중간에 중지시켰다가 다시 시작할 수 없다.

4. 서버 S_i가 작업을 처리하기 위해 시작 위치에서 병렬 머신 M_k로 이동하는 시간이 m_{ik}이면, 서버가 병렬 머신 M_k에서 시작 위치로 돌아오는 시간도 m_{ik}라고 가정한다.

5. 병렬 머신 M_k에서 준비 과정을 끝낸 서버 S_i는 다른 병렬 머신 M_l로 이동해서 이 병렬 머신에서 이미 처리된 작업의 제거 과정을 수행할 수 있다. 병렬 머신 M_k에서 병렬 머신 M_l로 이동하는 시간은 $|m_{ik} - m_{il}|$로 한다.

2. Proposed Algorithm

먼저 n 개의 작업 $J_1, J_2, ..., J_n$이 있을 때 작업 J_j의 준비 시간을 s_j, 처리 시간을 p_j, 제거 시간을 r_j라 한다.

다음에는 알고리즘에 사용되는 자료 구조로 세 개의 큐와 용어를 정의하기로 한다. 각 서버들은 사용 가능 시간이 빠른 순서로 우선 순위 큐(priority queue)에 저장되는 데 이 큐를 SQ라 한다. 또 각 병렬 머신들도 사용 가능 시간이 빠른 순서로 우선 순위 큐에 저장되는 데 이 큐를 MSQ라 한다. 초기에 모든 병렬 머신은 MSQ에 저장되어 있다. MSQ에서 삭제된 병렬 머신은 할당된 작업의 준비 과정을 시행한 후 처리 과정이 완료되면 제거 과정을 위해 MRQ에 저장된다. MRQ에는 제거 과정이 필요한 병렬 머신들만 처리 완료 시간이 빠른 순서로 저장되어 있다.

정의 1 : etS(i), etM(k), et(i, k)

서버 i를 사용할 수 있는 가장 빠른 시간을 etS(i)라 하고, 병렬 머신 k를 사용할 수 있는 가장 빠른 시간을 etM(k)라 한다. et(i, k)는 서버 i와 병렬 머신 k를 동시에 사용할 수 있는 가장 빠른 시간으로 정의 한다.

알고리즘이 실행되는 과정을 보면 각 작업들은 처리 시간이 긴 것부터 병렬 머신에 할당된다. 현재 할당될 작업을 j라 하자. 또 SQ에서 사용 가능 시간이 가장 빠른 서버를 S_i라 하고, MSQ에서 사용 가능 시간이 가장 빠른 병렬 머신을 M_k라 하자. 서버 S_i와 병렬 머신 M_k가 작업 j의 준비 과정을 시작할 수 있는 가장 빠른 시간은 etS(i)에다 서버 S_i가 병렬 머신 M_k로 이동하는 시간 m_{ik}를 더한 시간과 etM(k)를 비교해서 이중 큰 쪽이 된다. 즉 et(i, k)는 식 (1)과 같이 주어진다.

$$et(i, k) = \max(etS(i)+ m_{ik}, etM(k)) \quad \cdots\cdots\cdots\cdots\cdots\cdots (1)$$

서버 S_i와 병렬 머신 M_k는 작업 j의 준비 과정을 구간 $[et(i, k), et(i, k)+ s_j]$에서 수행하고, 병렬 머신은 준비 과정이 끝나면 바로 구간 $[et(i, k)+ s_j, et(i, k)+ s_j+ p_j]$에서 작업 j의 처리 과정을 실행한 후, 작업 j의 제거를 위해 MRQ에 삽입된다.

이 때 SQ에서 사용 가능 시간이 가장 빠른 서버를 q라 하고, MSQ에서 사용 가능 시간이 가장 빠른 병렬 머신을 x라 하자. 또 MRQ에서 작업의 제거를 가장 빨리할 수 있는 병렬 머신을 y라 하면, 서버는 i와 q, 병렬 머신은 x와 y가 있으므로 서버와 병렬 머신의 조합은 다음 네 가지 경우가 생기게 된다. 제안 알고리즘은 이 네 가지 경우 중 실행 가능 시간이 가장 빠른 경우를 선택해서 실행한다.

경우 1: 서버 i와 병렬 머신 x가 공동으로 일하는 경우

서버 i와 병렬 머신 x가 작업 (j+ 1)의 준비 과정을 수행하는 경우이다. 서버 i는 작업 (j+ 1)의 준비 과정을 수행하기 위해 시작 위치로 돌아와서 작업을 병렬 머신 x로 이동해야 하므로 사용 가능 시간은 $et(i, k)+ s_j+ m_{ik}+ m_{ix}$가 된다. 그러므로 공동으로 작업을 수행할 수 있는 가장 빠른 시간 et(i, x)는 식 (2)와 같이 주어진다.

$$et(i, x) = \max(et(i, k)+ s_j+ m_{ik}+ m_{ix}, etM(x)) \quad \cdots\cdots (2)$$

경우 2: 서버 i와 병렬 머신 y가 공동으로 일하는 경우

서버 i와 병렬 머신 y가 y에 이미 할당되어 있던 작업의 제거 과정을 수행하는 경우이다. 서버 i는 병렬 머신 k에서 병렬 머신 x로 이동해야 하므로 사용 가능 시간은 $et(i, k)+ s_j+ |m_{ik}-m_{ix}|$가 된다. 그러므로 공동으로 작업을 수행할 수 있는 가장 빠른 시간 et(i, y)는 식 (3)과 같이 주어진다..

$$et(i, y) = \max(et(i, k)+ s_j+ |m_{ik}-m_{ix}|, etM(y)) \quad \cdots\cdots (3)$$

경우 3: 서버 q와 병렬 머신 x가 공동으로 일하는 경우

서버 q와 병렬 머신 x가 작업 (j+1)의 준비 과정을 수행하는 경우이다. 서버 q는 작업 (j+1)의 준비 과정을 수행하기 위해 시작 위치에서 작업을 병렬 머신 x로 이동해야 하므로 사용 가능 시간은 $etS(q)+m_{qx}$가 된다. 공동으로 작업을 수행할 수 있는 가장 빠른 시간 et(q, x)는 식 (4)와 같이 주어진다.

$$et(q, x) = \max(etS(q)+m_{qx}, etM(x)) \cdots\cdots\cdots\cdots (4)$$

경우 4: 서버 q와 병렬 머신 y가 공동으로 일하는 경우

서버 q와 병렬 머신 y가 y에 이미 할당되어 있던 작업의 제거 과정을 수행하는 경우이다. 서버 q는 시작 위치에서 병렬 머신 x로 이동해야 하므로 사용 가능 시간은 $etS(q)+m_{qy}$가 된다. 그러므로 공동으로 작업을 수행할 수 있는 가장 빠른 시간 et(q, y)는 식 (5)와 같이 주어진다.

$$et(q, y) = \max(etS(q)+m_{qy}, etM(y)) \cdots\cdots\cdots\cdots (5)$$

알고리즘 기술에 필요한 함수들 중에서 우선 순위 큐 SQ, MSQ, MRQ와 관련된 함수를 보면 insertSQ()는 SQ에 서버를 삽입하는 함수이고 deleteSQ()는 SQ에서 사용 가능 시간이 가장 빠른 서버를 삭제하는 함수이다. insertMSQ()은 MSQ에 병렬 머신을 삽입하는 함수이고 deleteMSQ()는 MSQ에서 사용 가능 시간이 가장 빠른 병렬 머신을 삭제하는 함수이다. insertMRQ()은 MRQ에 병렬 머신을 삽입하는 함수이고 deleteMRQ()는 MRQ에서 할당된 작업의 처리 완료 시간이 가장 빠른 병렬 머신을 삭제하는 함수이다.

작업의 준비 과정을 수행하는 job_setup(j, S_i, et)와 제거 과정을 수행하는 job_remove(S_i, et)가 그림 1에 나타나 있다. 여기서 job(M_k)는 현재 병렬 머신 Mk에 할당되어 있는 작업을 가리킨다. job_setup(j, S_i, et) 함수는 먼저 MSQ에서 병렬 머신을 삭제한다. 삭제된 병렬 머신을 M_k라 하면 서버 S_i와 함께 작업 j의 준비 과정을 시각 et부터 $et+s_j$까지 수행한다. 서버 S_i의 다음 사용 가능 시간은 $et+s_j+m_{ik}$로 변경된다. 여기서 m_{ik}는 서버 S_i가 시작 위치로 돌아가는데 필요한 이동 시간이다. 준비 과정을 마친 병렬 머신은 바로 시각 $et+s_j$부터 $et+s_j+p_j$까지 작업 j의 처리 과정을 실행한다. 그러므로 병렬 머신 M_k의 다음 사용 가능 시간은 $et+s_j+p_j$로 변경된다. 그 다음 병렬 머신 M_k는 작업 j의 제거를 위해 MRQ에 삽입된다. job_remove(S_i, et) 함수는 먼저 MRQ에서 병렬 머신을 삭제한다. 삭제된 병렬 머신을 My라 하면 서버 S_i와 함께 M_y에 할당되어 있던 작업 j의 제거 과정을 시각 et부터 수행한다. 서버 S_i의 다음 사용 가능 시간은 $et+r_j+m_{iy}$로 업데이트 되고 S_i는 SQ에 삽입된다. 여기서 m_{iy}는 작업 j의 제거 과정을 마친 서버가 시작 위치로 돌아가는데 필요한 이동 시간이다. 또 병렬 머신 My의 다음 사용 가능 시간은 $et+r_j$로 변경되고, 병렬 머신 M_y는 MSQ에 삽입된다.

앞에서 설명한 서버와 병렬 머신의 네 가지 조합 중에서 실행 가능 시간이 가장 빠른 경우를 구하는 함수가 findcase(S_i, j, et)이다. 이 함수는 준비 과정을 마친 서버가 있을 경우(S_i > 0)에는 et(i, x), et(i, y), et(q, x), et(q, y) 중에서 그 값이 최소인 경우(각각 1, 2, 3, 4)를 리턴하게 된다. 준비 과정을 마친 서버가 없을 경우(S_i = 0)에는 et(q, x)와 et(q, y) 중

```
void job_setup(j, Si, et);
{
    Mk = deleteMSQ();
    etS(i) = et + si + mik;
    job(Mk) = j;
    etM(k) = et + sj + pj;
    insertMRQ(Mk);
    et = mik;
}

void job_remove(Si, et)
{
    My = deleteMRQ();
    j = job(My);
    etM(y) = et + rj;
    job(My) = 0;
    insertMSQ(My);
    etS(i) = et + rj+ miy;
    insertSQ(Si);
    Si= 0;
}
```

Fig. 1. Job setup and remove procedures

```
Parallel Machine Scheduling Algorithm
{
    initialize SQ MSQ, MRQ;
    for (each server i) insertSQ(Si);
    for (each machine k) insertMSQ(Mk);

    Si = 0;
    j = 0;
    done = false;
    while (not done) {
        switch (findcase(Si, j, et)) {
            case 1 : // et(i,x) is min
                j = j + 1;
                jop_setup(j, Si, et);
                break;
            case 2 :  // et(i,y) is min
                job_remove(Si, et);
                break;
            case 3 :  // et(q,x) is min
                j = j + 1;
                Si = deleteSQ();
                job_setup(j, Si, et);
                break;
            case 4 :  // et(q,y) is min
                Si = deleteSQ();
                job_remove(Si, et);
                break;
            case 5 : // algorithm termination
                done = true;
        }
    }
    retrurn max∀i∀k(etS(i), etM(k));
```

Fig. 2. Proposed parallel machine scheduling algorithm

et(q, x)가 최소인 경우에는 3, 아닌 경우에는 4를 리턴한다. 그리고 최소값은 et에 저장한다. 또 모든 작업들의 제거 과정이 완료된 경우에는 알고리즘을 종료하기 위해 5를 리턴한다.

모든 작업이 병렬 머신에 할당되면 각 병렬 머신에 할당된 작업의 처리가 끝나는 데로 제거하게 되고 모든 작업들이 제거되면 종료한다. 제안된 알고리즘이 그림 2에 나타나 있다.

우선순위 큐를 사용할 때 발생할 수 있는 에이징 문제를 보면, 서버 큐에서 사용 가능 시간이 가장 빠른 서버가 삭제되어 어떤 작업의 준비(제거) 과정을 수행하면 이 서버의 다음 사용 가능 시간은 처리된 작업의 준비(제거) 시간과 서버의 이동 시간만큼 증가된 후 다시 서버 큐에 삽입된다. 즉, 서버는 사용될 때마다 다음 사용 가능 시간이 상당히 증가하게 된다. 또 병렬 머신도 사용될 때마다 작업의 준비 시간과 처리 시간 또는 제거 시간만큼 다음 사용 가능 시간이 증가하게 된다. 그러므로 이전에 큐에 삽입된 서버나 병렬 머신들부터 사용 가능 시간이 빠른 순서대로 상당히 균일하게 사용되기 때문에 에이징 문제는 발생하지 않는다.

IV. Performance Evaluation

제안된 알고리즘은 C 언어를 사용해서 PC에서 구현되었고, 서버의 수와 서버의 이동 시간이 총 완료 시간에 어떤 영향을 미치는 지 실험을 통해 분석하였다. 실험을 위해 데이터는 다음과 같이 생성되었다. 병렬 머신의 수는 30을 사용하였고, 서버의 수(S)는 1, 2, 3, 4, 5 를 사용하였다. 서버 i가 시작 위치에서 병렬 머신 k로 이동하는 시간 m_{ik} = smtc*k로 가정하였다. 계수 smtc는 0, 5, 10, 20을 사용하였는데 smtc = 0이면 서버 이동 시간은 없음을 의미한다. 또 각 서버가 시작 위치에서 같은 병렬 머신으로 이동하는 시간은 동일하다고 가정하였다. 작업의 수는 1,000개를 사용하였고, 작업의 준비 시간과 처리 시간은 [24]에서와 같이 서버 부하(load)를 사용하여 임의로 생성되었다. 서버 부하는 준비 시간의 합을 처리 시간의 합으로 나눈 값으로 정의되는데, 서버 부하가 L일 때 처리 시간이 구간 [0, P]에서 임의로 생성되면 준비 시간은 구간 [0, L*P]에서 생성된다. 제거 시간은 준비 시간과 같은 구간에서 임의로 생성되었다. 서버 부하는 L=0.05, 0.5 두 가지 경우를 고려하였는데, 각 경우에 대해 10개씩 임의로 데이터를 생성하여 평균 결과를 측정하였다.

4.1 L = 0.5인 경우

서버 부하가 0.5인 경우는 작업의 처리 시간의 평균이 t라고 하면 준비 시간과 제거 시간의 평균이 각각 0.5t가 된다. 즉, 평균 준비 시간과 평균 제거 시간의 합이 평균 처리 시간과 같게 되는 경우로 서버 부하가 높은 경우이다. 각 작업의 준비 과정 과 제거 과정은 서버가 사용되므로 서버가 가동되는 시간이 많아지게 되고, 병렬 머신들은 서버가 작업을 준비해주거나 작업이 끝나면 제거해 줄 때까지 가동하지 않은 상태로 기다리게 된다.

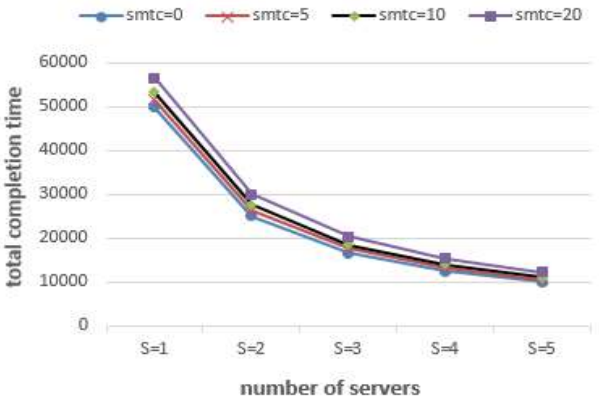

Fig. 3. The comparison of the total completion times for L=0.5

그림 3에 서버 부하가 0.5일 때 서버의 수와 서버 이동 시간에 대한 총 완료 시간의 변화가 그래프로 그려져 있다. 실험 결과를 보면 smtc가 0인 경우는 서버의 수가 1일 때의 총 완료 시간에 비해 서버의 수가 2, 3, 4, 5로 증가하면 총 완료시간이 각각 50%, 33.4%, 25%, 20%로 이상적으로 감소하고 있다. 즉 서버의 수가 S가 되면 총 완료시간이 1/S이 되는 것을 알 수 있다. smtc가 커질수록 서버의 이동 시간이 길어져 서버의 수가 S가 되면 총 완료시간이 1/S보다 약간 커짐을 나타내고 있다. 실험 결과에 의하면 smtc가 20일 때 서버의 수가 2, 3, 4, 5로 증가하면 서버의 수가 1인 경우에 비해 총 완료 시간이 각각 53.1%, 35.9%, 27.1%, 21.8%로 감소하고 있다.

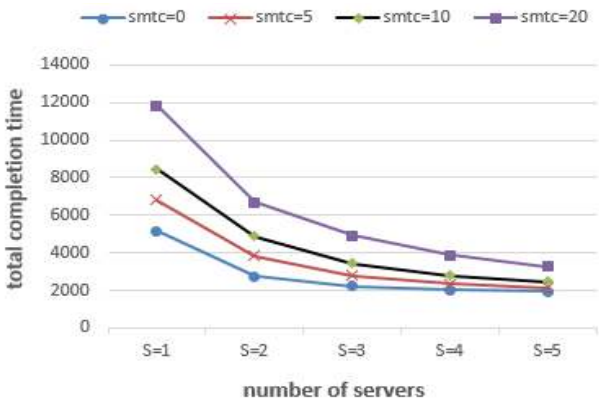

Fig. 4. The comparison of the total completion times for L=0.05

4.2 L = 0.05인 경우

서버 부하가 0.05인 경우는 작업의 준비와 제거의 평균 시간이 평균 처리 시간의 20분의 1이 되는 경우로 서버 부하가 낮은 경우이다. 서버와 병렬 머신이 평균 시간이 0.05t인 작업의 준비 과정을 끝내면 처리 과정은 병렬 머신이 단독으로 수행하고, 그 사이 서버는 다른 작업들의 준비 과정이나 제거 과정을 수행할 수 있다. 서버 부하가 낮기 때문에 서버의 수를 증가시켜도 총 완료 시간이

감소하는 비율이 L=0.5인 경우에 비해 작게 된다.

그림 4에 서버 부하가 0.05일 때 서버의 수와 서버 이동 시간에 대한 총 완료 시간의 변화가 그래프로 그려져 있다. 실험 결과를 보면 smtc가 0인 경우는 서버의 수가 1일 때의 총 완료 시간에 비해 서버의 수가 2, 3, 4, 5로 증가하면 총 완료 시간이 각각 53.5%, 43.1%, 39.5%, 37.8%가 된다. L=0.5인 경우와 비교하면 각각 3.5%, 9.7%, 14.5%, 17.8% 적게 감소하고 있음을 알 수 있다. 특히 서버의 수를 4에서 5로 증가 시킨 경우에는 총 완료 시간이 L=0.5인 경우에는 25%에서 20%로 5% 감소했는데 L=0.05인 경우에는 39.5%에서 37.8%로 1.7% 밖에 감소하지 않았다. smtc가 커질수록 서버의 수가 증가하면 총 완료 시간이 서버의 수가 1인 경우에 비해 상대적으로 더 많이 감소하였다. 실험 결과를 보면 smtc가 20일 때 서버의 수가 2, 3, 4, 5로 증가하면 서버의 수가 1인 경우에 비해 총 완료 시간이 각각 56.6%, 41.5%, 32.7%, 27.5%로 감소하고 있다. 또 L=0.5인 경우와 비교하면 각각 3.5%, 5.6%, 5.6%, 5.7% 적게 감소하고 있다.

V. Conclusion

병렬 머신 스케줄링은 서버와 병렬 머신들이 있을 때 총 완료 시간이 최소가 되게 작업들을 병렬 머신에 할당하는 문제로 다양한 응용 분야에서 사용되고 있다. 과거의 연구들은 단일 서버를 사용하는 연구가 주류를 이루었고, 다수의 서버를 사용하는 경우에도 서버의 이동 시간은 고려되지 않았다. 본 연구에서는 다중 서버를 사용하고 서버의 이동 시간을 고려하는 병렬 머신 스케줄링을 위한 효율적인 알고리즘을 제안하였다. 또 서버 부하를 고려하여 다양한 데이터를 생성하여 서버의 수와 서버 이동 시간에 따라 작업들이 완료되는 최종 시간이 어떻게 변하는지 실험을 통해 분석하였다.

본 연구에서 얻은 실험 결과는 서버 이동 시간과 작업들의 총 완료 시간을 고려할 때 몇 대의 서버들을 사용하는 것이 합리적인 가를 결정하는데 활용될 것으로 기대된다.

향후 연구 과제로는 각 서버의 이동 시간과 병렬 머신의 성능이 동일하지 않을 경우 총 완료 시간을 최소로 하는 효율적인 알고리즘의 개발이 필요한 것으로 사료된다.

REFERENCES

[1] J. H. Lee, J. M. Yu and D. H. Lee, “A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: minimizing total tardiness, ” Intl. J. of Adv. Manuf. Tech., 2013.

[2] K. Lee, J. Y.-T. Leung and M. L. Pinedo, “ Makespan minimization in online scheduling with machine eligibility,” Ann. Ope. res., vol. 204, pp. 189-222, 2013.

[3] X. Xie, H. Zhou, Y. Li, and Y. Zheng, “Scheduling Parallel Machines with a Single Server,”IEEE Intl. Conf. on MIC, pp. 453-456, 2012.

[4] K. Chong, “An efficient algorithm for scheduling parallel machines with multiple servers,” Journal of the Korea Society of Computer Information, vol. 19, no. 6, pp. 101-108, 2014.

[5] A. H. Abdekhodaee and A. Wirth, “Scheduling parallel machines with a single server: some solvable cases and heuristics,” Computers and Operation Research 29, pp. 295-315, 2002.

[6] A. H. Abdekhodaee, A. Wirth and H. S. Gan, “Equal processing and equal setup time cases of scheduling parallel machines with a single server,” Computers and Operation Research 31, pp. 1867-1889, 2004.

[7] A. H. Abdekhodaee, A. Wirth and H. S. Gan, “Scheduling parallel machines with a single server: the general case,” Computers and Operation Research 33, pp. 994-1009, 2006.

[8] H. S. Gan, A. Wirth and A. H. Abdekhodaee, “A branch-and-price algorithm for scheduling parallel machines with a single server,” Computers and Operation Research 39, pp. 2242-2247, 2012.

[9] F. Werner and S. A. Kravchenko, “Parallel Machine Scheduling with a Single Server,” Mathematics and Computer Modelling, vol. 26, pp. 1-11, 1997.

[10] N. G. Hall, C. N. Potts and C. Sriskandarajah, “Parallel machine scheduling with a common server,”Discrete Applied Mathematics, vol. 102, pp. 223-243, 2000.

[11] P. Brucker, C. Dhaenens-Flipo, S. Knust, S. A. Kravchchenko, and F. Werner, “Complexity results for parallel machine problems with a single server,” J. of Scheduling, vol. 5, pp. 429-457, 2002.

[12] J. Hu, Q. Jhang, J. Dong, and Y. Jiang, “Parallel Machine Scheduling with a Single Server: Loading and Unloading,” LNCS 8287, pp. 106-116, 2013.

[13] X. Xie, Y. Li, and Y. Zheng, “Scheduling Parallel Machines with a Single Server: a Dedicated Case,” Fifth Intl. Joint Conf. on Computational Science and Optimization, pp. 146-149, 2012.

[14] J. Ou, X. Qi, and C. Y. Lee, “Parallel Machine Scheduling with Multiple Unloading Servers,” J. of Scheduling, vol. 13, no. 3 pp. 213-226, 2009.

[15] F. Werner and S. A. Kravchenko, “Scheduling with Multiple Servers,” Automation and Remote Control, vol. 71, no. 10, pp. 2109-2121, 2010.

[16] C. Wu, L. Wang and X. Zheng, “An effective estimation of distribution algorithm for solving uniform parallel machine scheduling problem with precedence

constraints," 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 2626 - 2632, 2016.
[17] M.-A. Hassan, I. Kacem, S. Martin and I. M. Osman, "Valid inequalities for unrelated parallel machines scheduling with precedence constraints," International Conference on Control, Decision and Information Technologies (CoDIT) pp. 677-682, 2016.
[18] K. Abdellafou and Q. Korbaa, "Makespan minimization for two parallel machines with unavailability constraints," IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 601-606, 2016.
[19] Z. Xu, A. Liu and Q. Wang, "Mixed 0-1 programming model for three parallel machines scheduling problem with machine-dependent unavailable constraints," 13th International Conference on Service Systems and Service Management (ICSSSM), pp. 1-4, 2016.
[20] J. He, Q. Li and D. Xu, "Scheduling two parallel machines with maintenance-dependent availabilities," Computer and Operation Research 72, pp. 13-42, 2016.
[21] L. Y. Wang, X. Huang, P. Ji and E. M. Feng, "Unrelated parallel machine scheduling with deteriorating maintenance activities to minimize the total completion time ," Optim. Letters, 2012.
[22] C. W. Lin, Y. K. Lin and H. T. Hsieh, "Ant colony optimization for unrelated parallel machine scheduling," Intl. J. of Adv. Manuf. Tech., 2013.
[23] V. Kayvanfar, E. Teymourian and K. Alizadeh, " Intelligent water drops algorithm on parallel machines scheduling," International Conference on Industrial Engineering and Operations Management (IEOM) pp. 1-5, 2015.
[24] C. P. Koulamas, "Scheduling two parallel semiautomatic machines to minimize machine interference," Computers and operation Research, vol. 23, no. 10, pp.945-956, 1996.

Authors

Kyun Rak Chong received the B.S. degree in computer science and statistics from Seoul national university, Korea, in 1978, the M. S. degree in computer science from Korea advanced institute and technology, Korea, in 1980, and the Ph.D. degree in computer science from the university of Minnesota, Minneapolis, in 1991. He is currently a professor in the department of computer engineering, Hongik University. In 1998 and 2006, he visited the department of computer information science and engineering, university of Florida, Gainesville. His research interests include network algorithms, wireless sensor networks, public-shared networks, and parallel machine scheduling.