Journal of the Korean Institute of Industrial Engineers
Vol. 38, No. 1, pp. 31-40, March 2012.

동적 스케줄링을 위한 분산 도착시간 제어 (Distributed Arrival Time Control) 알고리즘의 개량

고재호 · 옥창수†

홍익대학교 산업공학과

Advanced Distributed Arrival Time Control for Single Machine Problem in Dynamic Scheduling Environment

Jea-Ho Ko · Chang-Soo Ok

Department of Industrial Engineering, Hongik University, Seoul

Distributed arrival time control (DATC) is a distributed feedback control algorithm for real-time scheduling problems in dynamic operational environment. Even though DATC has provided excellent performance for dynamic scheduling problems, it can be improved by considering the following considerations. First, the original DATC heavily depends on the quality of initial solution. In this paper, well-known dispatching rules are incorporated DATC algorithm to enhance its performance. Second, DATC improves its solution with adjusting virtual arrival times of jobs to be scheduled in proportion to the gap between completion time and due date iteratively. Since this approach assigns the same weight to all gaps generated with iterations, it fails to utilize significantly more the latest information (gap) than the previous ones. To overcome this issue we consider exponential smoothing which enable to assign different weight to different gaps. Using these two consideration This paper proposes A-DATC (Advanced-DATC). We demonstrate the effectiveness of the proposed scheduling algorithm through computational results.

Keywords: Distributed Arrival Time Control(DATC), Distributed Control, Exponential Smoothing, Advanced DATC, Dynamic Scheduling

1. 서 론

실제 작업 환경에서 긴급 주문, 주문 변경, 기계 고장, 작업자 오류 등으로 의해 야기되는 불확실성은 스케줄링에 대한 의사결정에 큰 영향을 미친다(Simchi levi, 2006). 이러한 동적 환경에서 효율적인 스케줄링 알고리즘은 (1) 운영 환경변화에 적절히 대응할 수 있어야 하고, (2) 환경변화에 적절한 새로운 스케줄을 생성할 수 있도록 실시간으로 리스케줄링 할 수 있도록 설계되어야 한다(Parunak, 1991).

이러한 스케쥴링 문제를 해결하기 위한 방안으로 먼저, 분지 한계법(Bagchi, 1987), 동적 계획법(Abdul-Razaq and Potts, 1988), 라그랑지안 완화 기법(Li, 1997)과 같은 최적해법(Exact solution)을 들 수 있다. 그러나 이러한 최적화 기법들은 많은 계산량이 요구되므로 실시간으로 변화하는 동적 환경의 제조 시스템에 적용하기는 적합하지 않다. 이를 보완하기 위해 담금질 기법(Simulated Annealing)(Aarts, 1986), 타부탐색(Tabu Search)(Glover, 1986), 유전자 알고리즘(Genetic Algorithm)(Goldberg, 1989)과 같은 휴리스틱 기법을 통한 스케줄링 시스템이 개발, 활용되고 있지만 실제 작업환경에 적용하기에는 여전히 계산량이 많고 시간이 오래 걸리는 단점을 가지고 있다(Cho and Erkoc, 2009).

이러한 최적화 기법과 기존 휴리스틱 알고리즘의 단점을 극복하기 위하여 Prabhu and Deffie(1999)는 분산 제어 구조 알고리즘 중의 하나인 DATC(Distributed Arrival Time Control) 알

이 논문은 2010학년도 홍익대학교 학술연구진흥비에 의하여 지원되었음.
† 연락저자 : 옥창수, 121-791 서울시 마포구 상수동 홍익대학교 산업공학과, Tel : 02-320-3026, Fax : 02-336-1130, E-mail : okcs@hongik.ac.kr
2011년 8월 18일 접수; 2011년 12월 4일 수정본 접수; 2012년 2월 4일 게재 확정.

고리즘을 제안하였다. DATC 알고리즘은 주어진 스케줄에 대하여 각 작업의 완료시간과 납기일의 차이를 계산하고 이 차이가 클수록 가상 도착시간이 빨라지도록 조정하여 이 가상 도착시간에 따라 새로운 작업 스케줄을 생성하는 피드백 제어 구조를 가지고 있다. 또한, 각 작업에 대한 가상 에이전트를 활용하는 분산 제어구조를 가지고 있어 생산 및 제조 현장에 존재하는 불확실성에 대한 적응성과 유연성을 제공한다.

이와 같이 DATC는 동적 환경 하에서 효과적인 스케줄을 제공하는데 적합하지만 해결되어야 하는 다음과 같은 단점을 포함하고 있다. 먼저, DATC 알고리즘에 의한 결과 값은 빠르게 지역 최적해(local opitmum)에 수렴하고 이를 극복하기 위한 근본적인 해결책을 제시하지 못한다. 이 단점을 극복하기 위하여 그동안 매개 변수 조정 등을 통해 좀 더 나은 해를 구하고자 하는 여러 연구들이 진행되어 왔지만(Cho and Erkoc, 2009; Kim and Ok, 2008), 이 DATC 알고리즘의 구조적인 문제를 직접적으로 다루지는 않았다. DATC에서는 가상의 도착시간을 계산할 때 알고리즘이 수행됨에 따라 발생되는 작업 완료시간과 납기일의 편차들을 동일한 가중치로 고려함을 의미하여 점차적으로 해는 개선되지만 최신의 편차에 의한 영향력이 점차 줄어들어 지역최적해로 수렴하게 된다고 판단된다. 이는 제 3장의 실험을 통해서 증명한다. 두 번째로 알고리즘의 성능이 초기해의 품질에 큰 영향을 받는다는 단점을 가지고 있다(Cho and Erkoc, 2009). 이는 좋은 초기 해를 사용하였을 경우 빠르게 좋은 해로 수렴하는 반면, 그렇지 않은 경우에 초기해로 인한 한계를 극복하지 못하고 좋지 않은 지역최적해로 수렴하게 된다.

따라서, 본 논문에서는 앞서 언급한 DATC의 한계를 극복하기 위하여 다음 사항을 반영하여 Advanced-DATC 알고리즘을 제안한다. 먼저 DATC의 가상 도착시간 계산 시 지수평활법의 특징을 활용하여 최근 변화에 대한 가중치의 영향을 크게 함으로써 보다 좋은 해를 찾도록 유도한다. 둘째, 여러 개의 초기해를 활용하여 초기해의 품질이 미치는 알고리즘에 대한 영향을 줄여 알고리즘이 안정적으로 좋은 해를 제공하도록 설계한다. 본 논문의 구성은 다음과 같다. 제 2장에서는 본 연구에서 고려하는 DATC 알고리즘에 대해서 알아보고 제 3장에서는 다양한 실험을 통하여 DATC의 한계점에 대하여 논의한다. 그리고 제 4장에서는 제 3장에서 제기한 문제를 해결하기 위한 새로운 A-DATC를 제안한다. 제 5장에서는 실험 결과를 통해 A-DATC가 불확실성이 존재하는 제조현장에서 효과적인 스케줄링을 제공할 수 있음을 보여주고 마지막으로 제 6장에서 본 논문의 결론 및 향후 연구에 대하여 논한다.

2. Distributed Arrival Time Control(DATC)

시간에 따라 동적으로 변화하는 작업환경에 적합한 작업 스케줄을 신속하게 제공하기 위하여 Prabhu(1999)는 DATC 알고리즘을 개발, 제안하였다. 먼저, DATC는 각 작업에 대한 가상의 도착시간을 이용하여 작업의 순서를 결정한다. 이렇게 결정된 스케줄에 따라 각 작업의 완료시간을 계산하고 이를 완료예정시간(납기일)과 비교하여 그 차이에 따라 도착시간을 변경한다. 도착시간 변경 시, 작업 완료시간이 예정보다 늦은 작업의 경우 도착시간을 빨라지도록 조정하고 작업 완료시간이 예정보다 빠를 경우 도착시간이 늦어지도록 하여 새로운 스케줄 생성에 반영이 되도록 한다. DATC는 이 과정을 반복하여 해를 개선해 나가는 비교적 간단한 구조를 채용하고 있어 작업의 추가, 변경, 취소 등에 의해 신속히 리스케줄 되어야 하는 동적인 작업환경의 스케줄링에 적합하다. 또한, DATC는 도착시간 제어를 위해 하나의 작업에 하나의 가상 에이전트를 채용하는 분산 의사 결정 구조를 가지고 있어 작업 수의 증가 또는 스케줄 기간의 증가에도 적절히 대응이 가능하다.

DATC 알고리즘을 설명하기 위하여 단일 공정 스케줄링 문제를 고려한다. 스케줄되어야 하는 n개의 작업이 있고 각 작업 $i=1, \cdots, n$는 납기일, d_i를 가지고 있다. DATC 알고리즘하에서는 각 작업을 대표하는 에이전트들은 각 작업에 대한 가상 도착시간, $a_i(t)$,를 중앙 제어 시스템에 보고한다. 중앙 제어 시스템을 이 가상 도착시간을 바탕으로 작업들의 우선순위를 결정하고 이 우선순위를 해당 작업들의 스케줄로 활용한다. 그리고 이 작업 스케줄에 따른 각 작업의 완료시간, $C_i(t)$를 계산(작업 처리시간(p_i)의 합으로 계산)하고 현재 스케줄에 대한 비용을 계산한다. 이때, 모든 부품들은 Earliness와 Tardiness에 따른 비용을 가지고 이 비용은 작업완료일과 납기일의 차이에 따라 동일한 비율로 증가한다고 가정한다. 이후 중앙 제어 장치는 각 작업에 대한 완료시간을 해당 에이전트에게 알리고 해당 에이전트는 식 (1)에 따라 새로운 가상 도착시간을 계산하여 중앙 제어장치에 보고하여 새로운 스케줄을 생성한다. 이 피드백 과정은 스케줄의 품질 향상을 위해 반복된다(Prabhu, 2000).

$$a_i(t) = k_i \int_0^t (d_i - C_i(t))\, dt + a_i(0) \tag{1}$$

여기에서 k_i는 통제자 이득(cotroller gain)이라 불리는 조절모수이고 $a_i(0)$은 임의의 초기 도착시간을 나타낸다. 식 (1)에 따라 새로운 가상의 도착시간 $a_i(t)$는 초기 도착시간($a_i(0)$)과 초기 도착시간에 의해 생성된 스케줄에 의한 납기일과 작업완료시간과의 편차($d_i - C_i(t)$)를 반영하여 계산된다. 계산상의 편의를 위하여 m반복 단계의 새로운 도착시간 $a_i(m)$은 다음의 식을 통해 근사치를 구할 수 있다.

$$a_i(m) = a_i(m-1) + k_i z_i(m-1) \tag{2}$$

$a_i(m\text{-}1)$은 (m-1)단계에서의 도착시간을 나타내고 $z_i(m\text{-}1)$는 작업완료시간과 납기일의 차이(납기일 편차)를 의미한다. 그리고 k_i는 조절모수인 통제자 이득이다. i번째 작업의 m번째

단계에서의 납기일의 편차 $z_i(m)$는 다음의 식으로 표현할 수 있다.

$$z_i(m) = d_i - C_i(m) \tag{3}$$

이때 $C_i(m)$은 주어진 스케줄 하에서 작업 i의 작업 완료시간이다. 이와 같이 각 에이전트들은 피드백 정보 또는 작업 완료시간과 만기시간의 차이를 활용으로 각 부품의 도착시간을 조절하고 이를 반복하여 스케줄의 품질을 향상을 추구한다.

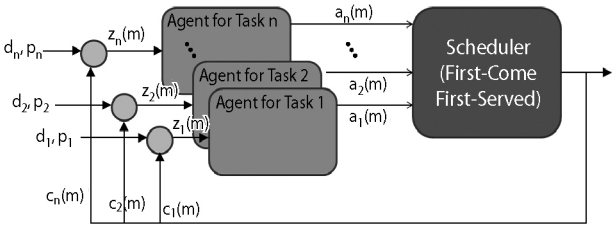

Figure 1. Feedback control structure of DATC

<Figure 1>은 DATC의 피드백 제어 구조를 보여준다. 각 작업을 대표하는 에이전트는 작업들로부터 납기일(d_i)과 작업처리시간(p_i)를 입력받아 스케줄러로부터의 피드백을 바탕으로 새로운 가상의 도착시간($a_i(m)$)을 생성, 스케줄러에 보고하고, 스케줄러는 FCFS(First-Come First-Served) 규칙에 의해 스케줄을 생성하여 각 에이전트에게 스케줄에 의한 가상 작업 완료시간을 통보한다. 이때, 각 에이전트는 작업 완료시간과 납기일의 차이를 최소화하기 위하여 해당 작업의 가상 도착시간을 조정하여 스케줄러에 보고한다. DATC는 기본적으로 이 피드백 제어를 반복적으로 수행하여 좋은 스케줄을 찾는 구조를 가지고 있다. 가상 도착시간 조정을 위한 통제자 이득 k_i는 일반적으로 아래의 범위에서 가장 좋은 성과를 나타낸다고 알려져 있다(Prabhu, 2000).

$$0 \le k_i \le 2 \tag{4}$$

본 논문에서는 DATC 알고리즘을 평가하기 위하여 단일 공정 문제에서 모든 부품들의 총 earliness와 tardiness 비용을 고려한 평균제곱편차(MSD : Mean Squared Deviations)을 고려한다(Prabhu, 2000). MSD는 다음의 식으로 표현된다.

$$MSD = \frac{\sum_i (d_i - C_i)^2}{n} \tag{5}$$

3. DATC의 한계

휴리스틱 방법은 크게 Construction Heuristic과 Improvement Heuristic으로 나눌 수 있다. Construction Heuristic은 초기해가 없는 처음의 상태에서 Local search를 통해 초기해를 구하는 방식이며 Improvement Heuristic은 초기해가 있는 상태에서 목적함수의 값을 향상시키는 방법이다(Baker, 1974; Pirlot, 1992). DATC 알고리즘은 이 두 방법의 장점을 모두 채용하여 초기해를 가상의 초기 도착시간($a_i(0)$)에 따라 생성하고, 가상의 작업 완료시간과 납기일의 차이에 따라 새로운 가상 도착시간을 계산하여 목적함수의 값 향상을 도모한다. DATC는 튜닝 파라미터 k를 통한 가상의 도착시간($a_i(m)$)의 가중치 변화, 가상 초기 도착시간($a_i(0)$)의 조정을 통하여 구하는 해를 향상시킬 수 있도록 고안되었다(Prabhu and Deffie, 1999). 그러나, 두 모수의 조정이 해의 품질에 어떤 영향을 미치는 지 또는 모수 설정을 위한 구체적인 가이드 라인에 대한 기존 연구는 없었다. 따라서, 본 장에서는 이를 확인하기 위하여 다음 두 가지 알고리즘 모수의 변화에 대한 실험을 다음과 같이 실시하였다.

- 초기해($a_i(0)$)의 변화에 따른 결과 값 비교
- 가상의 도착시간($a_i(m)$) 가중치 변화에 따른 결과 값 비교

실험의 각 작업시간은 1과 100사이의 임의의 값을 사용하였으며, 각 작업의 납기일은 기존 연구(Prabhu, 2000)에서와 같이 0과 모든 작업의 작업시간의 합 사이에 값을 랜덤으로 선정하여 활용하였다. 특히, Prabhu 논문에서 제안한 다양한 스케줄링 조건에 대한 실험을 위해 다음과 같이 납기일 생성을 위한 범위(c)를 조정한다. 먼저, 납기일이 촉박한 경우에 대하여 c를 0.5로, 납기일이 타이트한 경우에 대하여 c를 1.0으로 조정하고, 마지막으로 납기일이 여유로운 경우에 대하여 c값을 1.5로 설정한다. Job의 수는 10개 이내의 벤치마킹 문제를 사용했지만 다양한 상황에서의 DATC 성능을 확인하고자 Job의 크기 50, 100, 200에 대하여 실험을 진행하였다. 실험 데이터를 생성하는 데 사용된 조건을 정리하면 <Table 1>과 같다. 또한 앞으로의 실험들은 각각 실험 결과의 신뢰성을 높이기 위하여 모든 실험 조건에 대하여 100번의 실험이 반복하여 그 평균값을 사용하였다.

Table 1. Experimental conditions

The number of Job(n)	50, 100, 200
Processing Time(p_i)	[1, 100] Uniform distribution
Due date(d_i)	$c \times \sum_{i=1}^{200} p_i \times [0, 1]$ Uniform distribution c : Depending on the situation(0.5, 1, 1.5)
The number of iterations	100

3.1 초기 설정 변화 실험

첫 번째 실험의 경우 초기 설정에 따른 DATC 알고리즘의 결과 값 변화를 확인하기 위하여 초기해를 (1) 임의 선택, (2) 가

Table 2. Performance results of several dispatching rules and DATC

N	*c*	random	SPT	LPT	EDD	MST	DATC
50	0.5	540,930	577,820	542,450	568,140	588,900	565,390
	1	33,218	35,051	31,664	28,019	28,521	30,101
	1.5	460,489	432,320	472,070	470,180	469,820	466,540
100	0.5	2,103,500	2,129,100	1,998,200	2,140,600	2,177,500	2,142,900
	1	70,368	73,047	71,309	57,803	57,995	69,296
	1.5	2,098,300	1,895,100	2,114,328	2,110,320	2,085,240	2,100,400
200	0.5	8,490,000	8,333,400	7,817,400	8,472,600	8,551,000	8,493,800
	1	172,405	170,160	167,960	138,360	138,440	177,550
	1.5	8,231,161	7,540,200	8,457,200	8,445,500	8,446,300	8,439,300

장 짧은 공정시간을 필요로 하는 작업부터 선택(SPT, Shortest Processing Time), (3) 가장 긴 공정시간을 필요로 하는 작업부터 선택(LPT; Longest Processing Time), (4) 납기일이 가장 빠른 작업부터 선택(EDD; Earliest Due Date), (5) 납기일까지 여유시간이 작은 작업부터 선택(MST; Minimum Slack Time) 등 5가지 방식으로 구하여 동일한 실험조건으로 기존의 DATC 알고리즘(납기일에 10을 곱한 값과 0사이의 값을 임의로 선택하여 값이 큰 순서대로 초기해를 설정) 결과와 비교하였다. 실험 조건은 기존 연구(Prabhu, 2000)와 동일한 조건을 사용하여 튜닝 파라미터 *k*의 값은 0부터 2사이의 값을 사용하였다.

<Table 2>에 정리된 초기 설정 변화에 대한 실험결과에서 보듯이 초기해로 SPT, LPT, EDD, MST와 같은 Dispatching rule의 결과를 사용하였을 경우 기존의 알고리즘 보다 많게는 20% 정도 향상된 결과 값을 나타내었다. 납기일이 타이트한 경우(*c* = 1)에는 납기일(d_i)과 관련된 Dispatching rule(EDD, MST)을 사용하는 것이 좋은 결과 값을 보였고, 납기일이 촉박한 경우(*c* = 0.5)는 LPT, 여유로운 경우(*c* = 1.5)는 SPT rule을 사용하는 것이 좋은 결과 값을 나타냈다. 따라서 좋은 초기해가 DATC의 성능에 긍정적인 영향을 준다는 사실을 확인할 수 있었고 적용하는 문제에 따라 다른 Dispatching rule의 효과가 다르므로 적용문제에 적합한 Dispatching rule을 찾기 위한 방안도 필요하다고 판단된다.

3.2 가상의 도착시간($a_i(m)$) 가중치 변화 실험

DATC는 각 작업에 대한 가상의 도착시간을 이용하여 작업의 순서를 결정한다. 이렇게 결정된 스케줄에 따라 각 작업의 완료시간을 계산하며 이를 납기일과 비교하고 두 차이를 조절모수 *k*의 가중치를 반영한다. 그리고 이전 단계의 가상 도착시간($a_i(m-1)$)값을 더하여 새로운 가상의 도착시간($a_i(m)$)을 생성한다. 이 과정을 반복하면서 각 작업은 가상의 도착시간($a_i(m)$)을 값들을 계산하게 되는데 반복에 따라 생성되는 가상의 도착시간($a_i(m)$)은 다음의 식과 같다.

$$\begin{aligned} a_i(1) &= a_i(0) + kz_i(1) \\ a_i(2) &= a_i(1) + kz_i(2) \\ a_i(3) &= a_i(2) + kz_i(3) \\ &\vdots \\ a_i(m) &= a_i(m-1) + kz_i(m) \end{aligned} \quad (6)$$

반복이 계속되면서 가상의 도착시간의 크기는 누적되어 점점 커지게 된다. 반면, m번째 반복 단계에서의 현재 작업 완료시간과 납기일의 차이($z_i(m)$)는 *k*만큼의 가중치로 항상 일정하게 반영된다. 따라서 가상의 도착시간에 미치는 영향은 1/*m*의 비율로 점차적으로 줄어들게 되어 최신의 변화에 대해 둔감해진다. 따라서, 본 절에서는 가상의 도착시간 생성에 있어 이전 단계의 가상 도착시간($a_i(m-1)$)과 현재 단계(m)에서의 작업 완료시간과 납기일의 차이($z_i(m)$)의 가중치 변화를 통한 성능 변화를 살펴 보기위해 $a_i(m-1)$과 $kz_i(m)$의 가중치 변화에 따른 실험을 진행한다. 제 3장 제 1절에서 초기해 변화를 통해 해의 개선이 있었기 때문에 실험에서는 초기 설정을 변화시켜 실험을 진행하였다. 또한 납기일에 영향을 받는 EDD, MST 두 Dispatching rule의 결과가 비슷하므로 다음의 실험에서는 초기해를 EDD, LPT, SPT 세 가지만 고려한다.

3.2.1 $a_i(m-1)$과 $kz_i(m)$의 가중치 변화에 따른 실험

본 절에서는 *k*값의 변화는 유지하면서 $a_i(m-1)$과 $kz_i(m)$의 가중치 변화를 통한 알고리즘 성능 변화에 미치는 영향을 알아보고자 한다. 앞에서 설명한 것처럼 가상의 도착시간 계산 시 반복이 계속됨에 따라 m단계에서의 작업 완료시간과 만료일의 차이 또는 최신 변동이 미치는 실제 영향은 줄어들게 된다. 따라서 본 절에서는 이 최신 변동의 반영비율 조정 또는 가중치 조정이 알고리즘 성능에 어떤 영향을 미치는지 확인하고자 한다. $a_i(m-1)$과 $kz_i(m)$의 가중치 변화를 용이하게 하기 위해 가상의 도착시간을 다음과 같이 일반화하여 표현한다.

$$a_i(m) = wa_i(m-1) + (1-w)kz_i(m), \quad 0 < w < 1 \quad (7)$$

위의 식을 이용해 가상의 도착시간($a_i(m)$)을 계산하고, 이를 반복하면 m단계에서의 가상의 도착시간은 다음과 같이 된다. 초기 해를 EDD, SPT, LPT를 사용하므로 가상의 초기도착시간($a_i(0)$)은 0으로 둔다.

$$\begin{aligned} a_i(1) &= (1-w)kz_i(1) \\ a_i(2) &= wa_i(1)+(1-w)kz_i(2) \\ &= w(1-w)kz_i(1)+(1-w)kz_i(2) \\ &= (1-w)k(wz_1(1)+z_i(2)) \\ a_i(3) &= wa_i(2)+(1-w)kz_i(3) \\ &= w(1-w)k(wz_i(1)+z_i(2))+(1-w)kz_i(3) \\ &= (1-w)k(w^2z_i(1)+wz_i(2)+z_i(3)) \\ &\vdots \\ a_i(m) &= (1-w)k(\sum_{i=1}^{m} w^{m-i}z_i(i)) \end{aligned} \quad (8)$$

위의 식에서 w값이 1보다 작기 때문에 반복이 진행되면서 이전 작업 완료시간과 납기일의 차이의 가중치가 점점 작아지는 것을 알 수 있다. 따라서 최신 변동(m단계에서의 완료시간과 납기일 편차)이 가상의 도착시간 생성에 큰 영향을 줄 수 있게 된다. 이 식을 반영해 w의 값을 0.1부터 0.9까지 0.1씩 증가시키면서 $a_i(m-1)$과 $kz_i(m)$의 가중치 변화를 주면서 가상의 도착시간을 계산하고 목적함수의 변화를 알아보았다. 실험결과는 <Table 3>와 같다.

<Table 3>에 정리된 실험결과 $kz_i(m)$와 $a_i(m-1)$의 가중치를 변화시켜 가상의 도착시간($a_i(m)$)을 생성하면 초기 설정 변화만 적용(가중치 없이)한 값보다도 많게는 10% 이상 알고리즘 성능이 향상되는 것을 알 수 있다. 기존의 실험 결과와 마찬가지로 납기일이 타이트한 경우(c = 1)에는 EDD dispatching rule을 사용하는 것이 좋은 결과 값을 보였다. 하지만 기존의 실험결과와 반대로 납기일이 촉박한 경우(c = 0.5)는 SPT rule을 사용하는 것이 좋은 결과 값을 나타내고 여유로운 경우(c = 1.5)는 LPT rule을 사용하는 것이 좋지만 초기 설정 변화한 결과 값보다 좋지 않은 결과를 나타내었다.

제 3장 제 2절의 두 가지 실험을 통해 조절모수 k값의 변경을 통해 DATC 알고리즘은 좋은 해를 찾아가고 또한 $kz_i(m)$와 $a_i(m-1)$의 가중치를 변화시키면 평균적으로 DATC의 성능에 긍정적인 영향을 준다는 사실을 확인할 수 있었다. 본 장에서의 실험에서는 반복이 진행되면서 최근의 작업 완료시간과 납기일의 차이가 가상 도착시간에 영향을 크게 미치도록 하였다. 하지만 문제 상황에 따라서는 오히려 더 좋지 않은 결과 값을 나타내기도 하였다. 때문에 과거의 데이터에 가중치를 더 주는 방법과 같은 다른 접근 방식의 가중치 분배도 고려해야 할 것이다. 또한 변경해야 할 값이 조절모수 k와 가중치 w의 두 가지 모수를 변경하면서 결과 값을 얻어야 하기 때문에 계산시간이 오래 걸린다. 따라서 기존 DATC 알고리즘의 장점인 빠르게 좋은 결과 값을 얻을 수 있도록 다른 접근 방법이 필요하다고 판단된다.

Table 3. Performance result of DATC with different initial solutions and weight values

N = 50		$a_i(m) = wa_i(m-1)+(1-w)kz_i(m)$								
c	w	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.5	EDD	560,560	561,340	564,220	569,080	569,470	569,200	567,080	568,160	567,260
	LPT	572,180	568,260	560,760	553,820	545,450	529,430	523,500	542,560	555,250
	SPT	637,380	598,850	569,430	531,010	495,500	489,610	513,830	563,910	588,300
1	EDD	27,424	27,424	27,424	27,424	27,424	27,424	27,424	27,416	27,332
	LPT	412,870	367,620	271,290	171,160	101,380	62,600	43,710	36,010	33,030
	SPT	414,110	334,590	216,300	130,990	79,120	52,630	41,840	37,960	36,330
1.5	EDD	482,790	482,420	481,540	478,730	475,960	473,460	472,750	471,360	470,080
	LPT	529,210	468,370	446,890	450,270	464,650	472,490	471,620	471,080	471,830
	SPT	541,450	499,190	475,330	471,720	469,520	461,910	450,550	441,680	435,960

Table 4. Performance improvement with several initial settings and different weght values(* : best rate of improvement)

N	c	DATC	Initial settings	Improvement rate(%)	Initial settings + weight values	Improvement rate(%)
50	0.5	565,390	542,450	4.1	489,610	13.4*
	1	30,101	28,019	6.9	27,332	9.2*
	1.5	466,540	432,320	7.3*	435,960	6.6
100	0.5	2,142,900	1,998,200	6.8	1,638,100	23.6*
	1	69,296	57,803	16.4*	57,953	16.4
	1.5	2,100,400	1,895,100	9.8*	1,916,400	8.8
200	0.5	8,493,800	7,817,400	8	6,980,800	17.8*
	1	177,550	138,360	22.1	137,150	22.8*
	1.5	8,439,300	7,540,200	10.7*	7,576,400	10.2

<Table 4>의 결과를 보면 본 장의 실험을 통하여 Dispatching rule을 활용한 초기해 설정을 통하여 알고리즘 성능을 향상시킨다는 사실을 알 수 있었다. 또한 문제 상황에 따라서 $kz_i(m)$와 $a_i(m-1)$의 가중치 조정을 통해 알고리즘의 성능을 향상시킬 수 있다는 사실을 알 수 있었다. 하지만 계산시간이 기존 DATC 알고리즘보다 오래 걸린다는 단점과 때로는 가중치의 분배가 잘못되어 알고리즘의 성능을 감소시킨다. 다음 장에서는 이 실험 결과를 바탕으로 빠른 계산 속도를 유지하면서 DATC의 성능을 향상시킨 Advanced-DATC를 제안한다.

4. Advanced-Distributed Arrival Time Control

DATC는 기본적으로 각 작업의 도착시간을 제어함으로써 스케줄을 생성하고 이를 반복함으로써 좋은 스케줄을 탐색한다. 이때 각 작업에 대한 가상의 도착시간은 매번 생성되는 스케줄에 따라 작업 완료시간과 납기일에 차이가 누적되어 결정되는데 이때 시간에 따른 이 차이가 동일한 가중치로 누적된다. 그러나, 최근 스케줄에 의한 납기일 편차가 과거 스케줄에 의한 납기일 편차보다 새로운 스케줄을 생성하는데 보다 많은 정보를 제공할 것으로 판단되므로 최근 스케줄에 의한 납기일 편차를 비중이 높게 반영하여 새로운 도착시간을 결정하는 방법이 필요하다. 이를 위하여 본 논문에서는 지수 평활법을 활용하여 최신의 변동을 크게 반영하여 DATC의 성능향상을 도모한다.

지수 평활법(Exponential Smoothing)은 오늘날 가장 많이 이용되고 있는 단기예측기법으로 사용이 간단하며, 최신의 변동을 민감하게 반영할 수 있다(Holt, 1957). 가장 최근의 데이터에 가장 큰 가중치가 주어지고 시간이 지남(본 논문에서는 반복 단계)에 따라 가중치가 기하학적으로 감소된다. 이러한 장점을 이용한 A-DATC는 기존 Dispatching rule(SPT, LPT, EDD, MST)을 사용하며 각 반복 단계에서 최근의 납기일(d_i)과 작업완료시간(C_i)의 차이에 큰 가중치가 적용되어 빠르게 해의 개선을 유도한다. 또한 조절모수 k에 의한 결과 값의 변화 폭이 크기 때문에 조절모수를 변화시키면서 해 탐색 과정을 다단계로 나눠 지역최적해에 수렴하는 현상을 방지한다. 이러한 요구사항을 A-DATC 알고리즘에 반영하기 위하여 (1) 임의의 초기해 대신 기존 dispaching rule을 이용한 초기해 선정, (2) 지수평활법, (3) 조절모수 k값의 범위를 고려한다.

4.1 초기해 설정

기존 DATC에서는 초기 도착 시간을 납기일(d_i)에 10이나 100을 곱한 값과 0사이의 값을 임의로 선택하여 초기 해를 설정했다(Prabhu, 2000). A-DATC에서는 제 3장에서의 초기 설정 변화에 따른 결과를 바탕으로 기존 Dispatching rule(SPT, LPT, EDD, MST) 4가지 방법을 사용해 초기 해를 설정한다.

4.2 지수평활법의 특징을 반영한 A-DATC

지수평활법은 기간 m의 시계열 자료의 실측치 D_m에 가중치로 평활계수 α를 부여하고 같은 기간 m에 적용된 예측치 F_m에 가중치 $1-\alpha$를 부여하여 가중평균 한 값으로 기간 $m+1$의 예측치를 산정하는 예측기법이다. 식은 다음과 같다.

$$F_{m+1} = \alpha D_m + (1-\alpha) F_m \tag{9}$$

이 식을 DATC 알고리즘에 적용하기 위해 시계열 예측치 F_m는 $a_i(m)$로 시계열 자료의 실측치 D_m는 $z_i(m)$로 대체한다. 또한 이 식을 DATC 알고리즘에 맞도록 D_m을 D_{m+1}로 변형시키면 다음의 식으로 표현할 수 있다.

$$\begin{aligned}
a_i(m) &= \alpha k z_i(m) + (1-\alpha) a_i(m-1) \\
&= \alpha k z_i(m) + (1-\alpha)[\alpha k z_i(m-1) + (1-\alpha) a(m-2)] \\
&= \alpha k z_i(m) + \alpha(1-\alpha) k z_i(m-1) + (1-\alpha)^2 a(m-2) \\
&= \alpha k z_i(m) + \alpha(1-\alpha) z_i(m-1) + (1-\alpha)^2 \\
&\quad [\alpha k z_i(m-2) + (1-\alpha) a(m-3)] \\
&= \alpha k z_i(m) + \alpha(1-\alpha) z_i(m-1) + \alpha(1-\alpha)^2 \\
&\quad k z_i(m-2) + (1-\alpha)^3 a(m-3) \\
&\qquad \vdots \\
&= \alpha k z_i(m) + \alpha(1-\alpha) k z_i(m-1) + \alpha(1-\alpha)^2 \\
&\quad k z_i(m-2) + \cdots + \alpha(1-\alpha)^{m-2} \\
&\quad k z_i(2) + \alpha(1-\alpha)^{m-1} k z_i(1) + (1-\alpha)^m a(0)
\end{aligned} \tag{10}$$

위의 식에서 α는 m시간의 작업 완료시간과 납기일의 편차와 $m-1$시간의 가상의 도착시간에 대한 가중치이고 k값은 기존 DATC와 같이 조절모수이다. 위의 식을 기존 DATC의 장점인 간단하고 계산이 빠르게 진행되도록 식의 조정이 필요하다. 식 (10)의 식에서 $kz_i(m)$는 시간의 흐름에 영향을 받지 않고 $(1-\alpha)$의 가중치만 시간의 흐름에 따라 가중치가 분배되므로 지수평활법의 특징을 반영하면서 계산시간을 줄이기 위해 다음과 같이 식을 조정한다. 모든 식의 공통으로 들어가 있는 $\alpha k = 1$로 두고 $(1-\alpha)$을 새로운 조절모수 k^*로 나타내면 $a_i(m)$은 다음의 식으로 표현될 수 있다.

$$\begin{aligned}
a_i(m) &= z_i(m) + k^* a_i(m-1) \\
&= z_i(m) + k^*[z_i(m-1) + k^* a(m-2)] \\
&= z_i(m) + k^* z_i(m-1) + k^{*2} a(m-2) \\
&= z_i(m) + k^* z_i(m-1) + k^{*2}[z_i(m-2) + k^* a(m-3)] \\
&= z_i(m) + k^* z_i(m-1) + k^{*2} z_i(m-2) + k^{*3} a(m-3) \\
&\qquad \vdots \\
&= z_i(m) + k^* z_i(m-1) + k^{*2} z_i(m-2) + \cdots + k^{*m-2} \\
&\quad z_i(2) + k^{*m-1} z_i(1) + k^{*m} a(0)
\end{aligned} \tag{11}$$

즉, 조절모수 k와 가중치 w의 두 가지 모수를 동시에 결정해야 하는 단점을 새로운 조절모수 k^*를 사용하여 해결하였다. 기존 DATC 알고리즘의 식에서 조절모수 k값의 위치만 변한 모양으로 지수평활법의 특징을 반영하면서 간단한 구조로 식을 나타내면 다음의 식과 같다.

$$a_i(t) = (d_i - C_i(t)) + k^* \int_0^t a_i(t-1)dt \qquad (12)$$

식 (12)에 따라 A-DATC 알고리즘의 새로운 가상의 도착시간 $a_i(t)$는 $a_i(t-1)$에 의해 생성된 스케줄에 의한 납기일과 작업완료시간과의 편차($d_i - C_i(t)$)에 $a_i(t-1)$를 반영하여 계산된다. 기존 DATC 알고리즘 식과 마찬가지로 계산상의 편의를 위하여 m반복 단계의 새로운 도착시간 $a_i(m)$은 다음의 식을 통해 근사치를 구할 수 있다.

$$a_i(m) = z_i(m-1) + k_i a_i(m-1) \qquad (13)$$

4.3 조절모수 k^*값의 범위

기존 DATC 0-2 값을 사용하도록 권고하고 있으나 이 값의 범위가 A-DATC에도 유효한 지를 확인하기 위하여 k^*값을 변경하면서 실험을 수행하였다. <Table 7>은 초기 설정 4가지 Dispatching rule(SPT, LPT, EDD, MST)에 따른 각각의 최적 결과 값일 때의 조절모수 k^*의 값을 구한 후 평균값을 나타낸 것이다(Job의 수는 200개, 100회 반복 후 평균). 실험결과 대부분 k^*의 값이 0.1에서 2사이일 때 최적의 결과 값을 제시함을 알 수 있다. 따라서 A-DATC에서 k^*값의 범위는 기존 DATC와 같이 0과 2사이의 값으로 정한다. 3.2.1의 가중치 변화 실험에서는 ω의 크기를 0~1사이의 값으로 두어 이전 작업 완료시간과 납기일의 차이의 가중치가 점점 작아지도록 설계하여 항상 최근의 작업 완료시간과 납기일의 차이가 스케줄에 큰 영향을 미치도록 했다. 하지만 A-DATC에서는 k^*값의 범위를 0과 2사이의 값으로 두어 초기 설정에 따라서 최근의 변동이 큰 영향(k^*값이 1보다 작게)을 미치도록 하거나 이전의 변동이 큰 영향(k^*값이 1보다 크게)을 미치도록 설정하여 기존 가중치 분배의 단점을 극복했다.

Table 5. The mean value of k^* with 100 best solutions

c	SPT	LPT	EDD	MST
0.5	0.593	1.773	0.771	1.240
1	1.204	1.203	0.873	1.031
1.5	1.414	0.289	0.712	0.675

4.4 A-DATC

지금까지 설명한 세 가지 고려사항을 반영하여 A-DATC 알고리즘은 다음과 같다.

알고리즘 1 : n Job에 대한 A-DATC :

1단계 : 초기해 설정을 Dispatching rule(SPT, LPT, EDD, MST) 중 하나를 사용하여 초기 해를 구한다.

2단계 : A-DATC 알고리즘을 사용하여 도착시간을 업데이트 한다.

For i = 1 to n Do

$A_i(m) = [d_i - C_i(m)] + k^* A_i(m-1)$

FCFS 규칙에 근거한 도착시간으로 새로운 스케줄링.

End For

3단계 : 해의 개선이 일정 반복 수 이상 개선이 되지 않으면 알고리즘 반복을 그만둔다.

4단계 : 1단계로 넘어가 다른 초기 설정으로 2~3단계를 반복한다.

5단계 : 초기설정 변화에 따른 결과 값을 바탕으로 가장 좋은 스케줄을 찾는다.

제 4장에서는 지수평활법을 이용한 새로운 알고리즘인 A-DATC를 제안했다. 초기해 설정 변화를 통해 결과 값의 성능이 차이가 나는 것을 확인하고 초기해의 품질에 의한 알고리즘의 영향을 줄이기 위해 잘 알려진 Dispatching rule을 사용했다. 또한 지수평활법의 특징을 이용하여 해의 개선이 빠르게 이루어지도록 하였다. 따라서 제 5장에서는 이러한 A-DATC 알고리즘의 강점을 확인하고 수치 예제를 들어 성능을 평가한다.

5. 실험 및 결과 분석

5.1 실험

본 장에서는 제안된 A-DATC 알고리즘의 성능을 평가하기 실험을 진행하여 그 결과를 확인하고 기존문헌의 연구결과와 비교하여 효율성을 입증한다. 또한 제 3장의 실험결과를 바탕으로 Dispatching rule을 활용하여 초기해 설정을 적용한 DATC 알고리즘과 초기해 설정 및 가중치 조정을 적용한 DATC 알고리즘의 성능이 기존 DATC 알고리즘 보다 성능이 향상됨을 알 수 있었다. 따라서 본 실험에서는 기존 DATC 알고리즘의 성능뿐만 아니라 초기해 설정변화에 따른 효과와 가중치 분배에 따른 효과를 함께 확인하기 위해 (1) 기존 DATC, (2) DATC 초기설정 변화, (3) DATC 초기설정 및 가중치 변화를 통한 알고리즘의 성능과 3.2.1의 실험 결과를 통해 (3)의 가중치 분배 방법에 변화를 둔 (4) A-DATC 의 총 4가지 알고리즘의 성능을 평가하였다. 실험 조건은 기존 연구(Prabhu, 2000)와 동일한 조건을 사용하였다. 또한 다양한 상황을 반영하기 위해 3가지 상황(납기일이 촉박한 경우 : 0.5, 납기일이 타이트한 경우 : 1, 납기일이 여유로운 경우 : 1.5)으로 나누고 문제의 크기도 진행해야 할 작업이 20, 30, 50, 100, 200, 500의 6가지 문제로 나누어 총 18가

지의 상황을 가정하고 실험을 진행하였다. 각 실험 조건에 대하여 스케줄링 문제가 임의로 생성에 의한 영향을 최소화하기 위하여 각 조건에 대해 실험을 100회 반복한 후 그 평균값으로 해당 조건의 결과 값으로 사용하였다.

5.2 결과 분석

A-DATC 알고리즘의 스케쥴링 문제에 대한 효과를 증명하기 위하여 제 5.1절에 제시된 조건에 따라 스케쥴링 문제를 생성하고 이를 제 2장에서 설명한 평균 제곱편차(MSD; Mean Squared Deviations)에 대하여 기존 DATC와 제안된 A-DATC를 비교분석한다. A-DATC 알고리즘의 성능은 기존 DATC 알고리즘의 성능과의 차이를 나타내는 식 (13)의 POD(Percentage of Deviation)로 평가하며, 알고리즘이 우수할수록 음수값을 갖게 된다. POD 식은 다음과 같다(Lee, 2009).

$$\frac{MSD_{A-DATC} - MSD_{DATC}}{MSD_{DATC}} \times 100\% \qquad (13)$$

<Table 6>는 제 5.1절에서 제시한 4가지 알고리즘 대해 문제 크기 및 상황을 변경시켜 실험한 결과를 보여준다. 초기해로 기존 dispatching rule을 사용한 경우 기존 DATC 알고리즘에 의한 결과보다 평균 8.3% 향상 되었고, 초기해로 기존 dispatching rule을 사용하고 가상의 도착시간 가중치 변화를 통해서는 평균 11.3% 해가 개선됨을 확인 했다. 하지만 납기일이 여유로운 경우(c = 1.5)와 납기일이 타이트한 경우(c = 1)의 일부분 에서는 가중치에 변화를 준 알고리즘의 결과가 더 좋지 않음을 알 수 있다. 따라서 가중치의 변화가 항상 좋은 결과만을 보장하지 못함을 확인했다. 이러한 단점을 극복하기 위해 설계된 A-DATC 알고리즘은 기존 DATC 알고리즘 대비 평균 18% 향상을 보였고, 다른 모든 경우의 알고리즘 결과 값보다 우수한 성능을 보였다.

<Table 7>은 <Table 6>의 실험 결과를 바탕으로 기존 DATC 알고리즘의 결과와 초기설정변화, 초기설정 및 가중치 변화, A-DATC의 성능의 차이를 좀 더 구체적으로 확인하기 위해 작성한 표이다. 납기일이 타이트한 경우(c = 1) 초기설정변화를 통해 다른 경우보다 해가 크게 향상되며, 초기설정 및 가중치 변화를 통해서는 납기일이 촉박한 경우(c = 0.5)와 타이트한 경우(c = 1) 해의 개선이 크게 이루어짐을 알 수 있다. 반대로 납기일이 여유로운 경우(c = 1.5)는 다른 문제 상황과는 다르게 가중치를 변경시켰을 경우 초기설정만 변경시켰을 때보다 좋지 않은 결과 값을 나타냈다. 이러한 단점을 극복한 A-DATC 알고리즘의 결과 역시 다른 알고리즘 보다 향상된 결과 값을 제시하지만 납기일이 여유로운 경우는 다른 경우의 절반의 성능만을 보여준다. 본 연구에서 초기 설정을 4가지 기본적인 Dispatching rule만 사용했지만 다른 Dispatching rule을 사용한다면 납기일이 여유로운 경우에도 다른 경우와 비슷한 성과를 얻을 수도 있을 것으로 판단된다.

<Table 7>의 결과 값이 통계적으로 유의한지 확인하기 위하여 Minitab 16 프로그램을 이용하여 검증을 실시하였다. 변형된 DATC의 결과와 A-DATC의 성능의 차이는 paired t-test 유의수준 0.05%로 분석하였다. 본 실험은 <Table 6>에서 알고리즘의 성능차이가 적게 나타난 문제 크기 20, 30, 50의 세 가지 경우에 대한 검정을 통해 A-DATC 알고리즘의 우수함을 보이고자 한다. 이에 대한 실험 결과는 <Table 8>과 < Table 9>와 같다.

Table 6. A comparisons of experimental result between DATC and A-DATC

Problem		Orignal DATC	DATC + Initial settings	DATC + Initial settings + weight factor	A-DATC	POD(%)
c	n					
0.5	20	101,250	100,420	93,100	87,070	-14.0
	30	211,380	205,880	186,730	172,380	-18.5
	50	565,390	542,450	489,610	435,170	-23.0
	100	2,147,700	1,895,100	1,638,100	1,596,100	-25.7
	200	8,493,800	7,817,400	6,980,800	6,367,670	-25.0
	500	51,806,000	47,284,300	43,218,200	38,542,800	-25.6
1	20	12,491	12,210	12,230	11,450	-8.3
	30	14,369	13,484	13,560	12,621	-12.2
	50	30,101	28,019	27,332	25,893	-14.0
	100	69,296	57,803	57,953	53,398	-22.9
	200	177,550	138,360	137,150	117,312	-34.0
	500	497,941	420,968	419,324	346,353	-30.4
1.5	20	79,302	75,990	75,980	73,385	-7.5
	30	185,470	172,960	174,330	168,510	-9.1
	50	466,540	432,320	435,960	410,570	-12.0
	100	2,100,400	1,895,100	1,916,400	1,851,600	-11.8
	200	8,439,300	7,540,200	7,576,400	7,255,970	-14.0
	500	53,329,300	50,324,900	50,318,700	46,780,200	-12.3

Table 7. POD values of DATC + Initial setting, DATC + Initial setting + weighted and A-DATC

c	POD (DATC + Initial setting)	POD (DATC + Initial setting + weight factor)	POD (A-DATC)
0.5	-6.0	-15.2	-22.0
1	-11.6	-12.0	-20.3
1.5	-7.4	-6.9	-11.1

Table 8. A comparisons between DATC(Initial settings) and A-DATC in 100 Samples

Problem		DATC + Initial settings		A-DATC		t	p
c	n	Mean	SD	Mean	SD		
0.5	20	93,100	37,580	87,070	33,920	15.73	0.000
	30	186,730	61,708	172,380	56,202	22.77	0.000
	50	489,610	124,018	435,170	113,158	25.91	0.000
1	20	12,230	10,413	11,450	9,381	11.45	0.000
	30	13,560	11,421	12,621	9,584	13.59	0.000
	50	27,332	29,241	25,893	25,193	10.89	0.000
1.5	20	75,980	45,078	73,385	43,896	9.29	0.000
	30	174,330	102,868	168,510	102,359	9.73	0.000
	50	435,960	214,308	410,570	205,804	8.90	0.000

주) $^{*}p < 0.05$ with paired t-test.

Table 9. A comparisons between DATC(Initial settings + weight factor) and A-DATC in 100 Samples

Problem		DATC + Initial settings + weight factor		A-DATC		t	p
c	n	Mean	SD	Mean	SD		
0.5	20	100,420	34,018	87,070	33,920	6.83	0.000
	30	205,880	55,001	172,380	56,202	9.92	0.000
	50	542,450	115,538	435,170	113,158	13.26	0.000
1	20	12,210	10,259	11,450	9,381	8.20	0.000
	30	13,484	10,128	12,621	9,584	9.26	0.000
	50	28,019	26,710	25,893	25,193	8.89	0.000
1.5	20	75,990	44,389	73,385	43,896	8.31	0.000
	30	172,960	104,868	168,510	102,359	10.12	0.000
	50	432,320	205,538	410,570	205,804	9.54	0.000

주) $^{*}p < 0.05$ with paired t-test.

실험 결과 모든 경우의 실험에서 p값이 0.0001 이하로 A-DATC의 성능이 유의한 차이를 나타냈다. 따라서 A-DATC 알고리즘이 두 변형된 DATC에 의한 알고리즘의 결과보다 우수함을 알 수 있다.

6. 결론 및 향후 연구 과제

실제 작업 환경에서는 다양한 상황에 의한 여러 가지 불확실성 때문에 빠르게 리스케줄링이 되어야 한다. 본 연구의 바탕이 되는 DATC 알고리즘은 이러한 불확실성에 빠르게 대처할 수 있는 단일 공정 문제를 해결하기 위한 알고리즘이다. 본 연구에서는 기존 DATC 알고리즘을 분석하고 기존 연구들이 제시하지 않았던 초기 스케줄과 가상의 도착시간의 가중치의 변화를 통한 해의 개선을 유도했다. 계산 시간이 기존 DATC에 비하여 약간 증가하지만 지수평활법의 특징을 이용하여 짧은 시간 내에 좋은 결과 값을 제시하였다. 실험을 통해 모든 상황에서 문제 크기에 상관없이 A-DATC가 우수한 결과를 나타냄을 확인했다.

향후 연구방안으로는 본 연구에서 초기설정 변화로 기존에 알려져 있던 EDD, LPT, SPT, MST 등의 단순 Dispatching rule만 사용했지만 A-DATC 알고리즘의 성능 향상을 위해 다른 초

기 설정 방법을 개발해야 할 것이다. 또한 본 연구에서 다룬 단일공정 문제 이외에 다중공정 문제와 같은 복잡한 문제를 해결하기 위한 연구가 필요하다.

참고문헌

Aarts, E. H. L., Frans, M. J., and Habers, E. H. A. (1986), Parallel implementations of the statistical cooling algorithm, *VLSI Journal*, **4**(3), 209-238.

Abdul-Razaq, T. S. and Potts, C. N. (1988), Dynamic programming state-space relaxation for single-machine scheduling, *Journal of Operational Research Society*, **39**(2), 141-152.

Bagchi, U., Sullivan, R. S., and Chang, Y. L. (1987), Minimising mean squared deviation of completion times about a common due date, *Management Science*, **33**, 894-906.

Baker, K. R. (1974), Introduction to Sequencing and Scheduling, Wiley, New York.

Cho, S.-H. and Erkoc, M. (2009), Design of predictable production scheduling model using control theoretic approach, international, *Journal of Production Research*, **47**(11), 2976-2993.

Glover, F. (1986), Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research*, **13**(5), 533-549.

Goldberg, D. E. (1989), Genetic algorithms in search, optimisation, and machine learning, Addison-Wesley, Boston.

Holt, C. C. (1957), Forecasting Seasonal and Trends by Exponentially Weighted Moving Averages, Carnegie Institute of Technology, Pittsburgh.

Kim, J.-D. and Ok, C.-S. (2008), Distributed feedback control algorithm for dynamic truck loading scheduling problem, *Applied Mathematics and Computation*, **1**, 275-284.

Lee, S.-H. and Lee, S.-W. (2009), Ant colony system for vehicle routing problem with simultaneous delivery and pick-up under time windows, *Journal of the Korean Institute of Industrial Engineers*, **35**(2), 160-170.

Li, G. (1997), Single machine earliness and tardiness scheduling, *European Journal of Operational Research*, **96**, 546-558.

Parunak, H. D. (1991), Characterising the manufacturing scheduling problem, *Journal of Manufacturing Systems*, **10**(3), 241-259.

Prabhu, V. V. and Duffie, N. A. (1999), Nonlinear dynamics in distributed arrival time control of heterarchical manufacturing systems, *IEEE Transactions on Control Systems Technology*, **1**(6), 724-730.

Prabhu, V. V. (2000), Performance of Real-time Distributed Arrival Time Control in Heterarchical Manufacturing Systems, *IIE Transactions*, **32**, 323-331.

Pirlot, M. (1992), General local search heuristics in combinatorial optimisation : a tutorial, *Belgium Journal of Operations Research*, **32**, 8-67.

Sabuncuoglu, I. and Karabuk, S. (1999), Rescheduling frequency in an FMS with uncertain processing times and unreliable machines, *Journal of Manufacturing Systems*, **18**(4), 269-283.

Simchi-Levi, D. (2006), Designing and Managing the Supply Chain, **3**, McGraw-Hill, New York.