이열 배치설계를 위한 Cuckoo Search 기반의 최적화 알고리즘 개발

윤영민[1], 채준재[1,†]

[1]한국항공대학교 항공교통물류학부

Development of Cuckoo Search Based Optimization Approach for a Double Row Layout Problem

Yeongmin Yun[1], junjae Chae[1†]

[1]School of Air Transport, Transportation and Logistics, Korea Aerospace University

Effective layout design can improve productivity and manufacturing efficiency. Given a number of departments with known lengths and the flow cost between any two departments, a double row layout problem(DRLP) determines an arrangement of departments on either side of a central corridor so that the total material moved flow is minimized when the material handling path is the straight line along the corridor. DRLP arises in many other real-world applications including manufacturing system. This paper presents a approach in cuckoo search algorithm(CS) for obtaining good solutions for the DRLP. Random-key encoding scheme is applied to CS for representing a solution of DRLP and enabling Lévy flights to perform on DRLP. To evaluate the approach, computational experiments are conducted from benchmark problems. the obtained results show the better performance of proposed approach then a previous study.

Keywords: Facility layout, Double row layout problem, Meta-heuristic, Cuckoo Search

1. 서론

배치설계는 물류센터 나 생산시설에서 개체들의 상대적인 위치를 최적으로 배치하는 것을 목적으로 하는 전형적인 조합 최적화 문제로서, 문제의 크기가 커질수록 최적 값을 구하는데 걸리는 시간이 기하급수적으로 증가하는 NP-hard 문제이다(Garey and Johnson, 1979). 여기서 개체는 물리적인 형태의 설비뿐만 아니라 물류시설의 보관공간이나 제조시설의 생산공간과 같은 무형의 서비스 공간을 포함한 개념이다(Chae and Regan, 2020). 많은 연구 들에서 개체의 기본적인 형태는 직사각형의 형태로 가정한다(Chae and Regan, 2016; Kang and Chae, 2017). 또한, 배치설계에서 개체들은 기본적으로 주어진 공간안에서 겹쳐지지 않게 배치되어야 한다(Ingole and Singh, 2017). 배치설계에서 최적에 대한 기준은 다양한데 각 개체들 간의 자원의 흐름, 즉 물동량을 고려한 개체간 물류 이동 거리의 총합을 최소화하는 것을 일반적으로 많이 사용한다. 본 연구에서는 지금까지의 연구 들에서 일반적으로 가정한 것처럼 개체를 직사각형의 형태로 가정하였으며, 배치에 대한 최적의 기준 역시 총 물류 이동 거리의 합으로 가정하였다.

일반적으로 시설계획(Facility planning)에 있어서 효율적인 배치설계는 매우 중요하다(Liu et al., 2020). 물류센터나 생산시설 내에서 설비나 공간에 대한 배치형태는 작업자나 AGV(Automated Guided Vehicle)의 작업 동선과 같은 운영 효율성과 연관된 요소들에 많은 영향을 준다(Tompkins et al., 2010). 또한 제조

과정에서 생산비용의 15~70%은 개체의 배치형태와 관련되어 있으며, 효율적인 배치설계는 시설내 생산작업 운영비용을 10~30%까지 줄일 수 있다(Braglia et al., 2003).

현대에 와서 변화되고 있는 시장환경을 보면, 제품의 종류가 다양해지고 있어 각 제품의 수요량을 예측하기가 점점 더 어려워지고 있으며 제품의 수명 주기 또한 짧아지고 있다. 이러한 흐름에 따라 생산제조시설에서의 한 가지 제품라인은 더 이상 효율성이 없고 설비나 장비의 교체 없이 설비나 제조공간의 재배치를 통한 한 제품라인에서 다른 제품 라인으로의 빠른 변화가 요구되고 있다(Benjaafar and Sheikhzadeh, 2000). 배치설계가 유동적으로 변경 가능해지면 다양해지는 제품의 종류와 짧아지는 제품의 수명에 따라 배치를 효율적으로 바꿀 수 있어 생산성과 제조 효율성을 증가시킬 수 있다.

배치설계에 관련된 연구는 예전부터 오래도록 연구가 진행되어 왔지만 최근 배치설계의 효율화가 중요해진 시점에서 더욱 필요한 연구라고 할 수 있으며, 오랜 시간 들여 한 번 좋은 배치를 찾으면 끝이던 과거와는 달리, 갈수록 제품의 수명주기가 짧아지고 있고 제품의 종류가 다양해지고 있어 빠른 시간안에 효율적인 배치를 찾을 수 있는 연구가 필요한 상황이다.

여러 배치설계 유형 중 '열' 배치설계(row layout problems)는 개체를 줄지어 평행하게 배치하는 유형이다. 그리고 개체를 두 줄로 평행하게 배치할 때 해당 배치형태를 이열 배치라고 한다(Ahonen et al., 2014; Amaral, 2012a, 2013a, 2020; S. Wang et al., 2015; Zuo et al., 2014, 2016). 이열 배치설계는 배치해야 할 개체들의 길이(length)와 각 개체들 간의 물동량(flow)이 주어졌을 때, 개체들 간 물류 이동 거리의 총합을 최소화하도록 하는 개체들의 이열 배치를 찾는 문제이다. 이열 배치설계는 대표적으로 AGV와 같은, 개체들 사이에서 자원을 운반하는 운송체가 존재하는 제조 유연 생산 시스템내 개체들에 대한 배치설계에 주로 적용된다(Heragu and Kusiak, 1988; Tubaileh and Siam, 2017). 이때, <Figure 1>에서 볼 수 있듯이 개체들은 AGV가 작업을 위해 움직이는 경로 및 동선의 위쪽 또는 아래쪽에 배치되며, 설비내 자원의 입출력 지점은 각 설비의 AGV 동선 방향 쪽 중간 지점으로 설정한다. 중앙의 AGV의 경로는 일반적으로 작업 동선의 효율성을 높이기 위해 일직선의 형태로 설정된다. 그리고 이열 배치설계는 FMS 내에서 뿐만 아니라, LCD 생산라인(Chung and Tanchoco, 2010), 오피스나 병원과 같은 다양한 건물 내의 공간에 대한 배치(Ahonen et al., 2014), 반도체 제조공정내 설비배치(S.Wang et al., 2015; Zuo et al., 2014) 등 다양한 실제 업무 환경에서 많이 적용되고 있다.

이와 같이 이열 배치설계는 실제 업무 환경에서 많이 활용되고 있는 문제이며 기본적인 배치형태이기 때문에 다른 배치유형으로의 확장이 용이할 것으로 판단하여 본 연구에서는 이열 배치설계에 대해 다루고자 하였다.

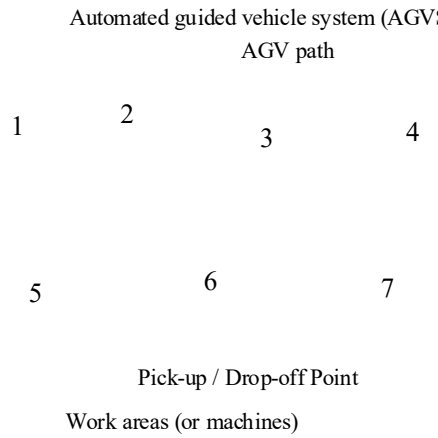


Figure 1. Double row layout problem

하지만 이열 배치설계 역시 배치설계의 한 종류이므로 NP-Hard 조합최적화 문제이다(Amaral, 2012a, 2013b; Secchin and Amaral, 2019). 그렇기 때문에 배치해야 할 개체의 수가 늘어날수록 문제를 풀기위한

계산시간은 기하급수적으로 증가한다(Chae and Regan, 2020; Secchin and Amaral, 2019). 따라서 개체 수가 많아질수록 분단탐색법과 같은 정확 기법을 통해 문제를 해결하기에는 시간적인 한계가 존재한다. 그렇기 때문에 많은 연구들에서 일반적으로 규모가 큰 문제들을 빠른 시간 내에 해결하기 위해 다양한 휴리스틱 알고리즘과 메타 휴리스틱 방법론이 개발되고 적용되어 왔다.

본 연구에서는 이열 배치설계에 대한 해결 방법론으로 메타 휴리스틱 알고리즘으로는 비교적 최근에 고안된 Cuckoo Search(CS) 알고리즘을 적용하였다. CS 알고리즘은 일부 뻐꾸기 종의 특이한 번식행위를 모방하여 만들어졌다(X. S. Yang and Deb, 2009). 몇몇의 뻐꾸기 종들은 다른 새의 둥지에 알을 낳아 새끼를 부화시키는 "탁란"이라는 행위를 통해 번식하는데 만약 둥지의 주인이 이를 알아채면, 뻐꾸기의 알을 버리거나 자신의 둥지를 버리고 다른 곳으로 이동해 새로 둥지를 만든다. 그렇게 되면 뻐꾸기의 알은 버려져 생존율이 떨어지기 때문에 주인 새의 알과 색깔 및 모양을 비슷하게 만들어 알이 버려지는 확률을 낮춰 생존율을 높인다. CS 알고리즘은 그동안 많은 연구들에서 다양한 최적화 문제들에 적용되었으며 다른 최적화 알고리즘들과 비교 시에도 CS 알고리즘이 매우 효율적이라는 것을 보여주었다(Civicioglu and Besdok, 2013; Gandomi et al., 2012, 2013; Srivastava et al., 2012; Walton et al., 2011; X. S. Yang and Deb, 2010, 2014). 하지만 CS 알고리즘은 해를 탐색하는 영역 즉, 가능해(feasible solutions)의 영역이 연속적인 연속최적화 문제에 맞춰 고안된 메타 휴리스틱 중 하나인데, 배치설계와 같은 조합 최적화 문제는 가능해의 영역이 이산 집합에 속하거나 이산적(discrete)인 이산 최적화 문제이기 때문에 CS 알고리즘과 같은 메타 휴리스틱을 조합 최적화 문제에 적용하게 되면 문제 복잡성이 증가하는 문제가 발생한다(Kang et al., 2018). 이와 같은 문제로 인해 CS 알고리즘을 배치설계를 포함한 조합 최적화 문제에 적용한 연구는 지금까지 많지 않았다.

CS 같은 메타 휴리스틱을 배치설계와 같은 조합 최적화 문제에 적용하기 위해서는 메타 휴리스틱의 좋은 성능은 잃지 않으면서 적절하게 조합 최적화문제에 적용시키는 것이 중요하다(Ouaarab et al., 2015b). 특히, CS 알고리즘은 Lévy flight라는 랜덤 워크 모델을 통해 새로운 해를 탐색하는데, Lévy flight는 결과값으로 특정 실수범위의 상수 값을 도출하기 때문에 Lévy flight상수 값으로 이열 배치설계의 새로운 해를 이산적으로 탐색하기 위한 방법이 필요하다. 따라서 본 연구에서는 이열 배치설계의해를 표현하고 Lévy flight 상수 값의 적용 방법으로 Random-key 방식을 사용했다.

본 논문의 다음 장 구성은 다음과 같다. 2장에서는 이열 배치설계와 Cuckoo search(CS)알고리즘에 대한 기존 연구에 대해 정리하고, 3장에서는 본 논문에서 다루고자 하는 이열 배치설계의 수리모형에 대해, 그리고 4장에서는 물류 비용이 최소화되도록 하는 좋은 이열 배치형태를 찾기 위해 본 연구에서 적용한 방법론에 대해 설명한다. 5장에서는 본 연구에서 제안한 방법론의 적용에 대한 실험 결과를 보이며, 이전 연구 결과와의 비교를 통해 분석한 내용을 담았다. 마지막 6장에서는 본 논문에 대한 결론과 함께 향후의 연구과제를 제시하였다.

2. 기존 문헌 연구

배치설계와 관련된 연구들은 지난 수십년 전부터 지금까지 많은 연구가 진행되었고, 배치설계를 해결하기 위한 모델과 방법론 또한 다양하게 제시되어 왔으며 이에 대해 다음과 같은 많은 연구 들에서 정리되어졌다(Drira et al., 2007; Heragu and Kusiak, 1988; Levary and Kalchik, 1985; Meller and Gau, 1996; Singh and Sharma, 2006). 그리고 다양한 방법론들은 크게 분단탐색법(Branch and bound method)과 같은 정확 기법(Exact methods)과 휴리스틱 접근법이나 메타 휴리스틱과 같은 근사적 기법(Approximated approaches)으로 분류될 수 있다(Drira et al., 2007). 하지만 배치설계는 서론에서 설명한 것과 같이, NP-hard 문제라 문제의 규모가 커질수록 최적 값을 찾기 어렵기 때문에 정확 기법으로 해결이 가능한 작은 규모의 문제들에 대해서만 정확 기법을 적용한 연구들이 진행되어져 왔으며(Chae and Regan, 2020), 배치설계와 관련된 대부분의 연구들은 근사적 기법을 적용해왔다(Chae and Regan, 2020; Garcia-Hernandez et al., 2019; Kang and Chae, 2017; Kim and Chae, 2019; Scalia et al., 2019).

또한, 배치설계는 매우 다양한 기준으로 분류될 수 있는데 개체들이 배치되는 형태나 방식(Layout configuration)에 따라서도 다양한 유형들이 존재한다. Drira et al.,(2007)에 따르면, 배치설계는 배치 형태에 따라 일렬 배치(single-row), 다열 배치(multi-row), 루프 배치(loop), 오픈필드(open field), 다층 배치(multi-floor)로 분류된다. 본 연구에서 다루는 이열 배치형태(double-row)는 제시된 분류에 속하지는 않지만 복도의 한쪽에만 개체를 배치하는 일렬 배치 형태와 달리, 중앙 복도의 양쪽으로 배치하므로 복도의 한쪽에만 개체를 배치하는 형태에서 복도의 양쪽에 배치한다는 의미에서 이열 배치설계를 일렬 배치설계의 확장된 문제 형태라고 볼 수 있다(Chae and Regan, 2020).

이열 배치설계(DRLP)는 서론에서 밝힌 것과 같이 실제 업무 환경에서 많이 활용되고 있는 문제지만 이전부터 연구가 많이 진행되어왔던 일렬 배치설계와 달리, 이열 배치설계는 비교적 최근부터 연구가 진행되기 시작했으며, 일렬 배치설계에 비해 연구에 대한 관심이 제한적이었다(Guan et al., 2020). 그렇기 때문에 이열 배치설계 관련 연구들에 대해서는 정확 기법과 근사 기법으로 구분하지 않고 각 연구들에 대해 조금 더 자세히 살펴보았다.

Chung and Tanchoco(2010)은 Heragu and Kusiak(1991)가 제시하고 Amaral(2006)의 연구에서 개선시킨 SRLP에 대한 MIP 수리 모형을 확장 시켜 처음으로 DRLP에 대한 MIP 수리 모형을 제시하였고, 제시된 수리 모형을 통해 정의된 DRLP를 풀기 위해 5개의 휴리스틱 알고리즘을 고안하고 적용하였다. Zhang and Murray(2012)은 Chung and Tanchoco(2010)의 MIP 수리 모형이 최소 이격 간격에 대한 제한조건이 없어 최소 이격 간격이 없는 배치형태가 허용되는 문제점이 있다고 지적하면서 Chung and Tanchoco(2010)의 MIP 모형의 수리 식을 수정하였다. Murray et al.(2013)는 이동하는 방향에 따라 두 설비 간의 이동횟수가 서로 다른 설비 간의 비 대칭적인 자원의 흐름을 고려하도록 Zhang and Murray(2012)가 수정한 수리 모형을 확장 시켰다. 그리고 Chung and Tanchoco(2010)가 제시한 휴리스틱 알고리즘에 간단한 주변 탐색과정(local search)을 추가하여 개선된 휴리스틱 방법론을 제시하였다. Zhang and Cheng(2014)은 DRLP를 조합 최적화 문제와 선형계획법(LP)문제로 구분해 조합 최적화 문제를 풀기위한 3가지의 휴리스틱 방법론과 선형계획법을 위한 프로그램으로 CPLEX를 사용하였다. Anjos et al.(2016)은 모든 설비들에 대해 인접한 설비 간의 거리가 모두 같은 DRLP를 풀기 위해 Zhang and Cheng(2014)처럼 DRLP를 2단계로 구분해 정수계획법(Integer linear programming)과 semidefinite programming(SDP)을 사용하는 방법론을 제시하였다. Amaral(2013b), Secchin and Amaral(2019), Amaral(2019), 그리고 Chae and Regan(2020)은 개체 간의 수평, 수직적 위치에 관한 상대적인 위치관계를 나타낼 수 있는 하나의 이진변수를 통해 DRLP 문제를 새로운 MIP로 표현하고 이후 순차적으로 발전시킨 논문들이다. 따라서 최근의 MIP 모델(Chae and Regan, 2020)은 Amaral(2013b)의 모델보다 확연히 효율적으로 DRLP 문제를 풀수 있다.

최근 DRLP에 대해 휴리스틱 방법론을 적용하는 연구들이 진행되었다. 방법론은 이전의 연구들처럼 DRLP를 개체들의 상대적인 위치를 찾아내는 조합 최적화 문제에 대한 부분과 개체 간의 상대적인 위치에서의 절대적인 위치를 찾는 해의 범위가 연속적인 문제로 구분하여 제시되었다. Amaral(2020)은 개체들의 상대적인 위치와 절대적 위치를 모두 휴리스틱으로 결정하는 방법론을 먼저 제시하고 더 좋은 배치를 찾기 위해 상대적인 위치만 휴리스틱으로 결정하고 절대적 위치를 LP로 해결하는 방법론을 제시하여 LP를 적용했을 때의 개선효과를 확인하였다. Guan et al.(2020) 역시 개체들의 상대적인 위치를 찾는 방법론과 절대적 위치를 찾는 방법론을 구분하여 제시했는데 상대적인 위치를 결정하는 방법론은 이전의 연구들처럼 휴리스틱 알고리즘을 사용하였지만 절대적인 위치를 결정하는 방법론은 이전 연구들과 달리, LP 사용 시 큰 규모의 문제에 대해 값을 도출하는데 너무 많이 걸린다는 단점을 해결하기 위해 LP와 같은 정확기법을 사용하지 않고 메타 휴리스틱인 PSO(Particle Swarm Optimization)를 사용하였다. 해당 연구는 처음으로 개체 수가 80개인 문제까지 제시하는 방법론을 적용하여 실험하였고, 실험 결과에서 이전 연구들보다 7개 문제들에 대해 더 좋은 값을 찾아냈다.

배치 설계문제 유형 중에서 DRLP와 연관되어 DRLP와 유사한 문제가 있는데 복도 할당 문제(corridor allocation problem; CAP)와 평행 열 배치 문제(parallel row ordering problem; PROP)이다. CAP(Ahonen et al., 2014; Amaral, 2012b)와 PROP(Amaral, 2013a; X. Yang et al., 2020)는 DRLP와 같이 이열(double

row)을 따라 개체들을 배치하지만 각 열의 가장 왼쪽에 위치하는 첫 개체가 미리 할당된다는 점에서 DRLP와 다르다고 할 수 있다. 그리고 CAP와 PROP는 두 인접한 개체 사이에 공간이 없다고 가정한다. 심지어 PROP는 각 개체들이 특정 열에 미리 할당되는 문제로 DRLP에 비해 더 제한적인 문제라고 할 수 있다. 이 두 유형은 주로 사무실로 사용되는 건물이나 학교, 병원 등의 공간에 대한 배치에 적용된다.

앞서 서론에서 언급한 것처럼, CS 알고리즘은 해의 범위가 연속적인 최적화 문제들에 쉽고 간단하게 적용이 가능하며 알고리즘이 최종 결과 값을 도출하는데 걸리는 시간 역시 다른 메타 휴리스틱 방법론들과 비교했을 때 상대적으로 빨랐다. CS 알고리즘의 적용과 성능에 대한 전반적인 정리 및 요약은 X.S. Yang and Deb(2014)가 다루었다.

CS 알고리즘은 해를 탐색하는 범위가 연속적인 최적화 문제에 맞춰 고안된 알고리즘으로 새로운 해를 탐색하는 방법으로 Lévy flight 라는 랜덤 워크 모델을 사용하기 때문에 CS 알고리즘을 조합 최적화 문제에 적용하기가 어렵다고 앞에서 설명하였다. 실수 값의 형태로 도출되는 Lévy flight 값을 정수 값이나 이진 값(1또는 0)으로 이루어진 배열의 형태로 표현되는 이산적인 해(discrete solution)에 적용하기 위해서는 일종의 변형이 필요하다. Burnwal and Deb(2013)은 FMS의 일정 최적화(scheduling optimization)를 위해 CS 알고리즘 기반의 접근법을 제시하였는데 Lévy flight 값에 의한 해 탐색의 방법으로 Lévy flight 값을 발생시켜 그 값에 가장 가까운 정수를 찾고 해당 정수 값을 정수로 이루어진 배열로 표현된 해에 더해줌으로써 해에 변형이 일어나게 하였다. 해당 연구에서 다루는 flow shop scheduling 문제에서는 작업(job)에 대한 순서정보를 담은 배열이 해(solution)가 되는데 Lévy flight 값에 의해 순서에 해당되는 정수 값을 임의로 바꿔 줌으로써 변형이 일어난다. Li and Yin(2013)와 Ouaarab et al.(2015a)은 앞에서 설명한 flow show scheduling 문제에 CS 알고리즘을 적용하였는데 Li and Yin(2013)은 Lévy flight 를 사용하는 방법으로 본 연구에서 적용한 Random-key 방법을 적용하였다. 하지만 본 연구에서 적용한 Random-key 방법과는 다르게, 1이 넘어가는 값에 대해 제한을 두지 않고 사용하였다. Ouaarab et al.(2015a)은 Lévy flight 값에 따라 구간을 설정하여 해당 구간에 따라 다른 주변 탐색 기법(local search)을 적용하여 해를 탐색하였다. Ain and Bey(2012)은 해가 0과 1의 이진 값으로 이루어진 배열로 표현된 배낭문제(knapsack problem)를 풀기 위해 기존의 CS 알고리즘을 변형시켜 binary cuckoo search(BCS)알고리즘을 고안하여 적용하였다. Ouaarab et al.(2014a, 2014b)은 외판원 문제 (traveling salesman problem; TSP)를 해결하기 위해 CS 알고리즘을 적용하였다. 그리고 Quaarab et al.(2015b)는 Bean(1994)이 처음 소개한 Random-key 방법을 TSP에 적용하여 소개하였다. Random-key 방식은 Lévy flight 값을 적용하기 위해 앞서 다른 연구 들에서 사용한 것처럼 Lévy flight 값을 발생시키고 나서 다시 2-opt와 같은 방식을 통해 적용해줄 필요없이 곧바로 조합 최적화 문제의 해에 적용 가능하다. 이러한 이점 때문에 이후 연구들에서는 CS를 조합 최적화 문제에 적용할 때 Random-key 방법이 주로 적용되어졌다. Kang et al.(2018)은 본 연구에서 다루는 배치설계에 Random-key 방법을 적용한 CS 알고리즘을 사용하였다. 해당 연구에서는 Drira et al.(2007)가 배치형태에 따라 분류한 유형 중 하나인 루프(loop) 형태로 개체를 배치하는 문제를 다루었으며 이열 배치설계는 개체의 길이가 정해져 있는 것과 달리 개체의 길이가 정해져 있지 않고 배치공간에 따라 유동적으로 바뀌는 개체에 대해 다루었다.

지금까지 살펴본 본 연구에 대한 기존 문헌들을 통해 알 수 있었던 것은 다음과 같다. CS 알고리즘은 연속 최적화 문제에서의 성능이 검증되어 최근 조합 최적화 문제에 적용해보는 연구가 늘어나고 있지만 아직까지 본 연구에서 다루는 이열 배치설계에 CS 알고리즘을 적용하여 진행된 연구는 없었다. 따라서 본 연구에서는 이열 배치설계(DRLP)에 대한 해결 방법론으로 메타 휴리스틱 방법인 CS 알고리즘을 적용하였으며, 여러 연구들에서 성능이 검증된 Random-key 방법으로 Lévy flight 값을 적용해주었다.

3. 이열 배치설계 문제 정의(Problem description)

이 장에서는 이열 배치설계 문제(DRLP)에 대해 자세히 설명하고 이열 배치설계의 수리 식 모형을 제시하고 설명해 본다. 본 연구에서는 서로 다른 길이의 개체들을 이열(Double row)로 배치하는 문제를 다루고

있다. 이열 배치설계의 기본적인 형태는 최적 값을 찾기 위한 혼합정수계획법(Mixed Integer Programming; MIP)모델로 수리 식을 표현할 수 있다(Chung and Tanchoco, 2010). 그리고 앞 장에서 설명한 것처럼 이열 배치설계에 대해 여러 연구들에서 문제 해결 시간을 단축시키고 효율성을 높이기 위해 MIP 모델에 제한조건을 추가하는 등의 변형을 주었다(Amaral, 2013b, 2019; Chae and Regan, 2020; Secchin and Amaral, 2019; Zhang and Murray, 2012). 하지만 본 연구에서는 이해를 위해 기본적인 개념적 모델을 소개한다.

이열 배치설계는 <Figure 2>에서 보여 지는 것처럼 시설 내에 중앙 복도가 있다고 가정하며 중앙 복도의 너비는 문제에서 고려하지 않는다. 각 개체들은 중앙 복도의 위쪽과 아래쪽에 배치된다. 각 개체들의 형태는 직사각형으로 가정되며, 개체의 길이와 개체 간의 자원에 대한 흐름(물동량)이 주어진다. 자원들이 나가거나 들어오는 개체의 입출력 지점은 중앙 복도 방향 중간 지점이라고 가정한다. 이 배치설계의 목적은 물류비용이 최소화되는 배치를 찾아내는 것으로, 이때의 물류비용은 각 개체 간의 입출력 지점 사이의 거리와 물동량의 곱의 총합으로 결정된다. 여기서의 거리는 중앙 복도에 다니는 작업자나 AGV의 작업동선을 최소화하기 위해 최단거리를 가정하는데 복도의 너비가 고려되지 않기 때문에 직선거리가 사용된다.

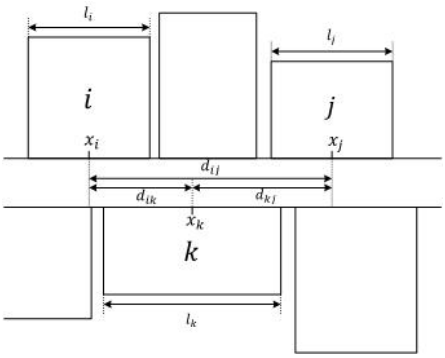


Figure 2. Representation of DRLP

또한, <Figure 2>에서 볼 수 있는 것처럼 x_i는 개체 i의 중간 지점의 좌표를 의미하며, l_i는 개체 i의 길이를 나타낸다. 그리고 개체 i와 j개체 의 거리를 d_{ij}로 표현한 것을 확인할 수 있다. 모든 개체들의 길이 합은 L로 나타냈으며, 개체 와 개체 사이의 물류 흐름 량인 물동량은 f_{ij}로 표현하였다. 본 연구에서 사용한 이열 배치설계 시에 주어지는 상수 값인 파라미터(Parameters)와 개체의 배치에 따라 변하는 값인 변수(Variables)는 각각 <Table 1>, <Table 2>에 요약하여 정리하였다.

Table 1. Parameters of the double row layout problems

n	The number of departments
f_{ij}	Amount of flow between department i and j
l_i	Length of department
L	Sum of total departments' length(L= l_1+l_2+l_3+... +l_n)

Table 2. Variables of the double row layout problems

x_i	Abscissa of the center or P/D point of department
d_{ij}	Distance between department i and j
α_{ij}	Horizontal and vertical relative location with a combination of variables (1, if department i is to the left of department j and 0, otherwise)

원래 배치설계 시에는 <Figure 2>에서처럼 모든 개체들에 대해서 근접한 두 개체 사이에 서로 다른 최소한의 이격 간격(clearance)가 존재한다. 이러한 경우, 이열 배치설계를 위해서 주어지는 파라미터는 <Table 1>에서 설명한 파라미터를 포함해서 개체 i와 개체 j사이에 필요한 이격 간격(clearance)이 추가되는데 (Heragu and Kusiak, 1988)실제 문제에서는 최소 이격 간격은 주로 유지보수(maintenance)를 위해 사용되기 때문에 개체의 배치순서가 아닌, 개체에 따라 달라지며 이런 경우에는 이격 간격을 개체의 길이에 포함하여 생각할 수 있다(Chung and Tanchoco, 2010). 따라서 본 연구에서는 가장 간단한 이열 배치설계 모형에 대해 설명하기 위해 이격간격이 없는 유형의 모형에 대해 설명하고 수리 모형의 목적함수와 제한조건들은 다음과 같다.

$$\text{Minimize} \quad \sum_{(i,j);i<j} c_{ij}d_{ij} \tag{1}$$

$$\text{Subject to} \quad d_{ij} \geq x_i - x_j, \quad \forall i,j(i<j) \tag{2}$$

$$d_{ij} \geq x_j - x_i, \quad \forall i,j(i<j) \tag{3}$$

$$d_{ij} - \alpha_{ij}\frac{(l_i+l_j)}{2} - \alpha_{ji}\frac{(l_i+l_j)}{2} \geq 0, \quad \forall i,j(i<j) \tag{4}$$

$$x_i + \frac{(l_i+l_j)}{2} \leq x_j + L(1-\alpha_{ij}), \quad \forall i,j(i<j) \tag{5}$$

$$\alpha_{ij} \in \{0,1\}, \quad \forall i,j(i \neq j) \tag{6}$$

$$\frac{l_i}{2} \leq x_i \leq L - \frac{l_i}{2}, \quad 1 \leq i \leq n \tag{7}$$

앞서 설명한 것처럼 이열 배치설계의 목적함수(The objective function)는 각 개체 간의 입출력 지점 사이의 거리와 물동량의 곱의 총합으로 계산되는 물류비용을 최소화하는 것으로 (1) 식으로 표현될 수 있다. 제한조건(Constraints) (2)와 (3)은 각 개체 간의 거리를 절대값으로 설정해주고 (4)는 만약 개체 와 개체 가 같은 열에 있으면 와 사이 거리인 는 적어도 는 되도록 해주는 조건이다. 제한조건(5)는 개체 간의 겹침 방지 조건으로 어떤 개체들도 서로 겹쳐지지 않게 만드는 조건이다. 제한조건 (6)은 이열 배치설계의 변수에서 설명한 개체 와 개체 의 상대적인 위치에 대한 이진변수를 정의해주는 조건이다. 마지막 제한조건(7)은 변수의 범위에 대한 조건이다.

본 연구에서 사용한 방법론에서는 위에서 설명한 제한조건들을 CS 알고리즘과정내에서 적용하였다. 따라서 CS 알고리즘에서는 제한조건들을 모두 만족하는 가능 해(feasible solution)만 탐색하게 되어 해를 탐색하는 과정을 더욱 효율적으로 만들어 준다.

4. Cuckoo Search 기반의 이열 배치설계 방법론

이 장에서는 이열 배치설계를 해결하기 위한 방법론에 대해서 자세히 다룬다. 본 연구에서는 메타 휴리스틱인 Cuckoo Search 알고리즘을 적용하였고 이열 배치에서의 개체 간 빈 공간(clearance)문제를 해결하면서 상대적인 위치에서의 가장 좋은 배치를 도출하기 위해 flow값이 존재하지 않는 가상의 Dummy 개체를 추가하였다. 하지만 메타 휴리스틱을 통한 새로운 이열 배치 탐색과 탐색한 이열 배치에 대한 평가를 위해서는 이열 배치형태를 순열(Permutation)과 같은 이산적인 해(Discrete solution)의 형태로 표현하는 과정이 먼저 필요하며 개체들을 어떻게 이열(double row)로 배치하는지에 따라 적용하는 메타 휴리스틱의 성능도 달라지기 때문에 적용한 CS 알고리즘을 설명하기에 앞서 먼저 이열 배치형태를 메타 휴리스틱에 적용할 수 있도록 이열 배치형태를 어떻게 표현했는지(표현방법)와 어떤 방식으로 각 개체들을 이열로 배치했는지(이열 배치 방식)에 대해 설명한다. 그 다음, 본 연구에서 사용하는 메타 휴리스틱 방법인 Cuckoo

Search(CS)알고리즘에 대해 설명하고, 조합 최적화 문제인 이열 배치설계에 CS(의 Lévy flight)를 적용하기 위한 방법으로 본 연구에서 사용한 Random-key 방식에 대해서 설명한다. 그리고 마지막으로 이열 배치에서 개체 간의 빈 공간(clearance)에 대한 문제와 함께 이를 해결하기 위해 적용한 Dummy 개체에 대해 설명한다.

4.1 이열 배치 방식(Layout representation)

본 연구에서는 개체들에 대한 이열 배치 형태를 기본적으로 개체들을 나열하는 순서(sequence)에 대한 정보를 담은 순열(permutation) π= (π[1], π[2], . . . , π[i], . . . , π[n])로 나타낸다. 하지만 어떤 순서로 개체들을 구성하여 어떻게 개체들을 윗 열(upper row)과 아랫 열(lower row)로 구분하는 지에 대한 방법을 본 연구에서는 두 가지로 사용하였다. 첫 번째는 upper row의 개체들 길이 합 와 lower row의 길이 합을 비슷하게 맞추는 방법이고 두 번째는 개체의 길이를 고려하지 않고 개체의 전체 개수를 고려하여 개체 개수의 반으로 upper row와 lower row를 구분하는 방법이다.

1) 개체의 길이를 고려해 이열(double row)간 길이 밸런스를 맞추는 방법: LBM

본 연구에서 개체들에 대해 윗 열과 아랫 열중 어느 열로 배치할지에 대한 첫 번째 방법으로 윗 열과 아랫 열에 배치되는 개체들의 길이를 고려하여 열 간의 길이 밸런스를 맞추는 방법(Length Balance based Method; LBM)을 사용하였다. 해당 방법에 대한 pseudo-code는 <Algorithm 1>에 제시하였다.

Algorithm 1. Length Balance based Method(LBM)

Input: A permutation π (sequence of departments)
Output: A upper row sequence π_U; A lower row sequence π_L;
$L_u \leftarrow 0$; $L_l \leftarrow 0$; $\pi_U = \varnothing$; $\pi_L = \varnothing$;
for $i \leftarrow$ from 1 to n **do**
  if $(L_u < L_l)$ **then**
    $L_u \leftarrow L_u + l_{\pi[i]}$; $\pi_U \leftarrow \pi_U \cup \pi$[i];
  else if $(L_l < L_u)$ **then**
    $L_l \leftarrow L_l + l_{\pi[i]}$; $\pi_L \leftarrow \pi_L \cup \pi$[i];
  else
    $r \leftarrow$ generate a uniform random number in (0, 1);
    if $(r < 0.5)$ **then**
      $L_u \leftarrow L_u + l_{\pi[i]}$; $\pi_U \leftarrow \pi_U \cup \pi$[i];
    else
      $L_l \leftarrow L_l + l_{\pi[i]}$; $\pi_L \leftarrow \pi_L \cup \pi$[i];
    end if
  end if
end for

upper row의 모든 개체에 대한 길이 합을 L_u, lower row의 모든 개체에 대한 길이 합을 L_l로 나타냈고 upper row의 순서에 대한 배열(sequence)과 lower row의 순서에 대한 배열을 각각 π_U와 π_L로 표현했다. LBM은 처음에 L_u과 L_l을 0으로 설정하고 π_U와 π_L을 비어있는 배열로 설정하여 초기화한다. 그 다음 각 개체가 upper row 또는 lower row에 할당되는 배치방식은 다음과 같다. 만약 L_u보다 L_l이 더 길면 모든 개체의 순서에 대한 순열 π의 i번째 개체의 길이 $l_{\pi[i]}$를 L_u에 더해주고 π_U에 해당 개체 π[i]를 추가한다. 반대로 만약 L_l보다 L_u가 더 길면 $l_{\pi[i]}$를 L_l에 더해주고 π_L에 π[i]를 추가해준다. 그리고 만약 L_u과 L_l이 같으면 위의 배치방식 중 하나를 랜덤하게 수행한다. 이 과정은 의 모든 개체가 upper 또는 lower row에 할당될 때까지 반복된다. 모든 과정이 끝나면 모든 개체가 또는 에 할당되어 upper row인지 lower row인지 결정된다. 위의 과정에 따라서 초기과정에 L_u과 L_l이 모두 0이므로 L_u과 L_l이 같기 때문에 π의 첫 번째 순서에 해당되는 개체 π[1]는 upper 또는 lower중 랜덤하게 배치된다.

2) 개체의 개수를 고려해 개체 개수를 고려해 cut point를 설정하는 방법: CSM

개체들에 대해 upper row와 lower row중 어느 열로 배치할지에 대한 두 번째 방법으로 모든 개체의 순서에 대한 배열 에 대해cut point를 설정하는 방법(Cut point selecting method; CSM)을 사용하였다. CSM은 배열 내의 첫 번째부터 설정한 cut point에 해당되는 순서까지의 개체들을 순서대로 upper row에 배치한다. 그리고 나머지 cut point+1번째부터 마지막 순서까지의 개체들을 순서대로 lower row에 배치한다. 본 연구에서는 cut point를 전체 개체 수의 반을 cut point로 설정하였다. 만약 개체 수가 홀수인 경우에는 개체 수의 반에 대한 값에 반올림을 적용하여 cut point를 설정하였다. <Figure 3>과 같이, 개체가 11개라고 하면 11의 반인 5.5에서 반올림하여 cut point를 6으로 설정하고 이렇게 되면 1~6번 개체가 upper row에 배치되며 7번부터 11번까지의 개체가 lower row에 배치되게 된다.

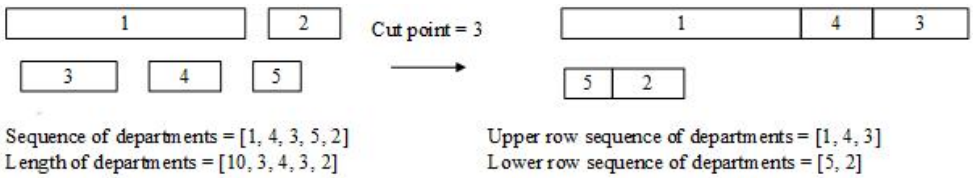


Figure 3. Cut point selecting method

4.2 Cuckoo Search 알고리즘

Cuckoo Search(CS) 알고리즘은 뻐꾸기들의 생존율을 높이기 위한 행동패턴에 영감을 받아 Yang과 Deb에 의해 고안되어진 발견적 기법이다(X. S. Yang and Deb, 2009). 일부 뻐꾸기 종은 매우 위험성이 큰 번식 전략으로써 탁란(brood parasitism)이라고 불리는 행위를 한다. 이 행위의 가장 큰 특징은 다른 새의 둥지에 자신의 알을 낳아 그 새가 자신의 알을 품어 알에서 새끼가 부화되도록 한다는 것이다. 이를 진화의 관점에서 보게 되면, 뻐꾸기들은 자신이 알을 낳은 둥지의 새가 자신의 알을 버릴 확률을 줄여 종의 생존율을 높여야 한다(Payne and Sorensen, 2005). 이러한 CS 알고리즘은 탐색이 효과적이면서 최적의 생존 전략인 Lévy flights를 새로운 해를 탐색하는데 사용함으로써 성공적으로 뻐꾸기들의 생존 행동패턴을 모방하였다.

프랑스 수학자 Paul Lévy의 이름을 딴 Lévy flights는 변화량(step length)이 확률밀도함수의 양쪽 꼬리 부분이 두꺼운 heavy-tailed 확률분포를 따르는 랜덤 워크의 한 종류이다(Brown et al., 2007). Lévy flights는 평균과 분산이 무한대라는 특성이 있어 CS 알고리즘이 해 공간을 더 효과적으로 탐색하게 해주며 정규분포를 따르는 랜덤 워크(Gaussian random walk)보다 효과적이다(X. S. Yang and Deb, 2014). 또한, 여러 연구에서 동물들이 먹이를 탐색하거나 사냥꾼이 사냥감을 탐색하기 위한 행동 패턴이 Lévy flights와 일반적으로 같은 특성이 가진다고 알려져 있다(Ouaarab et al., 2015b). Lévy flights 모형은 일반적으로 변화량의 범위가 작게 나타나고 가끔씩 변화량의 범위가 크게 나타나는 특징이 있다(Brown et al., 2007; Shlesinger et al., 1995).

CS 알고리즘에서 둥지안의 알(host egg)은 한 개의 해(solution)을 나타내고 뻐꾸기의 알(cuckoo egg)은 새로운 해를 의미한다. CS 알고리즘의 목표는 둥지 내에서 안 좋은 해(host egg)와 교체할 새롭고 더 좋을 가능성이 있는 해(cuckoo egg)를 만드는 것이다. CS 알고리즘에서 뻐꾸기의 행동패턴에 대해 모방한 3가지 규칙은 다음과 같다(X. S. Yang and Deb, 2009, 2013, 2014). (1) 각각의 뻐꾸기들은 한 번에 한개의 알(solution)을 낳고 랜덤하게 선택된 둥지에 알을 넣는다; (2) 가장 좋은 품질의 알을 가진 가장 좋은 둥지는 다음 세대에 유지된다; (3) 알을 낳을 수 있는 둥지의 수는 고정되어 있다. 뻐꾸기의 알은 둥지의 주인 새에게 $P_a \in [0,1]$의 확률로 발견되며, 그럴 경우, 둥지 주인 새는 뻐꾸기의 알을 버리거나 둥지를 버리고 새로운 곳에서 새 둥지를 짓는다. 이러한 규칙들을 기반으로 한 기본적인 CS 알고리즘의 과정은 <Algorithm 2>에서 제시하였다.

Algorithm 2. Pseudo Code of the Cuckoo Search via Lévy flights

Objective function $f(x)$, $x = (x_{1,\dots,}x_m)^T$;
Generate initial population of n host nests x_i $(i = 1, \dots, n)$;
While (t < the maximum number of iterations) or (stop criterion) **do**
  Get a cuckoo (say, x_i) randomly by Lévy flights;
  Evaluate quality/fitness of new solution x_i or $F(x_i)$;
  Randomly select a nest within the population (say, x_j) and evaluate its fitness $F(x_j)$;
  if ($F(x_i)$ > $F(x_j)$) **then**
    replace x_j by the new solution x_i;
  end if
  A fraction (p_a) of worse nests are abandoned and replaced by a new ones;
  Keep the best solutions (or nests with quality solutions);
  Rank the solutions and find the current best
end while

<Algorithm 2>를 보면, 초기화 단계에서 목적함수 $f(x)$를 정의하고, 각각의 목적함수에 대한 해 x_i $(i = 1, \dots, n)$를 랜덤하게 생성하여 해 집단(population)을 형성한 뒤, 그 중 가장 좋은 해(current best solution)를 찾기 위해 각각의 해에 대한 목적함수 값(objective function value; OFV)을 계산한다. 그리고 CS 알고리즘은 식(8)과 같이, Lévy flights를 통해 새로운 해 $x_i^{(t+1)}$를 생성한다(X. S. Yang and Deb, 2014).

$$x_i^{(t+1)} = x_i^{(t)} + \alpha L\acute{e}vy(s,\lambda) \tag{8}$$

where

$$L\acute{e}vy(s,\lambda) = \frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}},\ (0 < \lambda < 2) \tag{9}$$

여기에서 $x_i^{(t)}$는 t 번째 iteration에서의 해이다. 그리고 $\alpha > 0$은 CS 알고리즘에 적용하고 있는 최적화 문제의 크기와 관련해 해가 변동되는 정도(step size)를 조절해주는 요소이이다. $L\acute{e}vy(s,\lambda)$는 앞에서 설명한 Lévy flights 랜덤 워크로 발생되는 값으로 식(9)를 통해 도출된다. 여기에서 Γ는 감마함수(Gamma function)를 나타내고, $L\acute{e}vy(s,\lambda)$의 랜덤하게 변동되는 정도(step length)를 나타내는 s는 Lévy 분포를 따르는 값으로 Lévy 분포로부터 도출되는 랜덤 값이며 식(10)을 통해 값이 계산된다.

$$s = \frac{u}{|v|^{1/\lambda}} \tag{10}$$

식(10)에서의 u와 v는 식(11)같이 평균이 0이고, 표준편차가 각각 σ_u와 σ_v인 정규분포로부터 도출되는 랜덤 값들이고, σ_u과 σ_v은 식(12)를 통해 계산된다. (X. S. Yang and Deb, 2014).

$$u \sim N(0,\sigma_u^2),\quad v \sim N(0,\sigma_v^2) \tag{11}$$

where

$$\sigma_u = \left\{\frac{\Gamma(1+\lambda)\sin(\pi\lambda/2)}{\Gamma[(1+\lambda)/2]\lambda 2^{(\lambda-1)/2}}\right\}^{1/\lambda},\quad \sigma_v = 1 \tag{12}$$

이렇게 Lévy flights를 통해 새로운 해 x_i를 생성한 후, 이 해에 대한 목적함수 값(OFV) $F(x_i)$를 구한다. 그 다음, x_i와 OFV를 비교하기 위해 현재의 해 집단(population)에서 랜덤하게 하나의 해(x_j)를 선택한다. 새로운 해 x_i는 만약 목적함수 값 $F(x_i)$이 x_j의 목적함수 값 $F(x_j)$보다 좋다면 x_j를 x_i로 바꾸게 된다. 그 다음 단계로, $P_a \in [0,1]$의 비율만큼 안 좋은 해를 버리고 완전히 랜덤하게 해를 발생시키거나 Lévy flights가 크게 튀는 경우를 활용함으로써 새로운 곳에 새 둥지가 지어진다. 여기에서 P_a를 높게 설정하면 해를 넓게 탐색하는 과정(diversification)의 비율이 증가하고 반대로 P_a를 낮게 설정하면 그 과정의 비율

이 낮아진다. 이 과정이 끝나고 나면 현재 iteration까지 중의 해 집단에서 가장 좋은 해를 찾기 위해 OFV가 좋은 순서대로 정렬시킨다. 지금까지 설명한 과정들은 iteration의 최대 숫자에 도달할 때까지 반복된다.

본 연구에서는 CS의 개선된 알고리즘을 기반으로 적용하였다(Ouaarab et al., 2014b). 여기에서 개선된 점은 똑똑한 뻐꾸기(smart cuckoo)라는 새로운 유형의 뻐꾸기를 추가하였다. 이 새로운 유형의 뻐꾸기는 자신이 알을 낳기로 선택한 둥지가 좋은 선택이었는지 확인하기 위해 알을 낳기로 결정한 둥지를 알을 낳기 전과 후에 관찰하여 알이 버려 지기 전에 더 좋은 둥지에 알을 낳기 위해 둥지를 바꾸거나 다른 곳으로 날아간다. 그래서 현재의 해로부터 $P_c \in [0,1]$만큼의 비율의 뻐꾸기들은 Lévy flights를 통해 새로운 더 좋은 해를 탐색한다. 이러한 뻐꾸기를 추가함으로써 CS가 개선되는 점은 현재의 iteration에서의 해들의 주변 탐색을 더 강화할 수 있으며, <Algorithm 3>을 통해 개선된 CS의 과정을 제시하였다.

Algorithm 3. Pseudo Code of the Improved Cuckoo Search Algorithm

Objective function $f(x)$, $x = (x_{1,\dots,}x_m)^T$;
Generate initial population of n host nests $x_i (i = 1, \dots, n)$;
While (t < the maximum number of iterations) or (stop criterion) **do**
  Start searching with a fraction (p_c) of smart cuckoos
  Get a cuckoo (say, x_i) randomly by Lévy flights;
  Evaluate quality/fitness of new solution x_i or $F(x_i)$;
  Randomly select a nest within the population (say, x_j) and evaluate its fitness $F(x_j)$;
  if ($F(x_i)$ > $F(x_j)$) **then**
    replace x_j by the new solution x_i;
  end if
  A fraction (p_a) of worse nests are abandoned and replaced by a new ones;
  Keep the best solutions (or nests with quality solutions);
  Rank the solutions and find the current best
end while

4.3 Random-key encoding scheme

CS 알고리즘이 연속 최적화 문제에 맞춰 고안되었기 때문에 Lévy flights의 연속적인 해 탐색범위를 조합 최적화 문제의 해에 적용할 수 있도록 일종의 변형(modification)과정이 필요하다. 그래서 본 연구에서는 최근 많은 연구들에서 메타 휴리스틱을 조합 최적화 문제에 적용할 때 쉽고 성능의 손실 없이 적용 가능하다고 많이 사용되고 있는 Random-key 방법을 사용하였다. Random-key방법은 연속적인 공간(continuous space)을 이산적인 조합 가능한 공간(combinatorial space)으로 바꿔 주는데 사용하는 기법이다(Bean, 1994).

본 연구에서는 Random-key를 통해 초기해 생성 및 새로운 해를 탐색하는 과정에도 적용하였다. Random-key로 이열 배치설계의 해를 표현하는 방법은 <Figure 4>와 같이 설명될 수 있다.

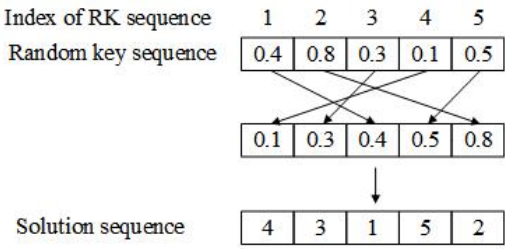


Figure 3. Random-key encoding scheme

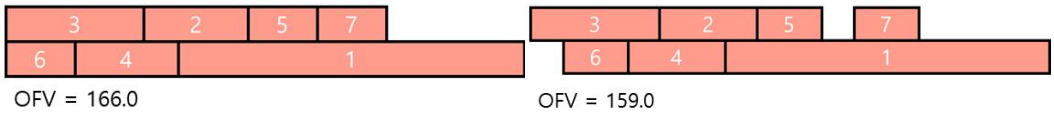
만약 이열 배치설계에서 배치해야 할 개체의 수가 5개라면, 그림 5와 같이, [0, 1]사이의 범위에 있는 실수를 균일한 분포(uniform)로 랜덤하게 5개를 발생시켜 배열에 저장한다. 여기서 Random-key 배열(sequence)의 순서(index)를 이열 배치설계에서 주어지는 개체들의 번호로 인식한다. 그 다음, Random-key

배열을 섞어서 개체들의 순서도 섞게 되는데 여기서 Random-key 배열을 섞는 방법은 Random-key 배열의 Random-key값을 작은 순서대로 정렬시키는 것이다. 그렇게 되면 가장 작은 0.1의 index인 4가 가장 앞으로 오고 가장 큰 0.8의 index인 2가 가장 뒤로 가면서 원래 개체의 순서가 [1,2,3,4,5]였던 것이 [4,3,1,5,2]로 바뀌게 된다.

4.4 Dummy 개체 적용

CS 알고리즘을 통해 얻게 되는 배치는 개체들의 상대적인 배치를 찾을 뿐 해당 위치에서 정확히 개체들이 어디에 배치되어야 하는지는 CS 알고리즘을 통해 찾을 수 없다. 배치하는 개체들의 각각의 길이와 물동량에 따라서 달라지지만 경우에 따라서 개체들에 대한 배치가 특정 개체들 간의 공간이 있는 경우에 총 물류이동 거리가 가장 낮을 수 있다. 이에 대한 대표적인 예시가 바로 DRLP의 여러 배치문제 중에서 'H7'문제 형태이다. 'H7'은 Hungerländer and Anjos(2012)의 연구에서 처음 제안된 문제 형태이며 문제의 구성은 <Figure 4>에서 보는 것과 같이, 개체의 수가 7개이고, 각각의 개체의 길이는 10, 3, 4, 3, 2, 2, 2이다. 해당 문제 형태의 최적 배치형태는 <Figure 5>와 같이 나타나는데, (a)는 공간을 넣지 않고 배치했을 때의 최적 배치 형태이며, (b)는 개체간의 공간을 넣어주었을 때의 최적 배치 형태이다. 두 배치 형태 모두 개체들의 상대적인 위치 즉, 배치 순서는 윗 열이 [3,2,5,7], 밑 열이 [6,4,1]로 동일하지만 공간을 넣지 않았을 경우에는 목적함수 값(OFV)이 166이고 공간을 넣었을 때는 목적함수 값이 159로 공간을 넣지 않았을 때보다 넣었을 때 목적함수 값이 더 낮아지는 것을 확인할 수 있다. 따라서 CS 알고리즘을 통해 탐색한 개체들의 상대적인 위치에 대해서 정확한 배치 위치를 찾는 방법이 필요하다. 본 연구에서는 물동량 값(flow)이 없는 가상의 Dummy 개체를 설정하여 개체 간에 공간을 넣어주었다.

Figure 4. 'H7' Instance



(a) Optimal layout without clearance (b) Optimal layout with clearance

Figure 5. Optimal layout of 'H7' Instance

Dummy 개체를 추가할 때는 2가지를 설정해줘야 한다. 먼저, Dummy 개체의 수를 몇 개로 설정할 것인지 결정해줘야 하고 두 번째로 Dummy 개체의 길이를 몇으로 설정할 것인지를 결정해야한다. Dummy 개체의 길이를 짧게 설정하고 개수를 많이 넣어줄수록 더 좋은 배치를 찾을 확률이 높아진다. 하지만 개체의 길이를 짧게 설정할수록 배치해야할 개체의 수가 증가해 문제의 복잡도가 높아져 문제를 해결하는데 소요되는 시간이 기하급수적으로 증가한다. 또한, 개체의 수를 필요 이상으로 많이 설정하게 되면, 불필요하게 설정된 Dummy 개체들로 인하여 문제를 해결하는데 있어서 불필요한 시간소요가 발생하게 된다. 따라서 Dummy 개체의 길이를 너무 짧지도 길지도 않게 설정해주며 개수 역시 너무 많지도 적지도 않게 적절하게 설정해주는 것이 중요하다. 본 연구에서는 많은 시행착오를 거쳐 Dummy 개체의 길이를 0.5로 설정하고 해결해야할 DRLP 유형의 개체 수에 따라 비례하게 설정하였다.

5. 실험 결과

본 장에서는 4장에서 설명한 본 연구의 방법론의 성능을 평가하기 위해 진행한 실험에 대한 결과를 살펴보고 이전 연구들과 비교해본다. 본 연구에서 설명한 CS알고리즘을 포함한 방법론은 JAVA 프로그래밍 언

어를 통해 구현되었다. 실험은 Microsoft Windows 10 운영체제, Intel Core i9 CPU (3.3 GHz), 32GB RAM의 컴퓨터에서 진행되었다. 실험한 이열 배치설계 문제 데이터는 잘 알려진 문제들로 이전 연구들에서 실험이 진행되었던 개체 수가 5개부터 16개까지 18개의 문제들을 사용하였다(Amaral, 2006, 2013a; Fischer et al., 2019; Heragu and Kusiak, 1991; Hungerländer and Anjos, 2012; Secchin and Amaral, 2019).

5.1 매개변수 설정 (Parameters setting)

CS 알고리즘의 매개변수는 해 집단의 크기(Population Size; n), 해 집단에서 해를 버리는 비율(p_a), 해 집단에서 smart cuckoo 역할을 하는 비율(p_c), *Lévy flight*의 step size 값(α)가 있다. 본 연구에서는 CS 알고리즘의 적절한 매개변수 값을 적용하기 위해 CS의 각 매개변수들에 대해 일정 간격을 두고 모든 매개변수 조합들에 대해 실험을 진행해 전반적인 문제들에 대해 가장 결과 값이 좋았던 조합으로 설정하였다. 그 결과 n = 10, p_a = 0.25, p_c = 0.6, α = 0.3으로 설정되었다. 해당 매개변수 조합으로 실험이 진행된 문제당 10번씩 실험을 진행하였다.

5.2. 이전 연구들의 MIP 모형과 실험결과 비교

비교를 진행한 연구에 대해서는 가장 최근에 이열 배치설계에 대한 연구가 진행이 되었던 MIP모형들의 결과로 비교하였다. 본 연구에서는 (Secchin and Amaral, 2019)에서 제안된 MIP 모형을 M1으로 표기하였으며, (Chae and Regan, 2020)에서 제안된 MIP 모형을 M2로 표기하였다. 그리고 본 연구에서 제시하는 방법론을 Improved CS로 표기하였다. 이전 연구들의 MIP 모형과 같이 실험이 진행된 문제들에 대한 연구결과는 <Table 3>와 같이 나타났다. 표의 행에 대한 표기를 설명하면, N은 개체의 수를 나타내며 Optimal은 해당 문제의 최적 값을 의미한다. Best는 가장 잘 찾은 결과 값이며 M1과 M2는 MIP에 대한 풀이 값으로 최적값을 찾기 때문에 Best는 CS 만 해당이 된다. 단 시간제한으로 최적 해를 찾지 못한 경우에 대해서는 가장 좋은 값을 표기 하였다. Time(sec)은 각 방법론이 해당 문제에 대한 결과 값을 찾아내는데 걸린 시간을 초단위로 표현한 것을 의미한다. STD는 본 연구에서 10번 진행한 결과 값에 대한 표준 편차 값이고 Avg Time은 10번 진행한 실험의 평균 소요시간을 의미한다. 문제의 <Table 3>의 결과를 살펴보면, 본 연구 방법론의 STD 값이 모든 문제들에 대해 0의 값을 보였다. 이는 20개 이하의 <Table 3>의 문제들에 대해서는 방법론의 성능이 변동이 적고 결과 값을 잘 찾아내는 것을 의미한다. 각 연구들의 소요시간에 대해 살펴보면, 12개 이하의 문제들에 대해서는 MIP 모형의 소요시간이 본 연구의 방법론보다 더 빠른 결과값을 보였다. 하지만 13개 이상의 문제들에 대해서는 문제의 크기가 커질수록 MIP 모형의 소요시간이 크게 증가 하였다. 또한, 16개의 문제에 대해 이전 연구들의 MIP 모형은 최적 값을 찾지 못했다. 본 연구에서 제안한 방법론은 모든 문제에 대해 이전에 MIP로 찾은 최적 값을 매우 빠르게 찾았고 또한 지속적이고 안정적으로 값을 도출하였다. 이는 이 연구에서 테스트 되지 않았지만 큰 크기의 문제에서도 빠르게 좋은 해를 찾아 줄 것을 기대하게 한다.

Table 3. Result comparison

Instance			M1	M2	CS (in this study)		
Name	N	Optimal	Time (sec)	Time (sec)	Best	STD	Avg Time (sec)
HA5	5	52.5	0.05	0.02	52.5	0.0	1.291
HA6	6	190.5	0.06	0.05	190.5	0.0	1.324
HA7	7	159.0	0.11	0.08	159.0	0.0	3.76
HA8	8	189.5	0.39	0.19	189.5	0.0	3.769
HA9	9	486.5	1.02	0.75	486.5	0.0	2.957
HA10	10	821.0	8.44	2.36	821.0	0.0	10.655
HA11	11	773.5	15.39	2.98	773.5	0.0	12.354
HA12	12	1021.0	76.39	28.59	1021.0	0.0	15.411
HA13	13	1520.5	327.97	100.80	1520.5	0.0	17.824
HA14	14	1833.5	3,358.94	882.23	1833.5	0.0	19.751
HA15	15	2624.5	3,214.06	1918.53	2624.5	0.0	23.932
SA14a	14	2904.0	3,254.69	1207.88	2904.0	0.0	20.102
SA14b	14	2736.0	5,469.83	1532.05	2736.0	0.0	21.023
Am14_1	14	2738.5	4,986.4	1533.27	2738.5	0.0	21.893
HK15	15	16570.0	12297.41	2148.73	16570.0	0.0	25.297
Am15	15	3195.0	11221.34	5002.63	3195.0	0.0	27.465
Am16a	16	7365.5*	86400** (27.09%)***	86400** (13.69%)***	7365.5	0.0	30.127
Am16b	16	5870.5*	86400** (17.80%)***	86400** (8.29%)***	5870.5	0.0	30.372

* best solution found before the process stops
** The search process stops due to time limit (24 hrs) and the optimality gap is shown in parenthesis
*** Optimality gap

6. 결론

예전부터 오래도록 연구가 진행되었던 배치설계에 관련된 연구는 최근 배치설계의 효율화가 중요해진 시점에서 더욱 필요한 연구 분야이다. 본 연구에서는 배치설계의 다양한 조합 최적화 문제 중 제조환경에서 뿐만 아니라, 다양한 실제 업무 환경에서 가장 많이 활용되고 있는 이열 배치설계(Double row layout problem)에 대해 다루었고, 최근 규모가 커지고 있는 제조시설이나 물류센터에서 유동적인 배치설계를 고려해 규모가 큰 문제유형 해결에 집중하였다. 그리고 이열 배치설계를 해결하기 위한 방법론으로 메타 휴리스틱 중 하나인 Cuckoo Search 알고리즘을 사용하였고, 이열 배치설계 문제에 CS 알고리즘을 적용하기 위해 Levy flight 값에 Random-key 방식을 사용하여 해 탐색 공간을 이산적으로 바꾸었다. 그리고 이열 배치 방식으로 upper row와 lower row의 길이 밸런스를 맞추는 방법(Length balance based method)과 cut point를 설정하는 방법(Cut point selecting method)을 사용하여 적용한 CS 알고리즘의 해 집단(population)을 2개로 가져갔다. 또한, 배치의 빈 공간에 대한 문제를 해결하기 하고 물류비용을 더 줄일 수 있는 배치를 찾기 위해 Dummy 개체들을 추가하여 CS 알고리즘을 적용하였다.

이전 연구들에서 사용되었던 여러 가지 이열 배치설계 문제들에 대해 실험을 진행해 결과를 도출하였다. 실험결과, 이미 정확 기법(Exact method)을 통해 최적 값이 도출된 문제의 크기가 작은 문제들의 경우, 빠른 시간 내에 최적 값을 찾아냈다. 그리고 정확 기법을 통해 찾아낼 수 없는 규모의 문제에 대해서는 본 연구의 방법론이 더 빠른 시간 내에서 결과 값을 찾아내었다.

본 연구는 물류 및 유통 산업뿐만 아니라 현실에서 활용 범위가 넓은 이열 배치설계에 대한 해법을 제시함으로써 일렬 배치와 관련된 다양한 분야에서 관련 비용을 절감시킬 수 있을 것으로 예상된다. 또한, 배치설계의 다른 유형의 문제에도 해당 방법론을 적용해 빠른 시간 안에 좋은 배치를 찾을 수 있을 것이라고 판단된다. 하지만 본 연구의 이열 배치설계는 배치되는 공간에 대해 제한을 두지 않아 현실적인 배치설계와는 조금 거리가 있다고 할 수 있다. 그렇기 때문에 배치공간에 대한 제한이 추가된 이열 배치설계에 대한 연구도 진행되면 실제 산업에 실용적으로 적용될 수 있을 것으로 예상된다.

참고문헌

Ahonen, H., De Alvarenga, A. G., and Amaral, A. R. S. (2014). Simulated annealing and tabu search approaches for the Corridor Allocation Problem. European Journal of Operational Research, 232(1), 221–233.

*Ain, R., and Bey, E. (2012). Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm Amira Gherboudj *, Abdesslem Layeb and Salim Chikhi. Science, 4(4), 2012.*

Amaral, A. R. S. (2006). On the exact solution of a facility layout problem. European Journal of Operational Research, 173(2), 508–518.

Amaral, A. R. S. (2012a). The corridor allocation problem. Computers and Operations Research, 39(12), 3325–3330.

Amaral, A. R. S. (2012b). The corridor allocation problem. Computers and Operations Research, 39(12), 3325–3330.

Amaral, A. R. S. (2013a). A parallel ordering problem in facilities layout. Computers and Operations Research, 40(12), 2930–2939.

Amaral, A. R. S. (2013b). Optimal solutions for the double row layout problem. Optimization Letters, 7(2), 407–413.

Amaral, A. R. S. (2019). A mixed-integer programming formulation for the double row layout of machines in manufacturing systems. International Journal of Production Research, 57(1), 34–47.

Amaral, A. R. S. (2020). A heuristic approach for the double row layout problem. Annals of Operations Research.

Anjos, M. F., Fischer, A., and Hungerländer, P. (2016). Solution approaches for the double-row equidistant facility layout problem. In Operations Research Proceedings 2014 (pp. 17–23). Springer.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing, 6(2), 154–160.

Benjaafar, S., and Sheikhzadeh, M. (2000). Design of flexible plant layouts. In IIE Transactions (Institute of Industrial Engineers) (Vol. 32, Issue 4).

Braglia, M., Zanoni, S., and Zavanella, L. (2003). Layout design in dynamic environments: Strategies and quantitative indices. International Journal of Production Research, 41(5), 995–1016.

Brown, C. T., Liebovitch, L. S., and Glendon, R. (2007). Lévy flights in Dobe Ju/hoansi foraging patterns. Human Ecology, 35(1), 129–138.

Burnwal, S., and Deb, S. (2013). Scheduling optimization of flexible manufacturing system using cuckoo search-based approach. International Journal of Advanced Manufacturing Technology, 64(5–8), 951–959.

Chae, J., and Regan, A. C. (2016). Layout design problems with heterogeneous area constraints. Computers and Industrial Engineering, 102(October), 198–207.

Chae, J., and Regan, A. C. (2020). A mixed integer programming model for a double row layout problem. Computers and Industrial Engineering, 140(June 2019), 106244.

Chung, J., and Tanchoco, J. M. A. (2010). The double row layout problem. International Journal of Production Research, 48(3), 709–727.

Civicioglu, P., and Besdok, E. (2013). A conceptual comparison of the Cuckoo-search, particle swarm optimization,

differential evolution and artificial bee colony algorithms. Artificial Intelligence Review, 39(4), 315-346.

Drira, A., Pierreval, H., and Hajri-Gabouj, S. (2007). Facility layout problems: A survey. Annual Reviews in Control, 31(2), 255-267.

Fischer, A., Fischer, F., and Hungerländer, P. (2019). New exact approaches to row layout problems. Mathematical Programming Computation, 11(4), 703-754.

Gandomi, A. H., Yang, X.-S., and Alavi, A. H. (2013). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Engineering with Computers, 29(1), 17-35.

Gandomi, A. H., Yang, X.-S., Talatahari, S., and Deb, S. (2012). Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization. Computers and Mathematics with Applications, 63(1), 191-200.

Garcia-Hernandez, L., Salas-Morera, L., Garcia-Hernandez, J. A., Salcedo-Sanz, S., and Valente de Oliveira, J. (2019). Applying the coral reefs optimization algorithm for solving unequal area facility layout problems. Expert Systems with Applications, 138, 112819.

Garey, M. R., and Johnson, D. S. (1979). Computers and intractability (Vol. 174). freeman San Francisco.

Guan, J., Lin, G., Feng, H. Bin, and Ruan, Z. Q. (2020). A decomposition-based algorithm for the double row layout problem. Applied Mathematical Modelling, 77, 963-979.

Heragu, S. S., and Kusiak, A. (1988). Machine Layout Problem in Flexible Manufacturing Systems. In Operations Research (Vol. 36, Issue 2, pp. 258-268).

Heragu, S. S., and Kusiak, A. (1991). Efficient models for the facility layout problem. European Journal of Operational Research, 53(1), 1-13.

Hungerländer, P., and Anjos, M. F. (2012). A semidefinite optimization approach to space-free multi-row facility layout. Groupe d'études et de recherche en analyse des décisions.

Ingole, S., and Singh, D. (2017). Unequal-area, fixed-shape facility layout problems using the firefly algorithm. Engineering Optimization, 49(7), 1097-1115.

Kang, S., and Chae, J. (2017). Harmony search for the layout design of an unequal area facility. Expert Systems with Applications, 79, 269-281.

Kang, S., Kim, M., and Chae, J. (2018). A closed loop based facility layout design using a cuckoo search algorithm. Expert Systems with Applications, 93, 322-335.

Kim, M., and Chae, J. (2019). Monarch butterfly optimization for facility layout design based on a single loop material handling path. Mathematics, 7(2).

Levary, R. R., and Kalchik, S. (1985). Facilities layout—a survey of solution procedures. Computers and Industrial Engineering, 9(2), 141-148.

Li, X., and Yin, M. (2013). A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem. International Journal of Production Research, 51(16), 4732-4754.

Liu, S., Zhang, Z., Guan, C., Zhu, L., Zhang, M., and Guo, P. (2020). An improved fireworks algorithm for the constrained single-row facility layout problem. International Journal of Production Research, 0(0), 1-19.

Meller, R. D., and Gau, K.-Y. (1996). The facility layout problem: recent and emerging trends and perspectives. Journal of Manufacturing Systems, 15(5), 351-366.

Murray, C. C., Smith, A. E., and Zhang, Z. (2013). An efficient local search heuristic for the double row layout problem with asymmetric material flow. International Journal of Production Research, 51(20), 6129–6139.

Ouaarab, A., Ahiod, B., and Yang, X. S. (2014a). Discrete cuckoo search algorithm for the travelling salesman problem. Neural Computing and Applications, 24(7–8), 1659–1669.

Ouaarab, A., Ahiod, B., and Yang, X. S. (2014b). Improved and discrete cuckoo search for solving the travelling salesman problem. In Cuckoo search and firefly algorithm (pp. 63–84). Springer.

Ouaarab, A., Ahiod, B., and Yang, X. S. (2015a). Discrete cuckoo search applied to job shop scheduling problem. In Recent Advances in Swarm Intelligence and Evolutionary Computation (pp. 121–137). Springer.

Ouaarab, A., Ahiod, B., and Yang, X. S. (2015b). Random-key cuckoo search for the travelling salesman problem. Soft Computing, 19(4), 1099–1106.

Payne, R. B., and Sorensen, M. D. (2005). The cuckoos (Vol. 15). Oxford University Press.

Scalia, G., Micale, R., Giallanza, A., and Marannano, G. (2019). Firefly algorithm based upon slicing structure encoding for unequal facility layout problem. International Journal of Industrial Engineering Computations, 10(3), 349–360.

Secchin, L. D., and Amaral, A. R. S. (2019). An improved mixed-integer programming model for the double row layout of facilities. Optimization Letters, 13(1), 193–199.

Shlesinger, M. F., Zaslavsky, G. M., and Frisch, U. (1995). Lévy flights and related topics in physics.

Singh, S. P., and Sharma, R. R. K. (2006). A review of different approaches to the facility layout problems. The International Journal of Advanced Manufacturing Technology, 30(5–6), 425–433.

Srivastava, P. R., Chis, M., Yang, X. S., and Deb, S. (2012). An efficient optimization algorithm for structural software testing. International Journal of Artificial Intelligence, 8(S12), 68–77.

Tompkins, J. A., White, J. A., Bozer, Y. A., and Tanchoco, J. M. A. (2010). Facilities planning. John Wiley & Sons.

Tubaileh, A., and Siam, J. (2017). Single and multi-row layout design for flexible manufacturing systems. International Journal of Computer Integrated Manufacturing, 30(12), 1316–1330.

Walton, S., Hassan, O., Morgan, K., and Brown, M. R. (2011). Modified cuckoo search: a new gradient free optimisation algorithm. Chaos, Solitons & Fractals, 44(9), 710–718.

Wang, S., Zuo, X., Liu, X., Zhao, X., and Li, J. (2015). Solving dynamic double row layout problem via combining simulated annealing and mathematical programming. Applied Soft Computing Journal, 37, 303–310.

Yang, X., Cheng, W., Smith, A. E., and Amaral, A. R. S. (2020). An improved model for the parallel row ordering problem. Journal of the Operational Research Society, 71(3), 475–490.

Yang, X. S., and Deb, S. (2009). Cuckoo search via Lévy flights. 2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings, 210–214.

Yang, X. S., and Deb, S. (2010). Engineering optimisation by cuckoo search. International Journal of Mathematical Modelling and Numerical Optimisation, 1(4), 330–343.

Yang, X. S., and Deb, S. (2013). Multiobjective cuckoo search for design optimization. Computers & Operations Research, 40(6), 1616–1624.

Yang, X. S., and Deb, S. (2014). Cuckoo search: Recent advances and applications. Neural Computing and

Applications, 24(1), 169-174.

Zhang, Z., and Cheng, W. (2014). Decomposition strategies and heuristic for double row layout problem. Computer Integrated Manufacturing Systems, 20(3), 559-568.

Zhang, Z., and Murray, C. C. (2012). A corrected formulation for the double row layout problem. International Journal of Production Research, 50(15), 4220-4223.

Zuo, X., Murray, C. C., and Smith, A. E. (2014). Solving an extended double row layout problem using multiobjective tabu search and linear programming. IEEE Transactions on Automation Science and Engineering, 11(4), 1122-1132.

Zuo, X., Murray, C. C., and Smith, A. E. (2016). Sharing clearances to improve machine layout. International Journal of Production Research, 54(14), 4272-4285.