JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 28 No. 12, pp. 79-87, December 2023
https://doi.org/10.9708/jksci.2023.28.12.079

Shipyard Skid Sequence Optimization Using a Hybrid Genetic Algorithm

Min-Jae Choi*, Yung-Keun Kwon**

*Student, School of IT Convergence, University of Ulsan, Ulsan, Korea
**Professor, Department of Electrical, Electronic and Computer Engineering, University of Ulsan, Ulsan, Korea

[Abstract]

In this paper, we propose a novel genetic algorithm to reduce the overall span time by optimizing the skid insertion sequence in the shipyard subassembly process. We represented a solution by a permutation of a set of skid ids and applied genetic operators suitable for such a representation. In addition, we combined the genetic algorithm and the existing heuristic algorithm called UniDev which is properly modified to improve the search performance. In particular, the slow skid search part in UniDev was changed to a greedy algorithm. Through extensive large-scaled simulations, it was observed that the span time of our method was stably minimized compared to Multi-Start search and a genetic algorithm combined with UniDev.

▸ **Key words**: Subassembly process, Skid sequence optimization, Conveyor environment, Hybrid Genetic Algorithm

[요 약]

본 연구는 조선소 소조립 공정에서 스키드 투입 순서 최적화를 통해 전체 작업시간을 단축시키는 새로운 유전 알고리즘 방법을 제안한다. 하나의 해는 스키드 번호들의 순열로 표현되며 그러한 표현에 적합한 유전 연산자들을 적용하였다. 또한 탐색 성능의 개선을 위해 UniDev라 불리우는 기존의 휴리스틱 알고리즘을 적절하게 변형하여 유전 알고리즘과 결합하였다. 특히 UniDev에서 느린 스키드 탐색 부분을 그리디 알고리즘의 형태로 변경하였다. 매우 큰 규모의 문제에 대해 시뮬레이션을 수행한 결과 Multi-Start 탐색과 UniDev기반 혼합형 유전알고리즘에 비해 본 연구에서 제안하는 방법이 안정적으로 작업시간을 최소화함을 관찰하였다.

▸ **주제어**: 소조립 공정, 스키드 순서 최적화, 컨베이어 환경, 혼합형 유전 알고리즘

• First Author: Min-Jae Choi, Corresponding Author: Yung-Keun Kwon
*Min-Jae Choi (chlalswo0925@naver.com), School of IT Convergence, University of Ulsan
**Yung-Keun Kwon (kwonyk@ulsan.ac.kr), Department of Electrical, Electronic and Computer Engineering, University of Ulsan
• Received: 2023. 11. 13, Revised: 2023. 12. 14, Accepted: 2023. 12. 14.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

선박의 건조 과정은 크게 선각 공정, 의장 공정, 도장 공정 등 세 부분으로 나뉘는데 이중 선각 공정은 전처리, 소조립, 그리고 대조립 공정의 순서로 이루어진다. 특히 소조립 공정은 스키드라 불리우는 블록이 순차적으로 사전에 계획된 일정대로 컨베이어에 투입되는데, 공정별 작업 시간 차이로 인해 병목현상이 발생하게 된다 [1]. 이 문제를 해결하기 위해 조선소는 작업장의 구조 변경, 크기 확장 등 물리적인 환경 개선을 고려할 수도 있지만 예산과 공간적인 한계로 인해 단기간에 개선이 어려운 실정이다. 따라서 작업장의 물리적인 환경 변화 없이도 전체 작업 소요 시간을 최소화하기 위한 소조립 공정 일정 최적화 연구가 필요하다.

실제로 선박 건조 소조립 공정의 작업 일정 최적화에 대한 연구가 일부 진행되었다. 예를 들면 기존의 한 관련 연구[2]에서 Simulated Annealing(SA)[3]을 사용하여 작업자의 배치와 운용을 감안한 스키드 투입 순서 최적화를 시도하였는데 이를 통해 전체 작업 시간을 감소시킬 수 있음을 보였다. 하지만 작은 크기의 고정된 문제(스키드 12개, 작업 6개)에 대해서만 분석했다는 한계가 있다. 또 다른 연구[4]에서는 동시간에 진행되는 작업 소요 시간의 차이에 관한 평균 절대 편차를 정의하고 이를 점진적으로 개선하는 휴리스틱을 제안하여 다양한 시뮬레이션 데이터에 대해 좋은 성능을 보여주었다. 하지만 그 방법은 지역 탐색 기법의 하나로써 초기해에 따라 성능이 달라지고 지역 최적해에서 벗어나지 못한다는 한계가 있다.

본 연구에서는 기존 연구[4]에서 제안된 휴리스틱 방법의 한계점을 개선하여 성능을 향상시키기 위한 방법을 제시한다. 구체적으로 UniDev라 불리우는 기존 휴리스틱과 유전 알고리즘(Genetic Algorithm: GA)을 결합하여 혼합형 유전 알고리즘(Hybrid GA: HGA)를 개발하였다. GA는 생물의 진화 매커니즘을 모방한 확률적 최적화 알고리즘인데 순수 GA는 지역 최적해로의 수렴이 어렵다는 단점이 있다. 이러한 단점을 보완하기 위해 휴리스틱을 결합한 혼합형 GA가 대안이 될 수 있다. 그러나 GA와 휴리스틱을 단순하게 결합하는 경우 지역 탐색 시간이 길어져 오히려 소요 시간에 비해 성능 향상 속도가 느리게 되는 비효율성이 나타날 수 있다. 이에 본 연구에서는 기존 UniDev 휴리스틱의 탐색 범위를 적절하게 조절하여 지역 탐색 시간을 줄이는 대신 GA를 통한 전역 탐색 시간을 충분하게 소모하도록 설계하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 선박 소조립 공정 최적화 관련 기존의 연구들을 소개하고 3장에서는 본 논문에서 다루는 스키드 투입 최적화 문제를 정의한다. 4장에서는 본 연구에서 제안하는 혼합형 GA의 작동 원리를 설명한다. 5장에서는 다양한 시뮬레이션 데이터에 적용한 실험 결과를 제시하며 6장에서는 본 연구의 결론을 설명한다.

II. Related Works

조선소 조립 공정 최적화는 스마트 조선소 분야의 주요 관심사 중 하나로서 여러 최적화 기법이 고려되어왔다. 예를 들면 연구[5]에서 제약 충족 문제 기법을 이용하여 베이의 할당과 조립 시간의 스케줄링을 하여 조립 비용의 효율성을 높이고자 하였다. 또 다른 연구[6]에서는 유전 알고리즘을 이용하여 조선소의 판넬 블록 조립공장의 스케줄링 문제를 다루었고, 또 다른 연구[7]에서는 혼합 유전 알고리즘을 이용하여 선각 평블록 조립공장 스케줄링 문제를 해결하고자 노력했다.

본 논문에서 다루어지는 소조립 공정에 대해서도 휴리스틱과 유전 알고리즘 등 다양한 방법론을 사용한 연구가 있었다. 예를 들면 선행 연구[4]는 작업 소요 시간의 평균 절대 편차를 이용하여 소조립 스케줄링 문제를 다루었다. 또 다른 연구[2]는 Simulated Annealing을 사용하여 작업자의 배치와 운용을 감안한 스키드 투입 순서 최적화를 시도하였다. 또한 유전 알고리즘을 이용하여 소조립 공정에서의 로봇 용접 공정의 스케줄링을 최적화하고 검증하기도 하였다 [8]. 또 다른 연구[9]에서는 소조립 공정의 스케줄링 최적화 문제를 해결하기 위한 순수 유전 알고리즘 기반 방법을 제안하였는데 유전 알고리즘의 다양한 특성, 예를 들어 선택, 교차, 변이 등을 미세 조정하면서 주목할 만한 성능을 달성하였다. 이처럼 스마트 조선소 분야에서 소조립 공정은 효율성을 높이기 위해 다양한 형태의 공정 최적화 문제가 존재하며 이를 해결하기 위해 서로 다른 방법론이 시도되었음을 알 수 있다. 그러나 기존 연구들은 실험 환경에서 스키드의 개수를 최대 개수를 수십 개 내외의 작은 문제로 고정하여 실제 조선소에서 수백 개에 이르는 경우를 검증하지 못하였다. 특히 순수 유전 알고리즘을 사용한 연구들의 경우 큰 규모의 문제에서 지역 탐색을 효과적으로 하지 못한다는 한계점이 있었다. 이에 본 연구에서는 혼합형 유전 알고리즘을 고안하여 단일 컨베이어에서 스키드 투입 순서 스케줄링을 최적화함으로써 총 공정 시간을 최소화하는 문제에 대해 큰 규모에서도 좋은 성능을 보일 수 있는 새로운 해법을 제안하고 검증한다.

III. Problem Formulation

선박 건조시 소조립 공정은 스키드라는 단위로 이루어지며, 이 과정에는 판계, 배재, 취부, 자동 용접, 수동 용접, 사상과 같이 여러 단계가 포함된다. 이러한 스키드는 전처리 공정을 통해 형성된 강재로 구성되어 컨베이어에 순차적으로 투입된다. 이 연구에서는 다음과 같이 선행연구[4]와 동일한 문제 정의를 따른다.

- 스키드의 각 공정은 순차적으로 이루어져야 하며, 한 공정이 완료되기 전에는 다음 공정을 시작할 수 없다.
- 컨베이어 상의 모든 스키드 공정이 완료되기 전까지 해당 스키드는 이동할 수 없다.
- 다양한 종류의 스키드가 있으며, 스키드의 종류에 따라 각 공정의 작업시간은 서로 다를 수 있다.

스키드의 투입 순서의 효율성을 평가하기 위해, 컨베이어 상의 작업 시간 및 대기 시간을 합산한 전체 작업 시간을 계산해야 한다. 다음은 이를 위해 정의된 변수들 이다.

- N : 스키드의 개수
- M : 공정 개수 ($M \leq N$)
- Π : 스키드 투입 순서, ($\{1,2,3,\cdots,N\}$의 순열)
- S_i^π : 스키드 투입 순서 Π의 i번째 스키드 ($i \in \{1,2,3,\cdots,N\}$)
- $T_{i,j}^\pi$: S_i^π의 j번째 공정 작업 시간, ($j \in \{1,2,3,\cdots,M\}$)

이제 스키드 투입 순서 Π에 대한 전체 작업 시간 $span(\Pi)$은 다음과 같이 정의한다.

$$span(\Pi) = \sum_{t=1}^{N+M-1} MAX(T_{i,t-i+1}^\pi, T_{i+1,t-i}^\pi, \cdots, T_{j-1,t-j+2}^\pi, T_{j,t-j+1}^\pi)$$

where $i = MAX(t-M+1)$ and $j = MIN(N,t)$.

스키드 투입 순서에 따라 전체 작업 시간은 크게 영향을 받는다. Fig. 1과 2는 스키드 투입 순서에 따른 전체 작업 시간 차이에 관한 예를 나타내고 있다. Fig. 1은 A, B, C 세 종류의 스키드에 대한 작업 X, Y, Z의 소요 시간을 의미한다. Fig. 2는 각 스키드 투입 순서가 A→B→C(Case1), B→C→A(Case2), C→B→A(Case3)인 경우에 대한 총 작업 시간을 보여준다. Case 1에서는 실제 작업 시간과 대기 시간이 각각 18시간 및 8시간이고, Case 2에서는 각각 15시간 및 6시간이었으며, Case 3에서는 각각 12시간 및 3시간이 소요되었다. 이처럼 스키드 투입 순서에 따라 $span(\Pi)$이 크게 달라진다. 따라서, 최적의 스케줄링을 선정함에 있어서는 단순히 개별 작업의 소요 시간

Skid	Processing Time		
	Process X	Process Y	Process Z
A	3	3	1
B	2	4	3
C	1	2	5

Fig. 1. Example of Processing Time for Each Skid

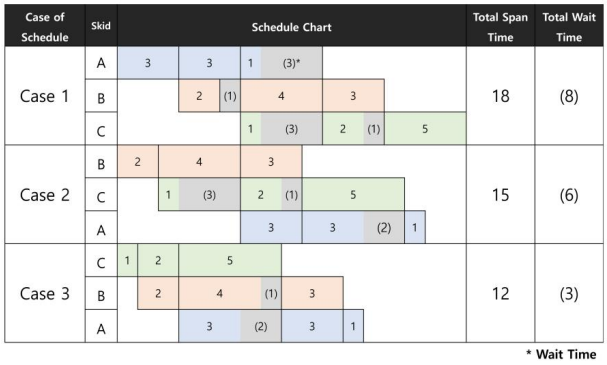

Fig. 2. Example of Span Time Change By Skid Ordering

만을 고려하는 것이 아닌, 그들 사이의 상호작용과 전체적인 흐름을 함께 고려해야 한다는 것을 알 수 있다.

IV. The Proposed Method

본 논문에서는 Fig. 3과 같은 혼합형 유전 알고리즘을 기반으로 소조립 공정순서 최적화 방법을 제안한다. 우선 초기 해 집단(population)을 랜덤하게 생성한다 (줄 1). 이후 매 세대마다 선택 연산(줄 3)을 통해 부모 해 p_1과 p_2를 선택한다. 이어서 교차 연산(줄 4)을 실행하여 새로운

```
Function HGA()
1  population ← Initialize()
2  for i =1 to generation
3     p1, p2 ← Select(population)
4     offspring ← Crossover(p1, p2)
5     offspring ← GUniDev(offspring)
6     if MAX(span(p1), span(offspring)) < span(p2):
7        population ← replace(offspring, p2)
8     elif MAX(span(p2), span(offspring)) < span(p1):
9        population ← replace(offspring, p1)
10 return best solution of population
```

Fig. 3. Pseudo Code of Hybrid Genetic Algorithm Used in This Study

자식 해(Offspring)를 생성하고 휴리스틱을 통해 지역 최적해로 개선시킨다 (줄 5). 만일 생성된 자식 해가 부모 해 중 품질이 낮은 해보다 더 좋은 경우 교체 연산(줄 6-9)을 실행한다. 끝으로 진화가 종료되면 해집단 중 최고해를 반환한다. 다음 각 절에서는 각각의 연산을 자세히 설명한다.

4.1 Representation

본 혼합형 유전 알고리즘에서 사용되는 해는 Fig. 4와 같이 일차원 배열로 표현된다. 이 배열은 [1, N]의 순열을 나타내는데, 각 원소는 투입될 스키드의 번호를 뜻한다. Fig. 4는 $N=7$인 경우의 예로서 3번, 5번, ..., 4번 스키드 순서로 스키드가 컴베이어에 투입되는 해를 의미한다.

Fig. 4. An Example of Encoding

4.2. Fitness Evaluation

적합도 평가는 생성된 자식 해의 적합도(Fitness)를 정량화하는 과정이다. 적합도 값은 GA의 선택, 교차 연산 등에서 사용된다. 본 연구에서는 3절에서 설명한 스키드 투입 순서 Π에 따른 전체 작업 시간 $span(\Pi)$에 반비례하는 값을 다음과 같이 적합도 값 $fitness(\Pi)$로 사용한다.

$$fitness(\Pi) = \frac{1}{span(\Pi)}.$$

4.3. Selection

선택은 다음 자식 해를 생성하기 위해 부모 해를 선택하는 과정이다. 높은 적합도를 갖는 부모 해를 선택하면 빠른 수렴을 보이지만, 지역 최적점에서 빠져나가지 못할 가능성이 있다. 반면 낮은 적합도의 부모 해를 선택하면 수렴을 하기까지의 시간이 오래 걸린다. 따라서 본 연구에서는 적합도 비례 확률 선택 연산자를 활용한다. 아래는 해집단 내 해 i의 선택 확률 $prob(i)$의 정의를 나타낸다.

$$prob(i) = \frac{fitness(i)}{\sum_{j=1}^{P} fitness(j)}$$

이 때 P는 해집단의 크기이다.

4.4. Crossover

교차는 선택된 부모 해들의 유전자를 결합하는 과정으로 이를 통해 새로운 자식 해(offspring)가 생성된다. 유전 알고리즘의 교차 연산자는 일반적으로 일점 교차, 다점 교차, 균등 교차 등이 있다 [10]. 본 연구에서는 부분 매칭

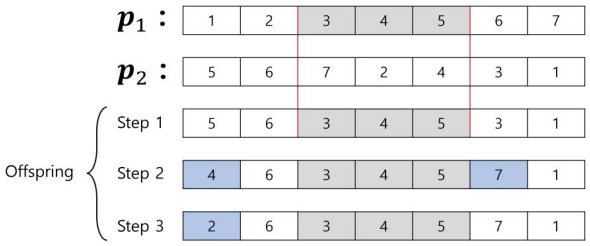

Fig. 5. An Example of Partially Matched Crossover

교차(Partially Matched Crossover, PMX)를 사용하였다. PMX는 주로 순열 기반의 최적화 문제에서 사용되는 교차 연산자로, 두 부모 해에 랜덤한 위치의 일정 부분을 서로 교환하는 방식이다. 이 교환 과정에서는 두 부모 해의 해당 부분이 매칭되어 있어야 하며, 나머지 부분은 상대 부모 해의 순서를 유지하면서 조정된다. 이를 통해 자식 해가 부모 해의 유전자 특성을 상속받으면서도 새로운 특성의 조합을 형성할 수 있다. PMX는 순열의 제약조건을 유지하면서 다양한 해를 탐색할 수 있다는 장점이 있다.

Fig. 5.는 본 연구에서 적용된 PMX의 예를 보인다. 그림에서 보듯이 먼저 교차될 부분을 결정하고, 분할된 영역을 두 부모해 p_1과 p_2를 번갈아가며 자식 해로 복사한다 (Offspring의 step1). 그런데 겹치는 숫자(유전자 값)가 있을 경우 그에 해당하는 위치의 다른 부모 해의 숫자로 교체한다. 만약 교체한 숫자가 여전히 중복된다면 교체과정을 반복한다. 예를 들면 offspring의 step1에서 1번째 위치의 '5'는 중복되므로 p_1의 '5' 위치를 찾아 대응하는 p_2의 숫자 '4'와 교체한다. 6번째 위치의 '3' 또한 중복되므로 p_1의 '3' 위치를 찾아 대응하는 p_2의 숫자 '7'로 교체한다 (offspring의 step 2). 마지막으로 여전히 1번 위치의 '4'가 중복되므로 p_1의 '4' 위치를 찾아 대응하는 p_2의 '2'와 변경해 준다 (offspring의 step 3). 따라서 최종 자식 해는 '2 6 3 4 5 7 1'이 되며 이는 순열의 조건을 만족한다.

4.5. Greedy Uniform Deviation-Based Heuristic

선행 연구[4]에서는 스키드의 투입 순서 최적화를 위해 UniDev라 불리우는 휴리스틱 탐색 방법을 제안하였다 (Fig. 6). 이 방법은 먼저 S_i^{π}(π의 i번째 스키드)의 j번째 공정과 동시에 컴베이어 위에 처리 중인 모든 공정의 작업 시간 평균 $m(S_{i,j}^{\pi})$과 이를 토대로 S_i^{π}의 j번째 공정에서 발생하는 절대 편차 $u(S_{i,j}^{\pi})$를 다음과 같이 정의하였다.

$$m(S_{i,j}^{\pi}) = \frac{1}{M}\sum_{k=1}^{M} T_{i+j-k,k}$$

1 **Function** UniDev(Π)
2 $bestSolution \leftarrow \Pi$
3 **for** e=1 **to** *epoch*
4 Randomly select $t \in \{1,\cdots,N+M-1\}$ with a probability proportional to $\frac{1}{B-A+1}\sum_{k=A}^{B} u(S^{\pi}_{t-k+1,k})$ where $A=\max(t-N+1,1)$ and $B=\min(t,M)$
5 **if** $t < M$
6 $m \leftarrow t$
7 **else if** $M \le t < N$
8 $m \leftarrow M$
9 **else**
10 $m \leftarrow N + M - t$
11 Randomly select $x \in \{t-m+1,\cdots,t\}$ with a probability proportional to $u(S^{\pi}_{i,t-i+1})$
12 Randomly select $y \in \{1,\cdots,N\}$ with a probability inversely proportional to $e(S^{\pi}_{x},S^{\pi}_{y})$
13 Swap the order of S^{π}_{x} and S^{π}_{y} in Π
14 **if** span(Π) < span(bestSolution)
15 $bestSolution \leftarrow \Pi$
16 **return** *bestSolution*

Fig. 6. Pseudo Code of UniDev heuristics [4]

$$u(S^{\pi}_{i,j}) = \left| T^{\pi}_{i,j} - m(S^{\pi}_{i,j}) \right|$$

또한 두 스키드 x와 스키드 y를 교환 후 발생하는 편차 합 $e(S^{\pi}_{x},S^{\pi}_{y})$을 다음과 같이 정의하였다.

$$e(S^{\pi}_{x},S^{\pi}_{y}) = \sum_{k=1}^{M} \left| T^{\pi}_{x,k} - m(S^{\pi}_{y,k}) \right| + \sum_{k=1}^{M} \left| T^{\pi}_{y,k} - m(S^{\pi}_{x,k}) \right|$$

UniDev는 매 반복마다 서로 투입 순서를 교체할 두 스키드 x와 y 쌍을 탐색한다. 구체적으로 먼저 전체 작업 스케줄 내에서 각 작업의 절대 편차에 비례하는 확률로 스키드 x를 선택한다 (Fig.6의 줄 11). 반면 스키드 y는 선택된 x와의 교환 후 발생하는 편차 합에 역비례하는 확률로 선택된다 (Fig.6의 줄 12). 초기 스키드 투입 순서 π를 기반으로 시작하며, 탐색 과정에서 전체 작업의 평균 절대 편차를 계산한다. 이후, 선택된 작업 시점에서의 절대 편차를 기반으로 스키드 x를 선택하고, 편차 합에 반비례하는 확률로 y스키드를 선택한다. 이렇게 선택된 두 스키드 x와 y의 순서 교환이 더 나은 성능을 보인다면 최선해로 갱신하며 이러한 교환탐색을 반복하여 해를 계속 개선시킨다. UniDev 알고리즘은 매 탐색마다 성능 개선을 보장

1 **Function** GUniDev(Π)
2 $bestSolution \leftarrow \Pi$
3 **for** e=1 **to** *epoch*
4 Randomly select $t \in \{1,\cdots,N+M-1\}$ with a probability proportional to $\frac{1}{B-A+1}\sum_{k=A}^{B} u(S^{\pi}_{t-k+1,k})$ where $A=\max(t-N+1,1)$ and $B=\min(t,M)$
5 **if** $t < M$
6 $m \leftarrow t$
7 **else if** $M \le t < N$
8 $m \leftarrow M$
9 **else**
10 $m \leftarrow N + M - t$
11 Randomly select $x \in \{t-m+1,\cdots,t\}$ with a probability proportional to $u(S^{\pi}_{i,t-i+1})$
12 $E \leftarrow \{1,2,3,\cdots,N+M-1\} \setminus \{x\}$
13 $MA \leftarrow [\,]$
14 **while** $E \neq \varnothing$:
15 Randomly select $y \in E$
16 $\Pi' \leftarrow bestSolution$
17 Swap the order of S^{π}_{x} and S^{π}_{y} in Π'
18 $\alpha = \frac{span(\Pi') - span(bestSolution)}{span(bestSolution)}$
19 $MA.append(\alpha)$
20 **if** $\alpha > 0$:
21 $bestSolution \leftarrow \Pi'$
22 break
23 **if** length(MA) > $thres_{L}$ **or** $average(MA[-W:]) < thres_{A}$:
24 break
25 $E \leftarrow E \,/\, \{y\}$
26 **return** *bestSolution*

Fig. 7. Pseudo Code of GUniDev

하지는 않지만, 확률적 선택 방법을 통해 다양한 해를 탐색하고 대기 시간 감소를 기대한다.

그러나 UniDev 알고리즘은 스키드 y를 선택하는 확률을 결정짓기 위해 모든 조합의 편차를 계산한다 (FIg.6 줄 12). 이는 알고리즘의 복잡도를 증가시켜 본 논문에서처럼 유전 알고리즘과 결합하여 사용하는 경우 지역 탐색 시간이 지나치게 증가하는 단점이 발생한다. 이러한 문제점을 해결하기 위해 본 논문에서는 기존 UniDev를 적절하게 변형한 GUniDev (Greedy Uniform Deviation-Based Heuristics)를 제안한다 (Fig. 7). 그 그림에서 보듯이 스

키드 x를 선택하는 과정(줄4-줄 11)까지는 UniDev와 동일하다. 반면 UniDev에서는 x이외의 모든 스키드에 대해 교환 후 성능을 계산한 반면 GUniDev에서는 그러한 계산 없이 임의의 순서로 교환 후 성능을 계산하고 성능 개선이 있으면 그것을 스키드 y로 선택하여 바로 교환한 결과를 반영한다 (줄 20-22). 즉 Fig. 6의 줄 12에서 $N-1$개 스키드에 대한 $e(S_x^\pi, S_y^\pi)$ 계산이 GUniDev에서는 필요하지 않아 매우 빠르게 스키들 y를 선택한다. 또한 Fig. 7의 줄 23-24에서 보듯이 스키드 y의 선택이 $thres_L$번 초과하여 성능 개선이 실패하거나 최근 W번의 시행 과정의 평균 성능개선비율이 $thres_A$보다 낮을 경우 스키드 x를 기준으로 한 교환 시도를 포기하고 새로운 x를 선택하여 시도한다. 이와 같이 기존 UniDev를 greedy 알고리즘의 형태로 변형하고 Early Stop 기법을 도입하여 y 선택 시도횟수를 제한함으로써 유전 알고리즘의 지역 탐색 연산자로 사용되기에 충분하도록 실행 시간을 단축하였다.

V. Experiments and Results

본 연구에서 제안하는 GUniDev기반 혼합형 유전알고리즘(이후 GA+GUniDev로 표기)의 성능을 검증하기 위해 Multi-Start방법, 기존 UniDev기반 혼합형 유전알고리즘(이후 GA+UniDev로 표기)와 비교하였다. 특히 탐색시도 횟수의 공정성을 유지하기 위해 Multi-Start 방법은 랜덤 초기해로부터 UniDev를 활용하여 지역최적해를 생성하는 횟수를 다른 혼합형 유전알고리즘의 세대수와 동일한 횟수로 시행하고 그 중에서 최고해를 반환한다. 실험은 스키드 개수(N = 50, 100, 200, 400)와 공정 개수(M = 6, 12, 24, 48, 96)를 바꿔가며 진행하였으며 (즉 총 20가지 종류의 문제), 유전 알고리즘의 세대수(GENERATION)는 3000, 해집단(POPULATION) 크기는 50으로 설정하였다. 각 실험은 총 15번 반복하여 진행하였다.

Fig. 8은 테스트한 세 가지 방법에 대한 실험 성능 결과이다. 그림에서 y축은 GA+UniDev와 GA+GUniDev가 Multi-Start에 비해 span time 성능을 개선시킨 평균 비율을 보이며 부록 Table 1은 자세한 수치 결과를 보인다. 그림에서 보듯이 본 연구에서 제안하는 GA+GUniDev는 MultiStart 와 GA+UniDev보다 모든 실험조건에서 좋은 성능을 보여준다. 반면 GA+UniDev는 Multi-Start에 비해 성능이 감소하는 경우가 많다. 이는 단순하게 휴리스틱을 유전알고리즘과 결합하는 경우 오히려 효과적으로 문제 공간을 탐색하지 않아 성능 향상이 어려울 수 있음을 암시한다. 또한 부록 Table 1에서 보듯이 세 가지 방법 모

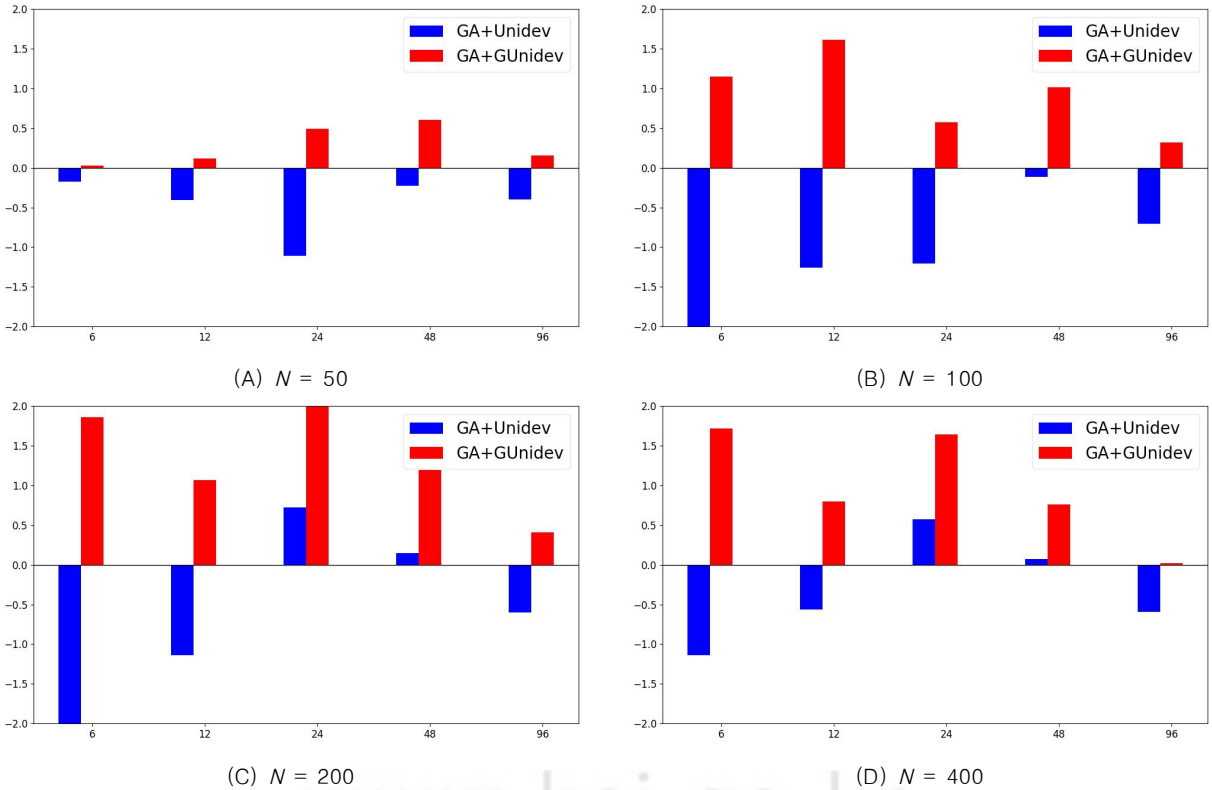

Fig. 8. Relative Performance Improvement Percentage Over Multi-Start With Respect To Span Time

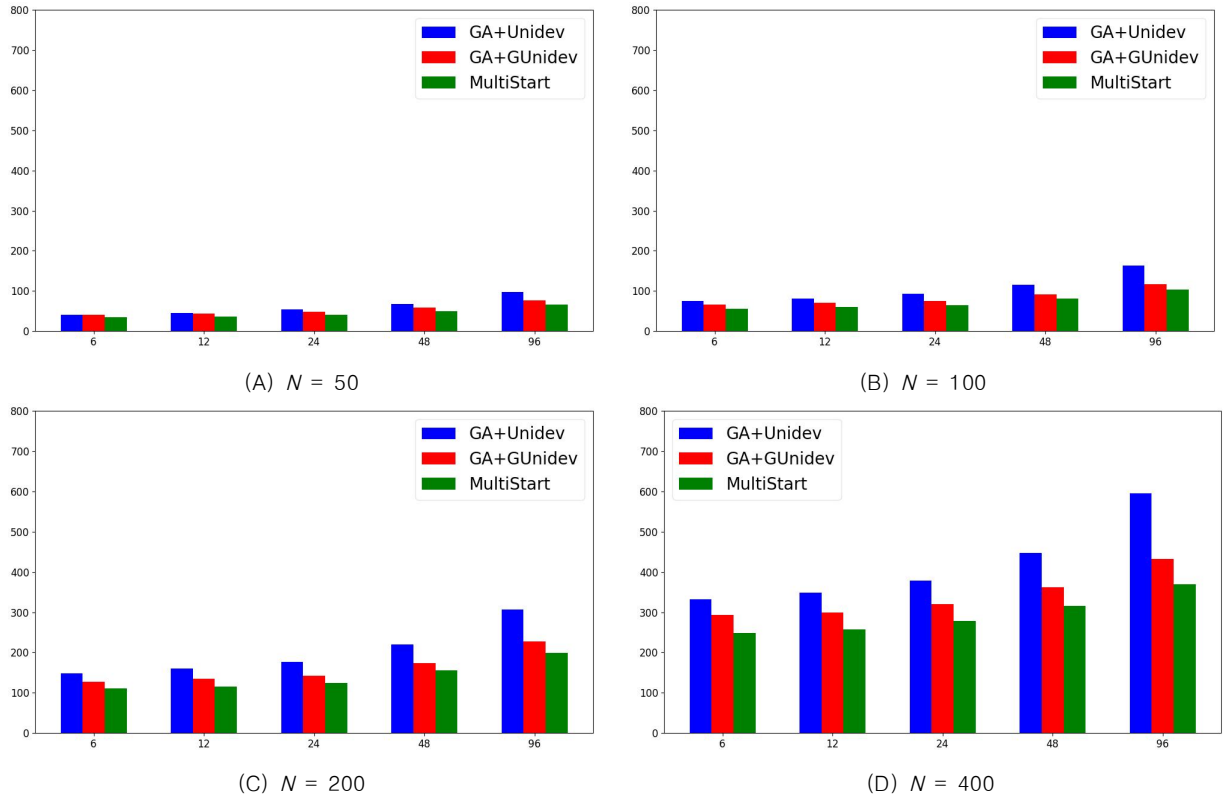

Fig. 9. Comparison of Running Time

두 스키드 개수(N)과 공정 개수(M)가 커짐에 따라 전체 작업 시간이 증가하는 현상을 확인할 수 있었다. 특히, M이 가장 큰 경우 Multi-start 대비 우리 방법의 성능 개선 비율이 가장 작았다. 이는 작업의 병목 현상이 두드러져서 컨베이어 위의 스키드 간 시간 공유가 증가하기 때문일 것으로 예상된다.

Fig. 9는 세 가지 방법의 평균 실행시간에 대한 결과이며 부록 Table 2는 구체적인 내용을 보인다. 그 그림에서 보듯이 N과 M의 증가에 따라 실행시간이 증가하였다. 이는 문제의 난이도가 증가함에 따라 문제공간 역시 증가하고 있음을 의미한다. 그리고 본 연구에서 제안하는 GA+GUniDev 방법은 Multi-Satrt보다는 느리지만 GA+UniDev보다는 빠르게 실행되었다. 이는 여러 진화 연산자들 소요 시간에 의해 Multi-Start보다 느려졌지만 기존 UniDev보다는 빠르게 지역최적화를 수행하고 있음을 의미한다. 따라서 본 논문에서 제안하는 혼합형 유전알고리즘은 GA+UniDev에 비해 안정적으로 실행속도가 빠르면서도 span time이 성능을 효과적으로 개선하였으며 이는 효율적으로 유전알고리즘과 휴리스틱을 결합했음을 의미한다.

VI. Conclusions

본 연구에서는 선박 소조립 공정의 스케줄링 문제에 대한 연구로 기존 휴리스틱을 변형한 GUniDev 휴리스틱과 유전 알고리즘을 결합한 혼합형 유전 알고리즘 방법을 제시하였다. 랜덤 탐색의 일종인 Multi-Start와 휴리스틱과 유전 알고리즘을 단순 결합한 방법과 비교를 통해 본 연구에서 제안하는 방식이 더 나은 성능을 안정적으로 보임을 확인하였다. 본 연구에서 제안하는 방법은 Multi-Start에 의한 탐색법에 비해 실행 시간이 조금 더 걸렸으나 해의 품질은 더 좋았고 단순 혼합형 유전알고리즘에 비해서는 실행 시간과 해의 품질 측면에서 모두 우수하였다. 특히 스키드 개수 96개, 공정 개수 400개에 이르기까지 매우 큰 규모의 문제에 대해 실험함으로써 실제 조선소에서 발생하는 문제들에 대해 적용될 수 있음을 확인하였다.

본 논문에서는 스키드 투입 순서에 제약이 없음을 가정하였으나 실제 조선소의 소조립 공정에서는 물류 관리의 한계로 인해 스키드 투입 가능 시간에 있어 제약이 있을 수 있다는 실험 방법의 개선 필요성이 있다. 또한 각 스키드의 공정에 따른 처리 시간을 랜덤하게 생성하였다는 한계점이 있다. 따라서 향후 실제 소조립 공정 데이터를 통

해 보다 정확한 시뮬레이션 문제를 생성하여 본 연구의 유효성을 검증할 필요가 있다.

ACKNOWLEDGEMENT

This work was supported by Institute of Information & Communication Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (2020-0-00869, Development of 5G-based Shipbuilding & Marine Smart Communication Platform and Convergence Service). This result was supported by "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(MOE)(2021RIS-003).

REFERENCES

[1] J. G. Bark, M. G. Kim “Optimization of Quantity Allocation using Integer Linear Programming in Shipbuilding Industry” JSNAK, Vol.57, No.1, pp.45-51, Jan, 2020. DOI: 10.3744/SNAK.2020.57.1.045

[2] I. H. Hwang, J. Y. Noh, K. K. Lee, J. Y. Shin “Short-term Scheduling Optimization for Subassembly Line in Ship Production Using Simulated Annealing” KSS, Vol. 19, No. 1, pp. 73-82, Mar, 2010. DOI: KSME-A.2005.29.2.262

[3] D. Bertsimas, J. Tsitsiklis "Simulated Annealing." Statist. Sci, Vol.8, No.1, pp. 10-15, Feb, 1993. DOI: 10.1214/ss/1177011077

[4] K. T. Lee, Y. K. Kwon, “A mean-absolute-deviation based method for optimizing skid sequence in shipyard subassembly” JKSCI , Vol. 27 No. 12, pp. 277-284, Dec, 2022. DOI: jksci.2022.27.12.277

[5] H. M. Kim, J. H. Kang, S. S. Park "Scheduling of shipyard block assembly process using constraint satisfaction problem." APMR, Vol. 7, No. 1, pp. 119-137, Oct 2001.

[6] H. R. Choi, K. R. Ryu, K. K. Cho, H. S. Lim, J. H. Hwang “A Scheduling System for Panel Block Assembly Shop in Shipbuilding using Genetic Algorithms” KIISS, Vol.2, No.2, pp.29-42, Dec, 1996.

[7] T. R. Ha, C. U. Moon, C. M. Joo, J. C. Park “A Hybrid Genetic Algorithm for Scheduling of the Panel Block Assembly Shop in Shipbuilding” KORMS, Vol. 17, No. 1, pp. 135-143, May, 2000.

[8] H. J. Kang, J. Y. Park, H. C. Park "Work planning using genetic algorithm and 3D simulation at a subassembly line of shipyard" KWJS, Vol.1, pp. 18-20, May, 2004.

[9] H. C. Bae, K. C. Park, B. C. Cha, I. K. Moon “Scheduling of Shipyard Sub-assembly Process using Genetic Algorithms” KIIE, Vol. 20, No. 1, pp. 33-40, Mar, 2007.

[10] A. J. Umbarkar1, P. D. Sheth "Crossover operators in genetic algorithms: a review." ICTACT, Vol. 6, No. 1, pp. 1083 - 1092, Oct, 2015. DOI: 10.21917/ijsc.2015.0150

Authors

Min-Jae Choi is currently a B.S student at Ulsan University in South Korea, and he is expected to graduate in 2024. Currently, Min-Jae Choi is actively engaged as a student in the Department of Computer Science. He is currently a B.S student in School of IT Convergence, University of Ulsan. He is interested in genetic algorithm and optimization.

Yung-Keun Kwon received the B.S., M.S. and Ph.D. degrees in Computer Science and Engineering from Seoul National University, Korea, in 1999, 2001 and 2006, respectively. He is currently a Professor in Department of Electrical, Electronic and Computer Engineering, University of Ulsan. He is interested in genetic algorithm and optimization.

Appendix Table 1. Detailed Span Time Results

(N, M)	Multi-start	GA + UniDev	GA + GUniDev
(50, 6)	12223.09	12244.58	12219.73
(50, 12)	16108.77	16174.09	16089.91
(50, 24)	17491.83	17687.98	17406.24
(50, 48)	25828.97	25887.65	25674.15
(50, 96)	40700.67	40862.34	40638.78
(100, 6)	23439.79	23946.50	23173.18
(100, 12)	27969.44	28326.37	27526.29
(100, 24)	32038.11	32430.07	31856.60
(100, 48)	40035.82	40082.66	39635.18
(100, 96)	54926.80	55316.75	54751.42
(200, 6)	48773.55	49893.88	47882.78
(200, 12)	55545.65	56183.48	54958.42
(200, 24)	62079.02	61633.41	60787.78
(200, 48)	69714.16	69613.90	68889.92
(200, 96)	84585.37	85097.74	84243.80
(400, 6)	97498.94	98623.72	95853.46
(400, 12)	111932.19	112564.56	111048.75
(400, 24)	118776.99	118098.34	116861.24
(400, 48)	128491.24	128397.31	127525.79
(400, 96)	144423.83	145288.13	144390.73

Appendix Table 2. Detailed Running Time Results

(N, M)	Multi-start	GA + UniDev	GA + GUniDev
(50, 6)	34.36	41.42	40.59
(50, 12)	36.66	45.22	43.09
(50, 24)	40.99	53.53	48.16
(50, 48)	49.90	68.15	58.22
(50, 96)	66.52	96.98	76.29
(100, 6)	56.36	74.64	65.60
(100, 12)	60.11	81.14	70.59
(100, 24)	65.39	92.96	75.23
(100, 48)	81.09	115.62	90.89
(100, 96)	103.97	163.68	117.08
(200, 6)	111.52	149.06	128.10
(200, 12)	115.90	160.85	135.35
(200, 24)	124.99	177.31	141.71
(200, 48)	155.55	220.21	173.28
(200, 96)	199.72	306.41	226.93
(400, 6)	249.07	331.64	293.16
(400, 12)	257.49	347.98	299.70
(400, 24)	277.72	378.83	320.11
(400, 48)	315.67	447.12	362.27
(400, 96)	368,93	595.13	432.04