**2023 겨울학기 연구참여 보고서**

TSPTW

2024.01.02~2024.02.08(6주)

Logistics Lab

지도교수: 김병인 교수님

지도사수: 이승엽

참여학생: 김도연

**목차**

1. Introduction
   1. Objectives
2. Method
   1. Insertion heuristic
   2. M-n opt(relocation)
3. Results
4. Conclusion
5. Review
6. appendix

**Introduction**

본 연구에서는 한대의 차량이 time window가 존재하는 각각의 장소를 가장 짧은 시간 안에, 가장 적은 거리를 이동하며 방문할 수 있는 route를 찾는 것이 목적이다. 참고한 논문에서 제시한 알고리즘을 통해서 solution을 찾고, 해당 결과를 다양한 optimization방법을 통해 향상시키고자 하였다.

**Method**

1. **Insertion heuristic**

“ALGORITHMS FOR THE VEHICLE ROUTING AND SCHEDULING PROBLEMS WITH TIME WINDOW CONSTRAINTS”논문에서는 TW가 존재하는 TSP문제에 대해 크게 4가지 방법을 제시하고 있다. 그중에서 본 연구에서 사용한 방법은 Insertion heuristic algorithm이다.

해당 논문에서는 insertion을 위한 초기 경로설정에 대한 언급은 없으므로, 다음과 같은 방법으로 초기 경로를 설정한 다음 insertion과정을 진행하였다. 우선 각각의 노드 사이 거리를 비교하여, 가장 거리가 먼 2개의 노드를 초기경로로 설정한다. 또한 논문 내에서는 depot을 설정하여, 차량이 depot에서 출발하고 다시 돌아오는 것으로 설명하고 있다. 하지만 주어진 instance의 특성에 맞게, 가상의 노드를 생성하여 노드 중 한 곳에서 출발하여 모든 노드를 방문하면 마지막 노드가 도착지가 되도록 즉, open path로 solution을 도출해내고자 하였다. 가상의 노드는 모든 노드들과의 거리가 0인 노드를 의미한다. 또한, instance내에서 거리와 시간개념을 구분하지 않고 사용하고 있으므로 본 연구에서도 노드 사이의 거리와 소요시간을 같은 개념으로 설정하였다.

위와 같이 초기 경로를 설정한 다음, 노드를 하나씩 삽입하는 과정은 다음과 같다텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명.

이미 정해진 경로에 추가로 노드를 삽입할 때 고려해야할 요소는 2가지이다. 외부에 존재하는 노드들 중 어떤 노드를 추가할 것인지 선택해야하고, 선택한 외부 노드를 어디에 추가할 것인지도 정해야한다. 이 2가지 요소를 결정하기 위해서 논문에서는 위와 같은 과정을 제시하였다. 우선 C11은 늘어나는 거리에 대한 가중치, C12는 늘어나는 시간에 대한 가중치이며 둘을 더한 다음 적절한 비례상수를 곱한 C1을 통해 해당 값이 가장 적은 것, 즉 가장 거리가 덜 늘어나면서 동시에 시간도 적게 늘어나는 노드를 찾는 방식이다.

우선 C11부터 보면 아래와 같은 방식으로 그 값을 구하고 있다.

텍스트, 스크린샷, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

diu+duj는 원래 있던 경로에서 u라는 외부노드를 추가하였을 때 늘어난 거리를 말하는 것이며, dij는 기존의 경로를 뜻한다. 그러므로 C11은 외부노드로 인해 전체 경로가 얼만큼 늘어나는가에 대한 값을 뜻한다.

다음으로 C12는 다음과 같은 방식으로 그 값을 구하고 있다.

스크린샷, 텍스트, 라인, 도표이(가) 표시된 사진

자동 생성된 설명

bju는 u노드를 추가함으로써 변경된 경로에서 j노드의 출발시간이며 bj는 기존 경로에서 j노드의 출발시간이다. 따라서 C12는 u노드를 추가했을 때 얼만큼의 시간이 늘어나는가에 대한 값을 뜻한다. 위와 같은 방법으로 구한 C11과 C12를 각각에 비례상수를 곱해서 더한 것이 C1이다. 이때 비례상수 값은 논문에서 이미 실험 후에 검증된 값을 대입하여 활용했다.

해당 과정을 외부노드 u에 대해서 path의 모든 경로사이에 추가했을 때, C1값을 구한 다음, 그 값이 가장 작은 경우를 외부노드 u의 고유 C1값으로 저장한다. 이후 경로에 포함되지 않은 나머지 외부 노드들에 대해서도 각각 외부 노드 고유의 C1값을 구한다. 그리고 최종적으로 외부노드들 중에서 고유 C1값이 가장 작은 외부노드를 경로에 추가하는 방식으로 insertion을 진행한다. Insertion은 경로에 포함되지 않은 외부노드가 존재하지 않을 때까지 반복하도록 하여 모든 노드가 경로에 포함될 수 있도록 한다.

1. **M-n opt(relocation)**

위와 같은 방법으로 solution을 구한 다음, 해당 sol을 좀 더 향상시키기 위해서 본 연구에서는 다음과 같은 방법을 사용하였다. 우선 M-N opt 알고리즘이란 M개의 노드로 이루어진 묶음을 N개의 노드 묶음과 맞바꾸는 것을 말한다. 이때 둘의 위치를 바꾸었을 때 거리 또는 시간이 줄어듦과 동시에 모든 노드들의 Time window를 어긋나지 않게 한다면 해당 경로로 업데이트하게 된다. 해당 알고리즘에서 M과 N은 사용자가 입력하도록 하여, 일반화된 알고리즘을 작성해야하지만, 본 연구에서는 M=1,N=0인 1-0 opt알고리즘만을 구현하였다. 이는 relocation과 같다고 볼 수 있다. 이때 N=0이라는 것은 노드와 노드를 맞바꾸는 것이 아니라, 노드를 노드와 노드 사이 엣지에 끼워 넣는 것으로 볼 수 있다.

1-0 opt는 다음과 같이 진행된다. 우선 relocate할 노드를 정한다. 그리고 해당 노드를 모든 엣지 사이에 끼워넣어 보고, 만약 바뀐 경로에서 거리 또는 시간이 줄어듦과 동시에 모든 노드들의 timewindow가 어긋나지 않는다면 해당 경로로 업데이트한다. 모든 노드에 대해 이와 같은 relocate를 적용 시켜 최종적으로 improve된 sol을 도출해낸다. 하지만 이때 relocate할 노드를 정하고 해당 노드를 다른 엣지에 넣어볼 때 이를 모든 엣지에 다 대입시킬 경우, 너무 많은 계산시간이 소요된다는 점을 고려하고자 본 연구에서는 다음과 같은 방법으로 탐색범위를 설정하고자 하였다. 텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

위 그림에서 Num.1이 relocate할 노드이다. 그림과 같이 Num1을 제거할 경우하더라도 Num.0의 departure time은 변하지 않는다. 그러므로 Num.1노드를 제거했으니 Num.2부터 경로를 보면, 만약 Num.1을 Num3과 Num.4 사이에 넣을 경우, 이미 누적된 시간값을 고려했을 때 이미 Num.1의 latest time을 넘어가게 된다. 그러므로 Num.4이후로 Num.1을 삽입하는 것은 무의미한 과정이라고 볼 수 있다. 이와 같은 과정으로 Num.1의 탐색범위를 좁히도록 하였다.

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

위 그림에서는 Num.4에 대해서 해당 노드를 앞으로 relocate할 때 탐색범위를 설정하는 과정이다. Relocate할 노드는 앞과 뒤 모두 탐색범위를 설정하고자 하였다. 우선 Num.4를 기존 위치보다 앞쪽으로 relocate했을 때 Num.3의 Latest time을 벗어나지 않는지 확인하는 것이 핵심이다. 예를 들어 만약 Num.4를 Num.0과 Num.1사이에 삽입할 경우를 보자. 그렇다면 Num.4에서 아무리 빨리 출발해도 Num.4의 early time이후 일 것이다. 그러므로 Num.4를 추가했을 때, Num.4의 early time에서 Num3까지 waiting time을 고려하지 않고도 거리만으로 계산했을 때 이미 Num.3의 latest time을 넘게 된다. 따라서 Num.4는 Num.1앞으로는 이동시켜도 의미가 없다는 것을 알 수 있다. 이와 같은 방법으로 각 노드마다 relocate할 때 앞 뒤로 범위를 설정하였다.

텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그 다음으로는, 범위를 설정한 다음 범위 내에서 relcate를 진행한다. 우선 먼저 relcate를 했을 때 거리가 줄어드는지를 확인한다. 만약 수정된 경로에서 거리가 줄어든다면 해당 경로에서 노드들의 time window가 어긋나는 것이 있는지를 확인한다. Time window까지 모두 만족한다면 최종적으로 경로를 업데이트하는 방식으로 solution을 improve하고자 하였다.

**Results**

1. **result of insertion heauristic**

Insertion heuristic알고리즘을 통해 구한 100개의 노드에 대한 solution은 다음과 같다.

텍스트, 도표, 패턴이(가) 표시된 사진

자동 생성된 설명

왼쪽은 결과를 csv파일 형태로 정리한 것이고, 오른쪽은 각 노드의 위치정보를 기반으로 노드간의 경로를 연결한 것이다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

다음으로 C1을 구할 때 가중치를 바꿔 위와 같은 결과값을 얻을 수 있었다. a1은 거리가중치에 대한 비례상수값이며, a2는 시간가중치에 대한 비례상수값이다. 따라서 위 결과를 보면, 거리 가중치를 두었을 때는 전체 이동거리는 짧아지는 대신 소요시간은 늘어나는 것을 확인할 수 있다. 반대로 시간에 가중치를 두었을 때는 전체이동거리는 늘어나지만, 전체소요시간은 줄어든 것을 확인할 수 있다. 앞서 instance의 특성상 시간과 거리를 같은 개념으로 본다고 했으나, 각 노드별로 일찍 도착한 경우 early time까지의 waiting time도 존재하므로, 실제 이동거리와 전체 소요시간은 다르게 나타날 수 있다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

또한 위 자료에서 왼쪽은 거리를 우선 고려했을 때 도출된 solution이며 오른쪽은 CPLEX를 이용하여 같은 instacne에 대해서 feasible한 solution을 구한 것이다. 두 결과를 비교해보았을 때 insertion heuristic으로 구한 결과가 전체 소요시간은 더 길지만, 전체 이동거리는 더 짧다는 것을 알 수 있다.

1. **result of 1-0 opt**

Insertion heuristic으로 구한 solution을 improve하고자 1-0 opt를 적용한 결과는 다음과 같다.

텍스트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

오른쪽은 노드의 time window를 벗어난 경우를 빨간색으로 표시해두었다. 이처럼 결과적으로 time window를 벗어나는 solution이 나오게 되었다. 이는 코드 구현과정에서 오류가 있었던 것으로 보여지며, 알고리즘 논리구조 상의 오류는 아닌 것으로 생각되어진다.

**Conclusion**

우선 1-0 opt에서 결과를 도출해내지 못하였으므로, insertion heuristic에 대해서 다루고자 한다. Insertion에서 어디에 가중치를 더 많이 부여하는가에 따라 결과가 달라지는 것을 확인할 수 있었다. 또한, 기존 계획은 insertion heuristic algorithm을 통해 구한 결과를 CPLEX를 통해 구한 결과와 비교하여 gap 차이를 통해서 insertion heuristic의 성능 평가 및 개선을 진행하려고 하였다. 하지만 insertion heuristic에서 거리 또는 시간에 어떻게 가중치를 두느냐에 따라 결과값이 다르게 나오며, 이러한 상황에서 CPLEX를 통해 구한 결과와 어떤 기준으로 gap을 정의해야 하는지에 대해서 명확한 기준이 필요하다고 생각된다. 또한 100개의 instance의 solution을 구하는데도 꽤 오랜 시간이 소요된다는 점에서 코드 최적화도 필요하다고 생각된다. 해당 알고리즘 구현에는 vector가 쓰였지만, 좀 더 연산에 최적화된 적절한 데이터 구조를 활용할 수 있을 것이라고 생각된다. 다음으로 초기경로 설정 과정에서도 노드 사이 거리가 가장 먼 2개의 노드를 초기경로로 설정하는 것이 아니라 다른 방법으로도 초기화하는 것이 가능할 것이라고 생각된다. 그 외에 나머지 반복문이나 조건문에서도 최적화가 가능할 것이다.

**Review**

우선 처음 시작할 때 6주라는 기간이 길다고 생각했지만, 실제로 연구참여를 하기에 매우 짧은 시간이라는 것을 느꼈다. 물론 방학기간 동안 계절학기 및 타 학생단체활동 때문에 바쁜 것도 있었지만, 논문을 읽는 것도 처음이었고 C++언어를 사용하는 것도 처음이라는 점에서 적응하는데 만해도 꽤 많은 시간을 소요한 것 같다고 생각이 든다. 2주에서 3주 정도 논문을 읽고 C++에 적응한 다음, 알고리즘을 구현하는 것도 꽤 오랜 시간이 걸렸다. 처음 작성하던 알고리즘에서 insertion으로 바뀌고 opt하는 알고리즘도 2-opt에서 M-N opt를 제시해주셔서 뒤늦게 열심히 참여하게 된것같아 매우 아쉬웠다. 조금 더 빨리 교수님, 사수님과 적극적으로 커뮤니케이션 했으면 더 좋은 결과물이 나왔을 거 같다는 생각이 들었다. “정보시스템기술”이라는 과목을 수강하면서 해당 연구분야에 흥미를 느껴 시작하게 된 연구참여였지만, 연구참여를 하면서 “정보시스템기술”에서 흥미를 느꼈던 부분은 해당 연구분야에서 매우 기본적인 내용이며, 실제 연구에서는 훨씬 더 많은 조건을 다뤄야한다는 점에서 스스로 조금 더 성장한 다음, 다시 연구참여에 도전해보고 싶다는 생각이 들었다.

**Reference**

MARIUSM. SOLOMO(1984) , “ALGORITHMS FOR THE VEHICLE ROUTING AND SCHEDULING PROBLEMS WITH TIME WINDOW CONSTRAINTS”