Journal of KIIT. Vol. 17, No. 2, pp. 1-11, Feb. 28, 2019. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2019.17.2.1

재유입 다중칩 조립라인을 위한 인공신경망 기반의 다중 목적 함수 디스패치 규칙 선택 기법

허재석*, 박종헌**

Artificial Neural Network Based Multi-Objective Rule Selection Dispatcher for Re-Entrant Multiple-Chip Product Assembly Lines

Jaeseok Huh*, Jonghun Park**

본 연구는 2018년도 정부(교육부) 의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업이며 (NRF-2015R1D1A1A01057496), 서울대학교 공학연구원의 지원에도 감사를 드립니다.

요 약

최근 반도체 제조사들은 고집적, 고성능의 반도체 생산을 위해 다중칩 제품의 생산에 집중하고 있다. 쌓아 올리는 칩의 수가 증가함에 따라, 다중칩 조립라인의 Die Attach 및 Wire Bonding 단계를 반복하는 재유입 흐름이 발생하게 되고 이는 흐름 시간의 증가와 설비가동률 하락을 일으킨다. 본 논문은 흐름 시간을 줄임과 동시에 설비가동률을 향상시킬 수 있는 인공신경망 기반의 규칙 선택 디스패처를 제안한다. 제안된 디스패처는 시뮬레이터에 의해 생성된 다중칩 조립라인 데이터에 따라 각 디스패치 규칙의 선호도를 학습한다. 학습이 완료된 후 디스패처는 디스패치 규칙 중 하나를 선택하여 디스패치 의사결정을 수행하게 된다. 또한, 제안된 디스패처가 성능 및 계산 시간 측면에서 기존 방법들 보다 우수함을 실험결과로 증명하였다.

Abstract

Recently, the semiconductor manufacturers have focused on the production of multiple chip products (MCPs) to achieve high capacities while preserving compactness. As the number of chips to be stacked increases, a re-entrant flow that repeats die attach and wire bonding assembly stages in MCP production is generated, which causes increased flow time and decreased resource utilization. In this paper, we propose a rule selection dispatcher (RSD) based on an artificial neural network, which reduces the flow time and increases the resource utilization. RSD learns the preferences of each dispatching rule according to assembly line data generated by a simulator. Then the proposed dispatcher performs lot dispatching decisions by selecting one of the dispatching rules. The experiments showed that the proposed dispatcher outperformed the existing methods in terms of the performance and computation time.

Keywords

re-entrant manufacturing lines, dispatching rules, intentional delay, artificial neural network, flow time, utilization

* 서울대학교 산업공학과(교신저자)
- ORCID: https://orcid.org/0000-0003-0055-919X
** 서울대학교 산업공학과 교수
- ORCID: https://orcid.org/0000-0001-7505-110X

· Received: Dec. 31, 2018, Revised: Jan. 24, 2019, Accepted: Jan. 27, 2019
· Corresponding Author: Jaeseok Huh
Dept. of Industrial Engineering, Seoul National University
1 Gwanak-ro, Gwanak-gu, Seoul 151-744, Korea
Tel.: +82-2-880-7361, Email: laonhjs@gmail.com

Ⅰ. 서 론

최근의 반도체 제조공정은 고집적, 고성능의 반도체 생산을 위해 기존의 단칩 제품 생산에서 다수의 칩을 쌓아 올리는 다중칩 제품(MCP, Multiple-Chip Product) 생산을 중심으로 발전하고 있다. 다중칩 제품은 쌓아 올리는 칩의 수가 증가함에 따라 DA(Die Attach)와 WB(Wire Bonding) 단계를 여러 번 반복하는 재유입 흐름(Re-entrant Flow)을 발생시키게 된다[1]. 이는 다중칩 제품 조립라인의 복잡도가 상승하는 원인이 되고, 그에 따라 흐름 시간(Flow Time)이 증가하고 설비가동률(Resource Utilization)이 하락하는 문제가 발생한다[2].

한편, 다중칩 조립라인에서 하나의 로트(Lot)를 하나의 설비(Resource)에 할당하는 것을 로트 디스패치 의사결정(Lot Dispatching Decision)이라 한다. 특히, WB 단계는 각 다이(Die)에 여러 개의 와이어(Wire)를 납땜하기 위한 작업시간이 오래 걸리기 때문에 DA 단계보다 병목 단계로 간주 된다[3].

이러한 특성이 있는 다중칩 조립라인에 대해, 디스패치 규칙 기반의 방법론들이 제안되어 왔다[4]-[8]. 이러한 방법론들은 대안을 산출하는데 걸리는 시간이 짧고 구현이 용이하다는 장점이 있지만, 대부분의 디스패치 규칙이 일반적으로 하나의 목적함수를 다루기 위해 설계되었기 때문에 동시에 여러 개의 목적함수를 고려하기에는 한계가 있다. [2]의 연구에서는 인공신경망을 활용하여 로트들의 대기시간과 설비들의 유휴시간의 합을 최소화하는 것을 목표로 하는 연구가 수행되었다. 해당 연구는 의도적인 지연을 고려하여 우수한 로트 디스패치 의사결정을 도출하였으나, 의사결정에 대한 설명력이 부족하고 공정의 수가 많아지면 계산 시간이 증가한다는 단점이 존재한다.

본 논문은 [2]의 연구의 단점을 해결하기 위해서 상황에 따라 적절한 디스패치 규칙을 선택하는 인공신경망(Artificial Neural Network: ANN) 기반의 디스패치 규칙 선택 디스패처(RSD, Rule Selection Dispatcher)를 제안한다. 구체적으로 로트 디스패치 의사결정이 요구되는 시점에, 흐름 시간과 설비가동률 측면에서 가장 우수하다고 판단되는 디스패치 규칙이 학습된 인공신경망에 의해 선택된다. 또한, 사용자가 두 개의 목적함수 간의 가중치를 설정하면 조립라인의 특성에 맞는 로트 디스패치 의사결정을 수행할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 다중칩 제품 조립라인의 로트 디스패치 문제를 정의하며, 3장에서는 본 논문에서 제안하는 방법론에 대해 자세히 기술한다. 4장에서는 실험내용과 그 결과를 논의하며, 마지막으로 5장에서는 결론과 본 연구의 한계점 및 향후 연구에 대하여 상술하도록 한다.

Ⅱ. 문제 정의

본 논문에서는 [2]에서 기술된 DA와 WB단계를 포함하는 다중칩 제품 조립라인에서의 로트 디스패치 의사결정 문제를 고려한다. 주어진 조립라인에서, 로트들의 대기시간과 WB 설비들의 유휴시간을 감소시키는 것이 본 연구의 목표이다. 이를 달성하기 위해, 우리는 비 병목 단계인 DA 단계에서의 로트 디스패치 의사결정에 집중하고, 병목 단계에 해당하는 WB 단계에서는 설비가동률을 최대화할 수 있도록 작업시간이 가장 긴 로트를 우선하여 설비에 할당한다. 이는 비 병목 단계에서의 로트 디스패치 의사결정이 조립라인의 재공(WIP, Work-In-Process)의 양을 관리하는 데 큰 영향을 미치기 때문이다[9].

고려하고 있는 다중 칩 제품 조립라인에는 N_M개의 설비 유형이 존재한다. q번째 설비 유형은 M_q로 표현되며, 각 설비 유형에는 n_q대의 설비들이 할당되어 있다. 이때, q번째 설비 유형에 할당된 s번째 설비는 $R_{q,s}$로 표현된다. 로트 디스패치에 필요한 N_J개의 잡 유형이 존재하며, 임의의 잡 유형 J_i는 미리 정해진 순서로 지정된 공정의 시퀀스로 구성되어 있다. J_i의 j번째 공정은 $O_{i,j}$로 표현되며, $A(O_{i,j})$는 $O_{i,j}$를 처리할 수 있는 설비 유형의 집합을 의미한다. J_i에는 n_i개의 로트가 존재하며 k번째 로트는 $L_{i,k}$로 표현된다. 각 로트 $L_{i,k}$의 작업시간은 로트에 할당된 칩의 수에 비례하며, $I(L_{i,k})$는 처리되어야 할 남은 공정 중에 가장 작은 인덱스를 반환한다.

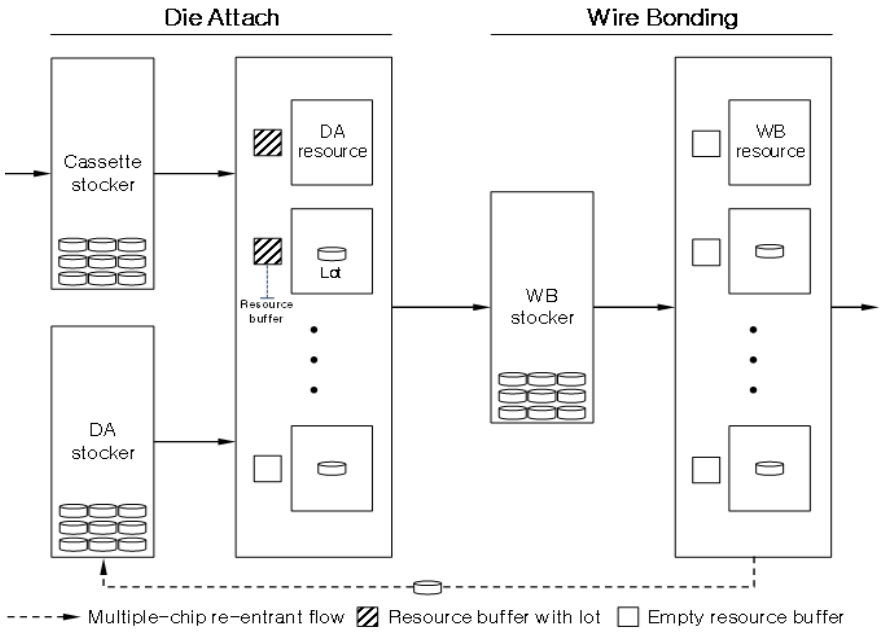

그림 1. 다중칩 제품 조립 라인에서 DA와 WB 단계의 로트 흐름
Fig. 1. Lot flow of DA and WB stages in MCP assembly line

그림 1에서 보듯이 다중칩 조립라인에는 cassette, DA, WB의 세 가지 종류의 stocker가 존재한다. Cassette stocker는 조립라인에 새로 도착한 로트가 첫 번째 DA 공정을 기다리는 공간이다. DA stocker는 WB 공정을 완료한 로트가 다음 DA 공정을 시작하기 전에 대기하는 stocker이다. 마지막으로, WB stocker에서는 DA 공정을 완료한 로트가 다음 WB 공정이 시작하기를 기다리게 된다.

각 설비 앞에는 설비가 유휴 상태가 될 때까지 로트가 공정을 기다리는 공간인 설비 버퍼(Resource Buffer)가 존재하며, 버퍼의 용량은 1로 가정한다. 공정이 한 번 시작되면 로트는 공정이 완료될 때까지 어떤 간섭도 받지 않으며 한 번에 하나의 설비가 하나의 공정을 수행한다. 또한, 다른 잡 유형의 로트 간의 셋업 교체 시간은 없다고 가정한다.

후보 로트(Candidate Lot)는 DA 설비 $R_{q,s}$의 버퍼가 비게 될 때 $R_{q,s}$에 할당 가능한 로트를 의미한다[2]. 로트의 상태에 따른 후보 로트들의 유형을 표 1에 나타내었다. 로트 디스패치 기법은 설비의 버퍼가 비어있는 $R_{q,s}$와 $M_q \in A(O_{i,J(L_{i,k})})$를 만족하는 후보 로트 $L_{i,k}$ 간의 할당을 결정한다.

표 1. 상태와 의도적인 지연에 따른 후보 로트들
Table 1. Types of candidate lots according to status and intentional delay

Status	Description	Intentional delay
In-Cassette-Stocker	Lots waiting in the cassette stocker	None
In-DA-Stocker	Lots that have finished WB operations and are waiting for the next DA operations	None
To-DA-Stocker	Lots being moved to the DA stocker after WB operations	Less
At-WB-Resource	Lots in process at WB resources	More

DA 설비에 대해, 상태가 In-Cassette-Stocker 또는 In-DA-Stocker인 후보 로트가 선택되면, 로트는 즉시 DA 설비 버퍼로 이동할 수 있다. 특히, cassette stocker의 로트가 설비에 할당되어 로트가 출발하면 흐름 시간의 측정이 시작된다. 반면, DA stocker로 이동 중인 로트나 WB 설비에서 작업 중인 로트가 선택되면, DA stocker까지 이동하거나 WB 설비에서의 남은 작업을 수행하기 위한 추가 시간이 필요하므로 DA 설비에 의도적인 시간 지연이 발생하게 된다[2].

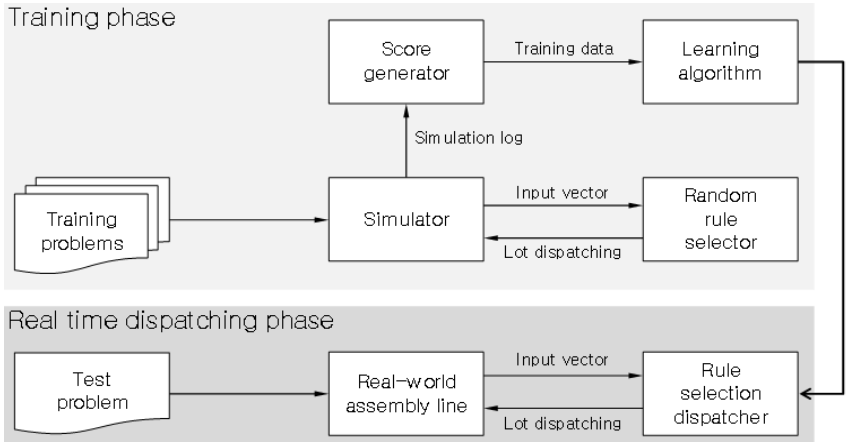

그림 2. 제안 기법의 전체적인 구조도
Fig. 2. Overall framework of the proposed approach

III. 규칙 선택 디스패처

그림 2는 제안 기법의 전반적인 작동 과정을 나타낸다. 학습 단계에서는 구현된 시뮬레이터를 사용하여 그림의 다중칩 제품 조립라인을 모사한다. 이 때, 디스패치 규칙 중에 임의로 하나를 선택하는 RRS(Random Rule Selector)가 로트 디스패치 의사결정을 수행하는 역할을 담당한다. 그 후 생성된 시뮬레이션 로그의 성능이 측정되고, 점수가 매겨진 시뮬레이션 로그는 RSD에 내장된 인공신경망을 학습시키는 알고리즘의 학습 데이터로 사용된다. 실시간 디스패치 단계에서는 시뮬레이터는 로트 디스패치 의사결정이 요구될 때 학습이 완료된 디스패처를 호출하게 된다.

3.1 디스패처의 구조

RSD는 1개의 입력층, 3개의 은닉층, 1개의 출력층으로 구성되어 있다. 입력층에는 37개의 노드가 존재하고, 3개의 은닉층은 각각 64, 32, 16개의 노드를 포함한다. 마지막으로 출력층은 단 하나의 노드를 가진다. 은닉층의 개수와 각 층에 포함되는 노드의 수는 검증 오차율 측정 실험을 통해 결정되었으며, 실험 결과는 4장에 제시하였다. 신경망에 비선형 변환능력을 부여하기 위해서, 각 은닉층 앞에는 식 (1)과 같이 정의되는 ReLU(Rectified Linear Unit) 활성화 함수를 적용하였다.

$$f(z) = \max(0, z) \tag{1}$$

입력층에 주어지는 입력 벡터는 상태 및 액션 벡터의 두 그룹으로 나눌 수 있다. 전자는 특정 DA 설비 $R_{q,s}$의 설비 버퍼가 비었을 때 다중칩 조립라인의 상태를 설명하는 역할을 담당하며, 후자는 $R_{q,s}$에 할당할 로트를 선택하는데 사용되는 디스패치 규칙을 나타낸다. 우리는 먼저 상태 벡터의 구성요소들을 소개한 후 액션 벡터에 사용되는 디스패치 규칙들을 기술한다.

표 2는 $R_{q,s}$에 대한 상태 벡터의 구성요소들을 요약하여 나타낸다. 상태 벡터로부터 RSD는 후보 로트들의 특성과 조립라인 내의 로트의 흐름을 파악할 수 있게 된다. 이를 토대로 RSD는 $R_{q,s}$와 조립라인 상황에 적합한 디스패치 규칙을 선택하게 된다. 구체적으로 상태 벡터는 $R_{q,s}$가 유휴 상태가 될 때까지 남은 시간, 후보 로트들의 통계량, 조립라인 내의 로트의 분포로 분류되는 총 26개의 요인으로 구성되어 있다.

후보 로트와 관련된 12개의 요인은 각 로트가 갖는 네 가지 속성의 최소, 최대 및 평균값을 나타낸다. 네 가지 속성은 로트가 cassette stocker를 떠난 시각, 로트가 보유한 칩의 수, 처리되어야 할 남은 공정의 수, $R_{q,s}$에서의 작업시간 등이다. 로트의 다른 속성들은 본 논문에서 고려하는 목적함수와 관련이 없으므로 상태 벡터에서 제외되었다.

표 2. RSD의 $R_{q,s}$에 대한 상태 벡터의 구성요소
Table 2. Components of the state vector for $R_{q,s}$ of RSD

Categories	Description	Number of features
DA resource	The time remaining until $R_{q,s}$ becomes idle	1
Statistics of candidate lots	Earliest, latest, and average time when lots depart the cassette stocker	3
	Minimum, maximum, and average number of chips in lots	3
	Minimum, maximum, and average number of remaining operations assigned to lots	3
	Minimum, maximum, and average processing time of lots on $R_{q,s}$	3
Lot distribution	The number of lots in each location of the MCP assembly line	13

표 3. RSD의 $R_{q,s}$에 액션 벡터를 생성할 때 사용되는 디스패치 규칙들
Table 3. Dispatching rules used to generate the action vector for $R_{q,s}$ of RSD

Rules	How to select one among candidate lots
FIFO	The oldest lot that has been dispatched from the cassette stocker
LOR	The lot with the smallest number of successor operations
MOR	The lot with the largest number of successor operations
LARGE	The lot with the largest number of chips
SMALL	The lot with the smallest number of chips
SPT	The lot with the shortest processing time on $R_{q,s}$
LPT	The lot with the longest processing time on $R_{q,s}$
SNQ	After grouping lots with the same $O_{i,I(L_{i,k})}$, select the lot of the group with the smallest number of lots.
LNQ	After grouping lots with the same $O_{i,I(L_{i,k})}$, select the lot of the group with the largest number of lots.
FLNQ	After grouping lots with the same $O_{i,I(L_{i,k})}$ in the WB stocker, select the lot corresponding to the group with the smallest number of lots.
STOCKER	Select one of lots in the cassette or DA stockers (The lots in question do not cause an intentional delay on $R_{q,s}$).

로트들의 분포를 나타내는 각 요인은 조립라인에서 로트가 존재할 수 있는 모든 영역을 포함하는 13개의 위치 중 하나에 해당하는 로트의 수를 나타낸다. 이러한 요인들을 사용하여 RSD는 로트들의 분포뿐 아니라 전체 공정의 진행 속도도 파악할 수 있게 된다.

표 3은 $R_{q,s}$에 대한 액션 벡터를 생성하는데 사용되는 디스패치 규칙과 각 규칙이 다중칩 제조라인에서 후보 로트 중 하나의 로트를 선택하는 방식을 요약한 것이다. 기존의 디스패치 규칙 중에 로트의 흐름 시간과 설비가동률에 영향을 주는 규칙들만 선정하였다[6][10][11]. STOCKER 규칙은 다중칩 조립라인에 알맞게 본 연구에서 개발된 규칙이다.

표 3의 디스패치 규칙을 이용하여 액션 벡터가 생성되는 과정은 다음과 같다. 로트 디스패치 의사결정이 요구될 때, 모든 원소의 값이 0인 11차원의 열벡터가 생성된다. 그 후, 특정 디스패치 규칙이 선택되면 해당 규칙에 대응되는 차원에 1이 할당된다. 예를 들어, MOR이 선택된 경우에 액션 벡터는 $(0,0,1,0,0,0,0,0,0,0,0)$이며, LPT가 선택된 경우에는 $(0,0,0,0,0,0,1,0,0,0,0)$ 형태의 액션 벡터가 생성된다.

출력층은 입력 벡터의 선호도 점수를 나타내는 역할을 담당한다. 각 노드의 모든 값은 서로 다른 단위 간의 불일치를 해결하기 위해 min-max 정규화를 사용하여 0과 1 사이의 범위로 변환된다[12].

3.2 학습 단계

학습 단계에서는 표 3에 제시된 디스패치 규칙 중 하나를 임의로 선택하는 RRS가 생성된 각 학습용 문제를 여러 번 반복하여 풀게 된다. 각 문제의 모든 로트 디스패치 의사결정이 RRS에 의해 결정되면, 디스패치된 로트와 할당된 DA 설비로 구성된 전체 디스패치 기록이 포함된 시뮬레이션 로그와 그것에 대응되는 입력 벡터가 점수 생성기로 전송된다.

점수 생성기는 디스패치 된 로트의 대기시간과 로트를 처리한 WB 설비의 유휴시간을 기반으로 각 로트 디스패치 의사결정을 평가한다. 여기서, 대기시간은 DA 설비에서 공정이 완료된 로트가 WB

stocker에서 머무는 시간을 나타낸다. WB 설비의 유휴시간은 설비가 로트를 작업하기 시작한 시각에서 설비의 마지막 공정이 완료되었던 시각을 뺀 값으로 계산된다.

각 로트 디스패치 의사결정에 대해 점수 생성기는 min-max 정규화 방식을 따라 0과 1 사이의 점수를 책정한다[2][12]. 대기 및 유휴점수를 산출하는 방법은 각각 식 (2), (3)과 같다. 여기서 w과 l은 각각 로트 디스패치 의사결정과 관련된 대기 및 유휴시간을 나타낸다.

$$s_w = \max(-\triangle W(s_{\max} - s_{\min}) + s_{\max}, s_{\min}) \quad (2)$$
$$-\triangle W = \frac{w - w_{\min}}{w_{\max} - w_{\min}}$$

$$s_l = \max(-\triangle L(s_{\max} - s_{\min}) + s_{\max}, s_{\min}) \quad (3)$$
$$-\triangle L = \frac{l - l_{\min}}{l_{\max} - l_{\min}}$$

위의 식들에서 $w_{\min}$과 $w_{\max}$는 w의 모든 가능한 값 중에 최솟값과 최댓값을 나타내고, $l_{\min}$과 $l_{\max}$는 l의 모든 가능한 값 중에서 최솟값과 최댓값을 나타낸다. 또한 $s_{\min}$과 $s_{\max}$는 가능한 점수 중 최솟값과 최댓값을 의미하는데, 본 연구에서는 각각 0과 1로 설정되었다.

식 (2), (3)에서, $w_{\max}$와 $l_{\max}$는 각각 w와 l값들의 평균값의 두 배로 설정된다. 이는 큰 값의 w와 l이 원하지 않은 높은 점수로 변환되는 것을 방지하여 균형 잡힌 학습 데이터를 구성하는 것을 가능하게 한다. 본 연구의 목적은 로트 디스패치 의사결정의 점수를 정확하게 예측하는 게 아니라 로트의 대기시간과 WB 설비의 유휴시간을 최소화할 것으로 예측되는 로트 디스패치 의사결정을 수행하는 것이기 때문에 이러한 처리는 합리적이다.

두 가지 유형의 점수인 s_w와 s_l을 사용하여 생성된 학습 데이터를 기반으로 각 점수에 대응되는 두 개의 인공신경망이 각 점수를 예측하도록 학습된다. 우리는 손실함수로 $(s_w - s_w^{pre})^2$ 와 $(s_l - s_l^{pre})^2$ 정의되는 제곱 오차를 사용하였으며, 손실함수의 값을 최소화하기 위해 역전파 알고리즘을 사용하였다[13]. 여기서 s_w와 s_l은 식 (2)와 (3)을 사용하여 로트 디스패치 의사결정에 대해 계산된 점수이며, s_w^{pre}와 s_l^{pre}는 각각 학습 시에 예측한 대기 및 유휴시간에 대한 예측된 점수이다.

3.3 실시간 디스패치 단계

실시간 디스패치 단계에서는, DA 설비의 버퍼가 비게 되면 해당 설비에 대해 표 3에 제시된 디스패치 규칙의 수만큼의 입력 벡터가 생성된다. 동시에 여러 대의 DA 설비가 비어있는 경우엔, 가장 먼저 유휴 상태로 진입할 것으로 예상되는 DA 설비에 대해 위의 과정이 수행된다. 생성된 입력 벡터들은 RSD의 입력층에 주어지고, RSD는 각 벡터에 대해 두 가지 유형의 점수를 예측한다. 각 입력 벡터의 최종 선호 점수는 식 (4)에 의해 계산된다.

$$totalscore = \lambda_w \times s_w^{pre} + \lambda_l \times s_l^{pre} \quad (4)$$

위의 식에서 λ_w와 λ_l의 합은 항상 1로 일정하다. λ_w의 값은 로트의 흐름 시간에 대한 중요도를 의미하고, λ_l의 값은 병목 단계의 설비가동률의 중요도를 나타낸다. 식 (4)로 산출된 점수 값이 가장 큰 입력 벡터에 관련된 디스패치 규칙이 로트 디스패치 의사결정으로 선택되고, 해당 규칙에 따라 후보 로트들 중에 하나의 로트가 DA 설비에 할당된다.

제안된 RSD는 미리 학습된 인공신경망을 사용하여 간단한 계산을 통해 대안을 산출하기 때문에 일반적인 메타 휴리스틱 기법보다 신속하게 로트 디스패치 의사결정을 도출할 수 있다. 메타 휴리스틱 기법은 충분한 시간이 주어지면 우수한 성능의 의사결정을 도출할 수 있지만, 다중칩 조립라인은 실시간으로 디스패치 의사결정이 요구되기 때문에 메타 휴리스틱 기반의 기법들은 실제 현장에서 사용하기 어렵다. 반면에, RSD는 짧은 시간 내에 상대적으로 우수한 디스패치 의사결정을 산출하기 때문에 현장의 다중칩 조립라인에 도입될 수 있다.

Ⅳ. 실험 결과

RSD의 성능을 검증하기 위해서 표 4와 같이 설비 수, 잡 유형 수, 평균 공정 수가 서로 다른 네

가지 데이터 세트를 준비했다. 실제 다중칩 조립 라인의 규모는 굉장히 크기 때문에 원활한 실험을 수행하기 위해서, 데이터 세트 1, 2와 데이터 세트 3, 4는 각각 현실의 다중칩 제품 조립라인의 1/10과 1/3 규모로 설정되었다[1]. 로트에 포함된 칩의 수는 100과 750 사이의 유니폼 분포(Uniform Distribution)를 따른다. 이는 실제 다중칩 조립라인의 생산 요구량은 항상 모든 공정을 완료하는데 2일 이상이 소요된다는 현장의 관행을 충족시키기 위한 조건이다[1].

각 데이터 세트마다 로트에 포함된 칩의 수와 각 잡 유형의 할당된 로트의 수를 변경해가면서 100개의 문제를 생성했다. 구체적으로, 디스패처의 학습과 검증하는데 30개와 20개의 문제가 사용되었고, 학습이 끝난 후 성능을 평가하는데 나머지 50개의 문제를 사용하였다.

학습 단계에서는, RRS를 사용하여 각 문제의 모든 로트 디스패치 의사결정을 100번씩 수행함으로써 시뮬레이션 로그를 생성하였다.

표 4. 실험에 사용된 데이터 세트에 대한 설명
Table 4. Descriptions on the datasets used for the experiments

Dataset No.	Number of DA resources	Number of WB resources	Number of job types	Avg. number of operations
1	4	12	5	524.90
2	4	12	8	571.30
3	20	50	5	1609.84
4	20	50	8	1670.28

표 5. 은닉층 구조에 따른 대기시간과 유휴시간에 대한 검증 오차율
Table 5. Validation error of waiting and idle time according to the structure of hidden layers

Number of hidden layers	Number of nodes per each hidden layer	Validation error of waiting time	Validation error of idle time
2	(32, 64)	4.99E-05	8.54E-02
2	(64, 32)	4.91E-05	8.55E-02
3	(16, 32, 64)	4.64E-05	8.57E-02
3	(32, 32, 32)	4.68E-05	8.58E-02
3	(64, 32, 16)	**4.44E-05**	**8.53E-02**
4	(8, 16, 32, 64)	4.65E-05	8.67E-02
4	(64, 32, 16, 8)	4.78E-0.5	8.54E-02

생성된 시뮬레이션 로그의 점수는 식 (2), (3)을 기반으로 점수 생성기에 의해 계산되었다. 데이터 세트 1, 2와 데이터 세트 3, 4는 DA 단계에서 한 문제당 각각 약 260개와 800개의 디스패치 의사결정이 수행된다. 학습에 총 30개의 문제가 사용되고 하나의 문제를 100번씩 풀게 되므로 학습에 사용되는 데이터의 수는 각각 약 78만개와 240만개가 된다.

표 5는 학습 단계에서 RSD의 구조를 결정하기 위해 은닉층의 수와 층별 노드 수를 변화시켜가며 대기 시간과 유휴시간 대한 검증 오차율을 측정한 결과이다. 측정에 사용된 데이터는 데이터 세트 3번이며, 각 오차율마다 가장 낮은 수치를 기록한 경우 숫자를 굵게 표시하였다. 은닉층의 수가 증가할수록 학습 오차율은 감소하였으나, 과적합의 영향 때문에 검증 오차율이 특정 수준 이하로 줄어들지 않았다. 3개의 은닉층에 각각 64, 32, 16개의 노드가 할당된 구조가 대기시간과 유휴시간 측면에서 가장 낮은 검증 오차율 값을 기록하였다. 따라서 해당 구조를 RSD의 은닉층으로써 사용하였다.

로트 디스패치 의사결정의 성능을 평가하기 위해 [2]의 연구에서 사용된 척도인 AWT(Average Waiting Time)과 AIT(Average Idle Time)를 사용하였으며, 각 척도의 계산식은 식 (5), (6)에 나타나 있다. 식 (5)의 $c_{i,k}$와 $r_{i,k}$는 각각 $L_{i,k}$의 최종 WB 공정이 완료되는 시각과 $L_{i,k}$가 cassette stocker를 떠난 시각을 의미하고, $t_{i,k}$는 $L_{i,k}$이 설비들에서 작업된 시간의 총합을 나타낸다.

식 (6)에서 $f_{q,s}$는 $R_{q,s}$가 마지막 공정을 완료하는 시각을 의미하며, $t_{q,s}$는 시작 시점부터 $f_{q,s}$까지 $R_{q,s}$가 로트를 처리한 시간의 총합을 나타낸다. Ω는 WB 단계에 속하는 설비의 인덱스 집합을 의미한다.

$$AWT = \frac{\sum_{i=1}^{N_J}\sum_{k=1}^{n_i}(c_{i,k} - r_{i,k} - t_{i,k})}{\sum_{i=1}^{N_J} n_i} \quad (5)$$

$$AIT = \frac{\sum_{\forall q \in \Omega}\sum_{s=1}^{n_q}(f_{q,s} - t_{q,s})}{\sum_{\forall q \in \Omega} n_q} \quad (6)$$

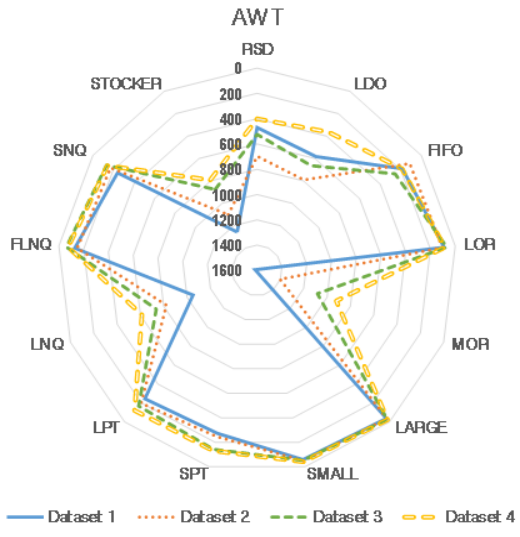

그림 3. 디스패치 기법에 따른 각 데이터 세트별 AWT 성능
Fig. 3. AWT results of dispatching methods for each dataset

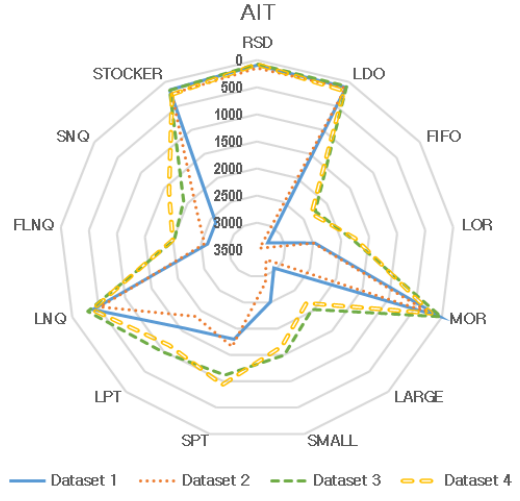

그림 4. 디스패치 기법에 따른 각 데이터 세트별 AIT 성능
Fig. 4. AIT results of dispatching methods for each dataset

성능 비교 실험에 사용된 디스패치 방법은 제안된 RSD, 표 3에 나타낸 디스패치 규칙들, [2]에서 제안된 디스패처로 총 13가지이며 [2]에서 제안된 디스패처는 LDO로 표기하였다. LDO는 로트 디스패치 의사결정이 요구될 때, 모든 후보 로트들을 하나씩 평가하여 가장 높은 점수의 로트를 DA 설비에 할당하는 디스패처이다. RSD의 경우, 식 (4)의 λ_w와 λ_l의 값은 0.5로 동일하게 설정하였다. 13가지 디스패치 방법의 AWT와 AIT측면의 성능 비교 결과는 각각 그림 3과 그림 4에 제시되어 있다. 그래프의 외곽에 값이 나타날수록 해당 디스패치 기법이 우수한 성능을 도출한 것을 의미한다.

FIFO, LOR, SMALL, LARGE, SPT, LPT, FLNQ, SNQ는 AWT 측면에서 제안된 기법보다 더 우수한 성능을 보였다. 구체적으로, FIFO는 cassette stocker에 있는 로트에 낮은 우선순위를 부여하고, LOR은 남은 공정의 수가 적은 로트를 선호하기 때문에 흐름 시간을 굉장히 낮게 유지한 것으로 보인다. LARGE, SMALL, SPT, LPT, FLNQ, SNQ의 경우, 각 규칙이 선호하는 로트가 해당 로트의 모든 공정이 완료되어 조립라인을 빠져나가기 전까지 반복적으로 선호될 확률이 높다. 그 결과, DA stocker로 이동 중인 로트나 WB 설비에서 작업 중인 로트가 빈번하게 선택되어 새로운 로트가 cassette stocker에서 출발하는 것을 방해하게 된다. 따라서 재공의 양이 줄어들어 흐름 시간이 감소하지만, WB 설비들의 유휴시간이 증가하게 된다.

한편, MOR, LNQ, STOCKER는 AIT측면에서 다른 디스패치 규칙보다 우수한 성능을 보였다. MOR은 남은 공정이 많은 로트에 우선순위를 할당하기 때문에 설비들이 유휴 상태가 되는 것을 허용하지 않는 경향이 있다. LNQ는 공정 유형 별로 로트 수의 균형을 유지하려는 규칙이기 때문에, 재공의 양이 적은 초반에 각 잡 유형의 첫 번째 공정에 해당하는 로트가 주로 선택된다. 이는 재공의 양을 늘리는 결과로 이어져서 설비들의 유휴시간이 감소하게 된다. STOCKER는 DA 설비에 의도적인 지연을 일으키는 로트를 선택하지 않으므로 자연스럽게 WB 설비들의 유휴시간도 감소하게 된다.

표 6은 그림 3과 그림 4에 표현된 각 디스패치 기법의 성능을 AWT와 AIT의 측면에서 비교하기 용이하도록 하나의 형태로 나타낸 것이다. 각 칸의 위와 아래에 표시된 수치는 각각 해당 디스패치 기법이 도출한 AWT와 AIT 값이며, RSD가 LDO보다 우수한 성능을 보인 경우에는 숫자를 굵게 표시하였다. LDO와 제안 기법인 RSD는 각 성능 지표에서는 기존의 디스패치 규칙을 압도하는 결과를 보이지 않았다.

표 6. 디스패치 방법에 따른 각 데이터 세트별 AWT, AIT 성능
Table 6. AWT and AIT results of dispatching methods for each dataset

Dispatching methods	Dataset No.			
	1	2	3	4
FIFO	184.8	112.61	255.44	187.1
	3270.02	3444.46	2249.04	2326.17
LOR	89.83	103.58	67.88	83.96
	2469.37	2491.06	1808.09	1750.62
MOR	1617.11	1402.88	1083.38	920.05
	56.34	252.57	14.85	201.6
LARGE	36.94	33.43	26.02	22.93
	3050.79	3257.45	2031.38	2185.66
SMALL	59.23	48.32	53.37	41.57
	2511.02	2877.93	1498.68	1665.95
SPT	272.29	246.41	142.23	131.03
	1801.54	1659.38	1119.56	935.54
LPT	245.08	196.95	167.88	120.1
	1566.29	1858.86	979.87	1149.35
LNQ	1051.87	822.63	736.12	612.23
	386.17	527.45	291.34	378.46
FLNQ	126.95	146.65	59.16	73.76
	2614.87	2562.81	2020.68	1994.19
SNQ	242.1	172.82	134.46	143.01
	2612.8	2194.51	1918.46	1605.32
STOCKER	1257.88	1102.1	877.81	792.41
	168.26	247.4	172.97	262.67
LDO	583.5	792.02	664.78	365.84
	111.71	129.41	92.39	188.56
RSD	**470.21**	**697.43**	**524.14**	401.66
	90.69	147.91	**70.79**	**77.55**

그러나 표 6에 따르면, 디스패치 규칙이 두 성능 지표 중 한 지표에 대해서는 취약한 반면에, 두 기법은 우수한 성능을 도출한 것을 확인할 수 있었다. 즉, RSD는 로트의 흐름 시간을 감소시키면서 병목 단계의 설비가동률을 높게 유지할 수 있는 디스패처라 할 수 있다. LDO와 비교했을 때, AWT와 AIT 측면 모두에서 세 개의 데이터 세트에 대해 RSD가 더 우수한 성능을 보였다.

그림 5는 RSD와 LDO가 공정 수 변화에 따라 주어진 로트를 설비에 할당하는데 걸린 계산 시간을 나타낸다. 공정의 수에 상관없이 RSD가 LDO보다 로트 디스패치 의사결정 산출에 더 적은 계산 시간을 요구하는 것을 확인할 수 있었다. 또한, 공정의 수가 늘어남에 따라 LDO는 RSD보다 계산 시간의 증가 폭이 훨씬 크게 나타났다. 이 결과는 로트 디스패치 의사결정이 요구될 때, 두 가지 디스패처의 의사결정 대안 수의 차이로 설명할 수 있다. RSD는 공정의 수가 증가하더라도 대안의 수는 언제나 학습에 사용된 디스패치 규칙의 수로 일정하지만, LDO는 대안의 수가 의사결정 시점의 후보 로트의 수와 같다. 이 때문에, 공정의 수가 증가하게 되면 LDO는 평가해야 할 대안의 수가 많아지므로 계산 시간이 큰 폭으로 증가하게 된다.

디스패치 패턴을 분석하기 위해, RSD가 데이터 세트 3번에 속하는 문제의 로트 디스패치 의사결정을 수행할 때 선택했던 디스패치 규칙들을 시간에 따라 히트맵(Heat Map)으로 시각화하여 그림 6에 나타내었다.

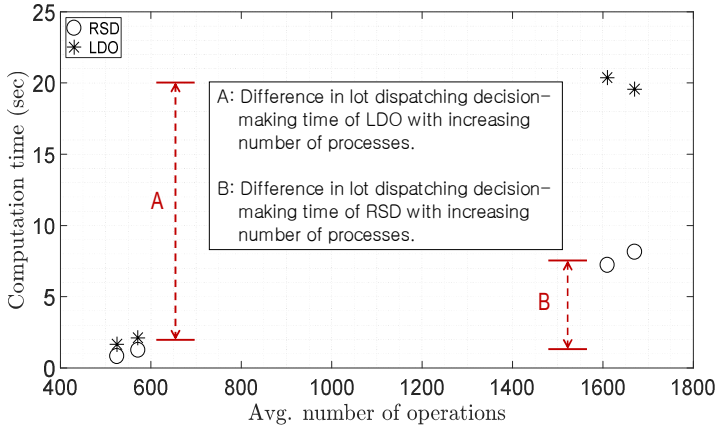

그림 5. 평균 공정 수의 변화에 따른 RSD와 LDO의 계산 시간
Fig. 5. Computation time of RSD and LDO according to the average number of operations

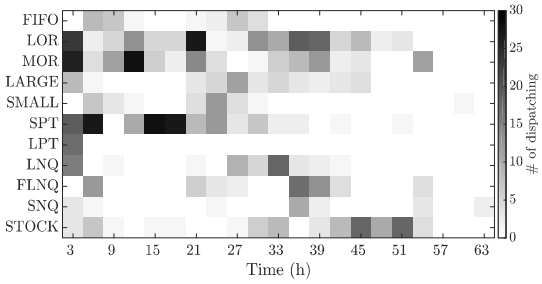

그림 6. 시간에 따른 디스패치 규칙의 선택 빈도
Fig. 6. Selection frequencies of dispatching rules over time

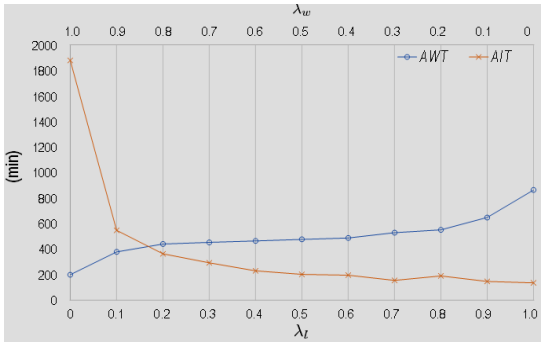

그림 7. λ_w와 λ_l에 따른 AWT와 AIT의 변화
Fig. 7. Changes in AWT and AIT depending on λ_w and λ_l

그림 6에 나타난 값은 3시간 간격으로 선택된 디스패치 규칙의 수를 나타낸다. [2]의 연구에서 제안된 LDO는 각 로트에 점수를 책정하여 디스패치 의사결정을 수행하기 때문에, 로트가 설비에 할당되는 근거를 로트의 상태 정보에서만 추론할 수 있다는 한계가 존재했다. 반면에, RSD는 의사결정 시점에 적합한 디스패치 규칙을 선택하기 때문에 사용자 입장에서 로트가 설비에 할당되는 이유에 대해 이해하기 용이하다는 장점이 존재한다.

그림 6에서 뚜렷한 경향성이 발견되지 않았지만, 재공의 양이 상대적으로 적은 초반에는 MOR을 위주로 다양한 규칙들이 선택되는 양상이 관찰된다. 또한, 재공의 양이 증가한 후에는 흐름 시간을 감소시킬 수 있도록 SPT와 LOR과 같은 규칙들이 선택되는 것을 확인할 수 있었다. 한편, RSD는 FIFO, LNQ, SNQ와 같은 규칙들에 대한 선택 빈도는 시간과 관계없이 매우 낮았다.

마지막으로, 식 (4)의 두 가중치의 값에 따른 RSD의 AWT와 AIT 측면에서의 성능 변화를 관찰하기 위한 실험을 수행하였다. 그림 7은 데이터 세트 1번의 하나의 문제를 λ_w과 λ_l의 값을 바꾸어 가며 푼 결과를 나타낸 것이다. 예상대로 λ_w값이 증가하면 AWT가 감소하고 λ_l이 증가하면 AIT가 감소하는 경향이 관찰되었다. 이 실험을 통해서 RSD는 주어진 다중칩 조립라인의 상황이나 현장 운영자의 판단에 따라 가중치를 조절함으로써 상황에 맞는 로트 디스패치 의사결정을 수행할 수 있음을 확인할 수 있었다.

V. 결론 및 향후 과제

본 논문에서는 다중칩 조립라인의 로트의 흐름시간 감소와 병목 단계의 설비가동률 증가를 위한 디스패치 규칙 선택 기법을 제안하였다. 제안 기법은 로드 디스패치 의사결정이 요구될 때, 기존의 디스패치 규칙 중에 가장 상황에 적절하다고 판단되는 디스패치 규칙을 선택하여 로트를 설비에 할당한다. 최신 기법과 기존의 디스패치 기법과의 비교실험을 통해 의사결정의 성능과 계산속도 측면에서 모두 제안 기법이 우수한 결과를 도출함을 확인하였다. 또한, 제안 기법을 활용하면 가중치 설정을 통해 조립라인의 상황에 적절한 로트 디스패치 의사결정을 수행할 수 있다.

향후, 제안 기법의 성능을 고도화하기 위해 궁극적인 보상을 최대화하는 정책을 수립할 수 있는 강화학습을 제안 기법에 도입하는 것을 연구할 계획이다. RSD의 액션 벡터를 표현하는 디스패치 규칙들은 강화학습의 액션 공간을 정의하는데 사용될 수 있다. 그러나 모든 시뮬레이션이 완료된 후에 로트 디스패치 의사결정의 가치를 나타내는 현재 방식은 강화학습의 즉각적인 보상함수에 부합하는 방향으로 더 연구되어야 한다.

References

[1] B. S. Chung, J. Lim, I. B. Park, J. Park, M. Seo, and J. Seo, "Setup change scheduling for semiconductor packaging facilities using a genetic

algorithm with an operator recommender", IEEE Transactions on Semiconductor Manufacturing, Vol. 27, No. 3, pp. 377-387, Aug. 2014.

[2] J. Huh, I. Park, S. Lim, B. Paeng, J. Park, and K. Kim, "Learning to dispatch operations with intentional delay for re-entrant multiple-chip product assembly lines", Sustainability, Vol. 10, No. 11, pp. 4123-4143, Nov. 2018.

[3] D. Quadt, "Simulation-based Scheduling of Parallel Wire-bonders with Limited Clamp&Paddles", in Proceedings of the 38th Conference on Winter Simulation, Monterey, California, pp. 1887-1892, Dec. 2006.

[4] C. Pickardt, J. Branke, T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Generating dispatching rules for semiconductor manufacturing to minimize weighted tardiness", in Proceedings of the 2010 Winter Simulation Conference, pp. 2504-2515, Dec. 2010.

[5] L. Li, Z. Sun, M. Zhou, and F. Qiao, "Adaptive dispatching rule for semiconductor wafer fabrication facility", IEEE Transactions on Automation Science and Engineering, Vol. 10, No. 2, pp. 354-364, Apr. 2013.

[6] D. M. Chiang, R. S. Guo, and F. Y. Pai, "Improved customer satisfaction with a hybrid dispatching rule in semiconductor back-end factories", International Journal of Production Research, Vol. 46, No. 17, pp. 4903-4923, Sep. 2008.

[7] F. Zhou, C. Wu, and C. Yu, "Dynamic dispatching for re-entrant production lines – A deep learning approach", in 2017 13th IEEE Conference on Automation Science and Engineering (CASE), pp. 1026-1031, Aug. 2017.

[8] S. Jia, D. J. Morrice, and J. F. Bard, "A performance analysis of dispatch rules for semiconductor assembly & test operations", Journal of Simulation, pp. 1-18, Mar. 2018.

[9] J. Potoradi, O. S. Boon, and S. J. Mason, "Using simulation-based scheduling to maximize demand fulllfilment in a semiconductor assembly facility", in Proceedings of the Winter Simulation Conference, Vol. 2, pp. 1857-1861, Dec. 2002.

[10] R. Haupt, "A survey of priority rule-based scheduling", Operations-Research-Spektrum, Vol. 11, No. 1, pp. 3-16, Mar. 1989.

[11] M. Đurasević and D. Jakobović, "A survey of dispatching rules for the dynamic unrelated machines environment", Expert Systems with Applications, Vol. 113, pp. 555-569, Dec. 2018.

[12] J. Han, J. Pei, and M. Kamber, "Data mining: concepts and techniques", Elsevier, 2011.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", Nature, Vol. 323, No. 6088, pp. 533, Oct. 1986.

저자소개

허 재 석 (Jaeseok Huh)

2013년 8월 : 서울대학교 산업공학과(공학사)
2013년 9월 ~ 현재 : 서울대학교 산업공학과 박사과정
관심분야 : 제조 시스템 디스패치/스케줄링 , 강화학습, 사례기반추론

박 종 헌 (Jonghun Park)

1990년 2월 : 서울대학교 산업공학과(공학사)
1992년 2월 : 서울대학교 산업공학과(공학석사)
2000년 2월 : Georgia Institute of Technology, 산업시스템공학과 (공학박사)
2019년 2월 현재 : 서울대학교 산업공학과 교수
관심분야 : 기계학습, 산업응용