**2023 여름학기 연구참여 보고서**

**: K- means 알고리즘을 활용한 Balanced Graph Partitioning 문제의 개선 알고리즘 고안**

2023.06.26~2023.08.09 (7주)

Logistics Lab.

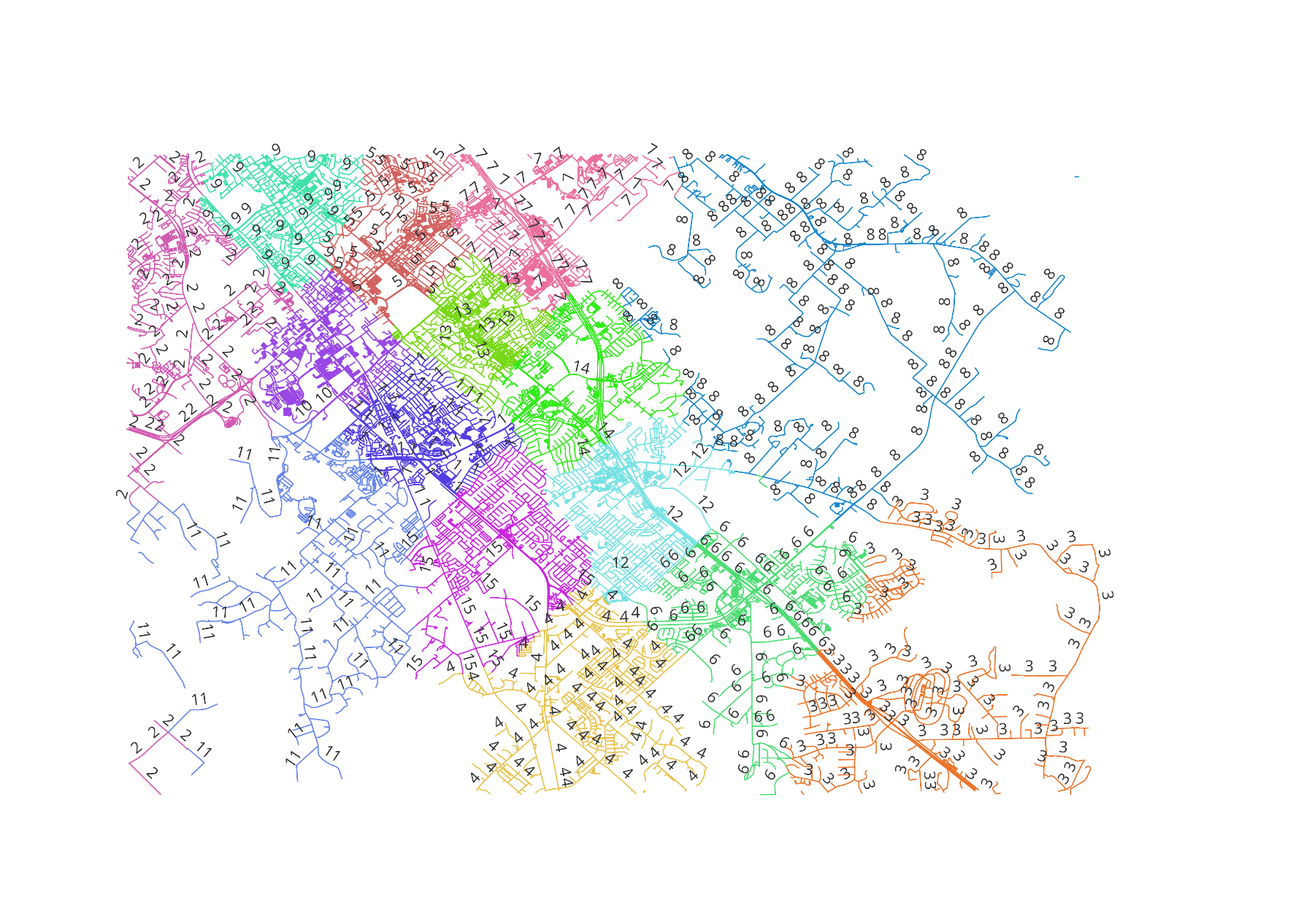
지도교수 김병인

멘토 이승엽

참여자 박성수

**목차**

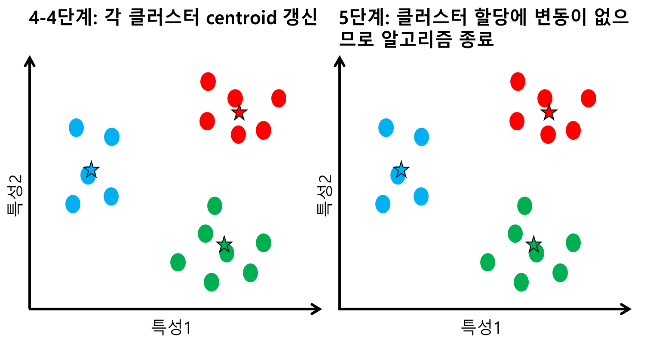
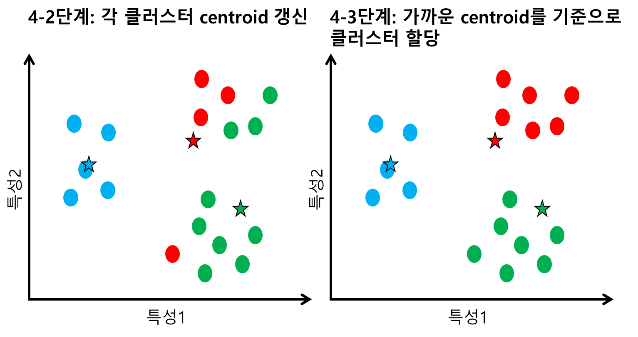
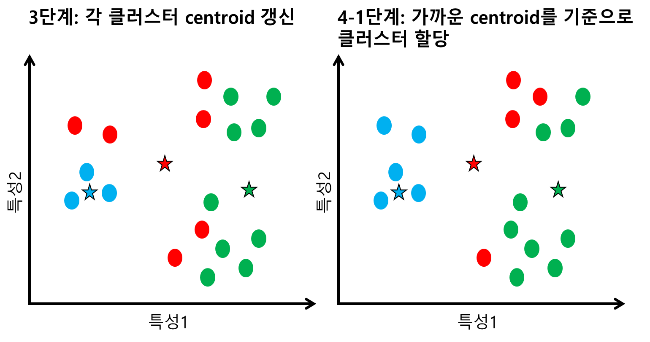
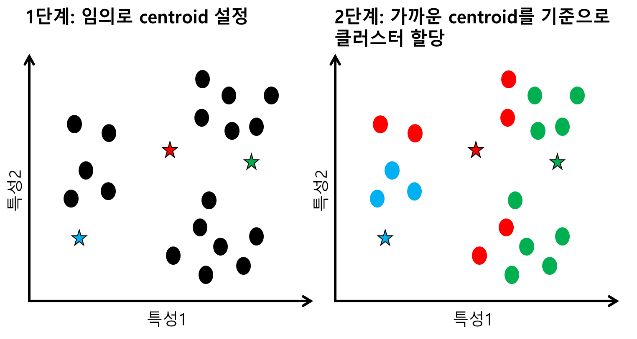
1. **연구참여 주제**
2. **알고리즘 개요**
3. **연구결과**
4. **결론**
5. **후기**
6. **연구참여 주제**



**그림 1 Balanced Graph Partition 문제 시각화**

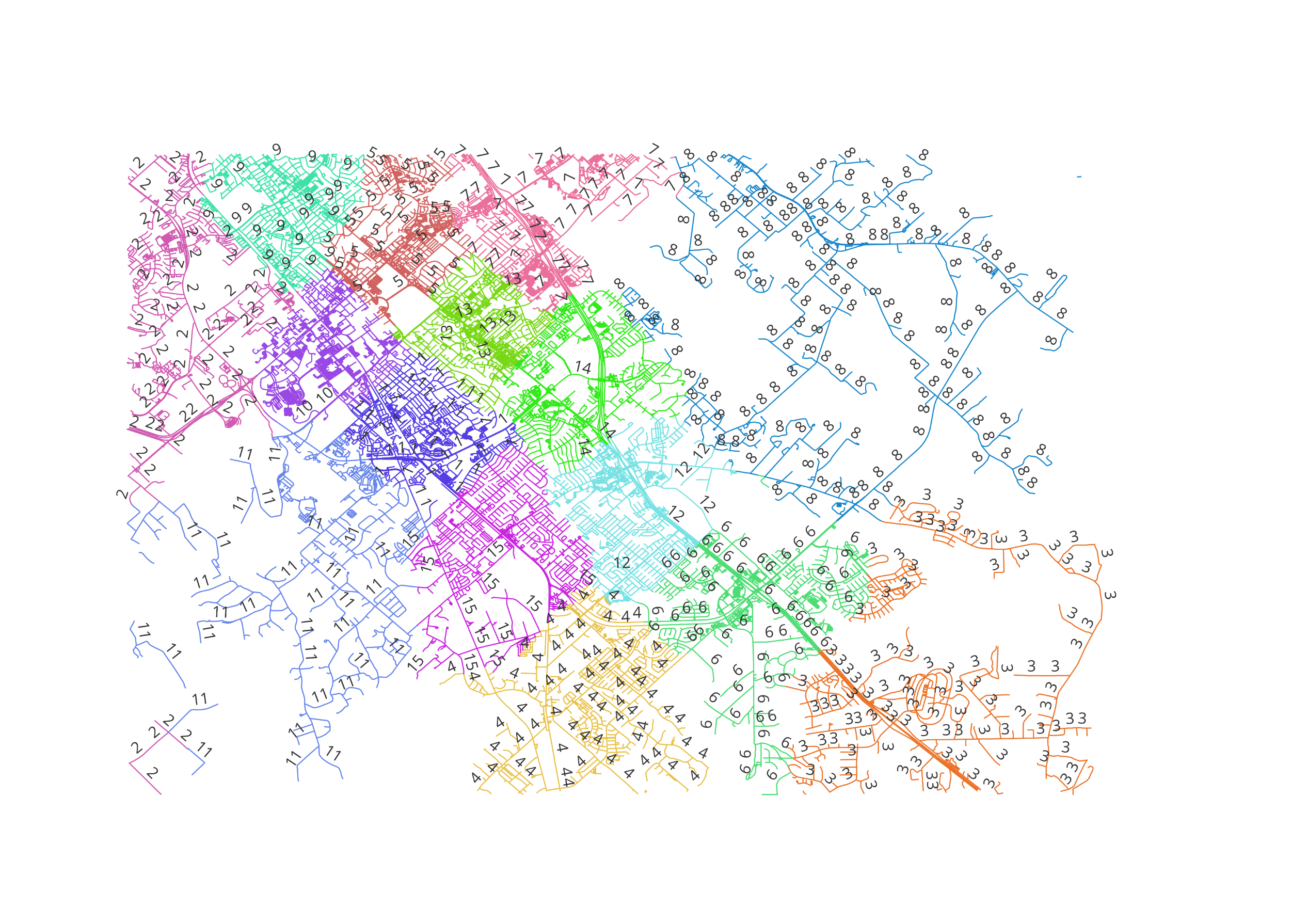
본 과제는 G = (N, A)가 있는 Graph가 있다고 할 때 (이때 N은 node set, A는 arc set이라 가정한다) Arc distance 합의 balance를 고려하여 A를 15개의 cluster로 나누는 문제이다. 15개로 나누는 이유는 3대의 청소차가 5일에 걸쳐 청소구역을 청소하는 문제이기 때문이다. Arc distance의 값을 고려할 때 단방향 Arc의 경우에는 거리 그대로 계산을 해주고 양방향 Arc일 경우에는 거리의 2배로 계산을 하였다.

기존에 Clustering을 할 때는 K-means 알고리즘을 사용하여 각 arc에 index를 배정하게 된다. 이때 K-means 알고리즘에서는 직선거리만 고려하여 배정을 하기 때문에 비효율적인 Clustering 결과가 발생하여 이를 해결하고자 하였다.



**그림 2 K-means 알고리즘 동작 방식**

K-means 알고리즘은 주어진 데이터를 k개의 클러스터로 묶는 알고리즘으로 각 클러스터와 centroid사이의 직선 거리를 최소화하는 방식으로 동작한다. 그림 2와 같이 임의로 centroid를 설정한 후 centroid 사이와의 직선 거리를 측정하여 cluster를 할당한다. Cluster를 할당한 이후 centroid를 갱신하게 되고 이 과정을 반복하다가 cluster 할당에 변동이 생기지 않는 경우 알고리즘을 종료하게 된다.



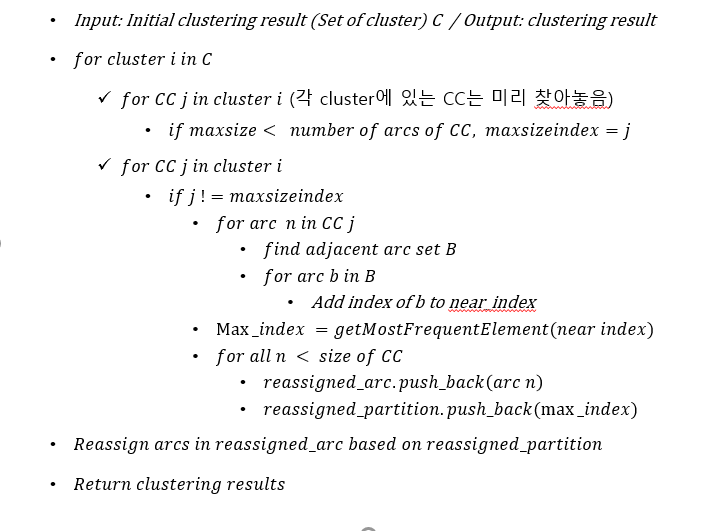
**그림 3 기존 알고리즘 문제 상황**

기존 알고리즘의 한계점을 개선하기 위해 새로 고안한 알고리즘에서는 connected component(CC)라는 개념을 사용하였다. CC는 한 index내에서 N과 A로 이루어져 있지만 서로 연결되어 있지 않은 것들 각각을 의미한다. 그림 3에서 3번 index를 보면 크게 연결되어 있는 부분과 6번 index와 연결되어 있는 부분을 확인할 수 있다. 이 각각의 요소들을 CC라고 할 수 있다.

그림 3과 같이 기존 알고리즘에서는 각 cluster의 centroid와 직선거리 만을 고려하여 CC의 index를 배정하게 된다. 이 경우에 clustering을 최적화하지 못해 청소차의 동선을 낭비하는 상황이 발생한다.

이와 같은 문제를 개선하기 위해 Index 내에 여러 CC들 중 제일 큰 CC를 제외한 나머지 CC들에 대해서 다른 index와 연결되어 있는 arc의 수를 계산하여 가장 많이 연결되어 있는 index로 배정하는 방식으로 개선을 할 것이다.

1. **알고리즘 개요**



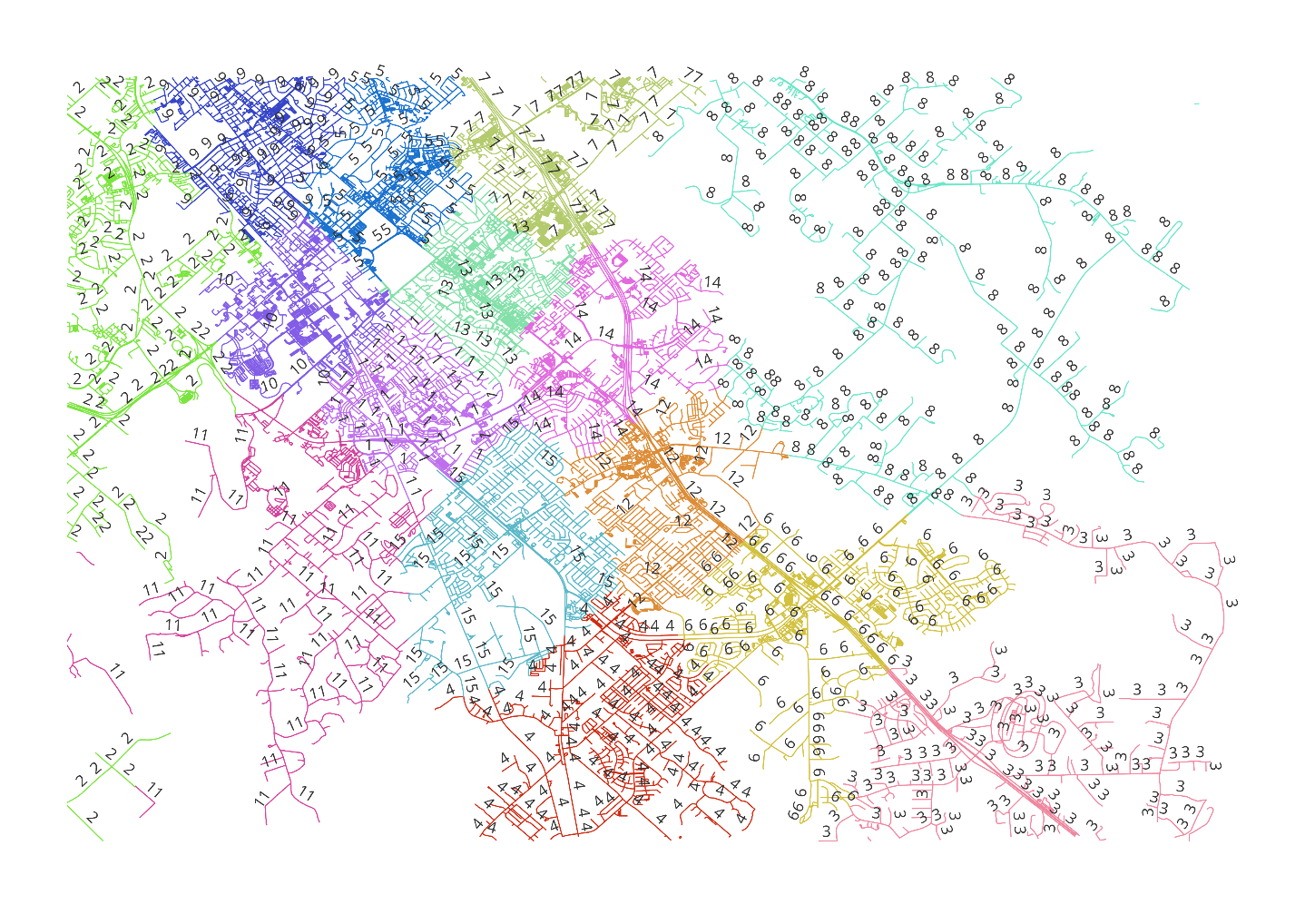
**그림 4 개선 알고리즘 의사코드**

그림 4의 의사코드는 문제상황을 개선한 알고리즘(Imp\_CC)의 의사코드이다. 기존에 있던 코드를 개선한 것이기 때문에 이미 존재하던 clustering 결과에서 새로 clustering 결과가 나오는 방식의 코드이다.

개선 알고리즘 의사코드의 시작 부분을 살펴보면 이미 찾아 놓았던 각 cluster의 CC에서 가장 큰 사이즈의 CC의 번호를 찾게 된다. Maxsize는 CC들의 크기를 의미하고 Maxsizeindex는 특정 cluster 내에서 가장 큰 CC의 번호를 의미한다.

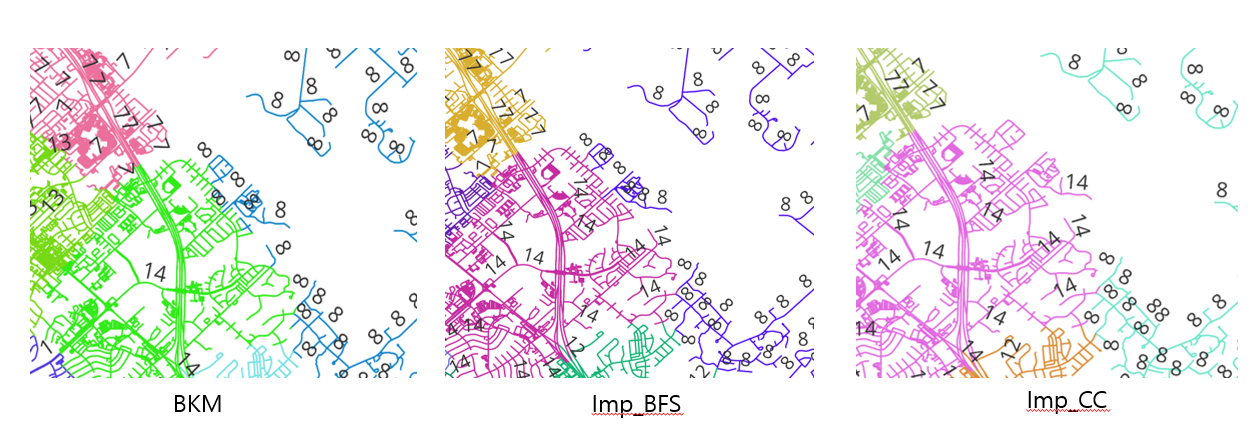
중간 부분을 살펴보면 가장 큰 CC를 제외하고 나머지 CC들에 대해 인접한 arc들 중 index가 다른 것들만 찾아 near\_index에 저장하게 된다. Max\_index는 near\_index에서 가장 많은 빈도의 index이고 getMostFrequentElement 함수는 입력 받은 배열의 정수 값 중 가장 많은 빈도의 정수 값을 반환해주는 함수이다.

마지막으로 Reassigned\_arc에 index 값 변화로 재배정되는 arc들을 저장해주고 reassigned\_partition에 재배정되는 index값을 저장해주면 기존에 코드들로 clustering 결과가 바뀌게 된다.

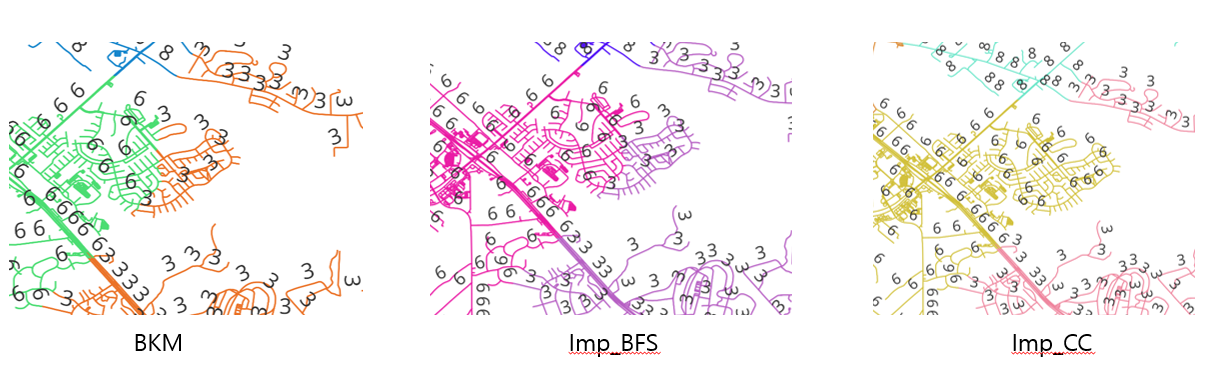
1. **연구결과**

**그림 5 Imp\_CC 시각화**

그림 5는 개선 알고리즘(Imp\_CC)을 적용한 뒤 시각화한 그림이다.

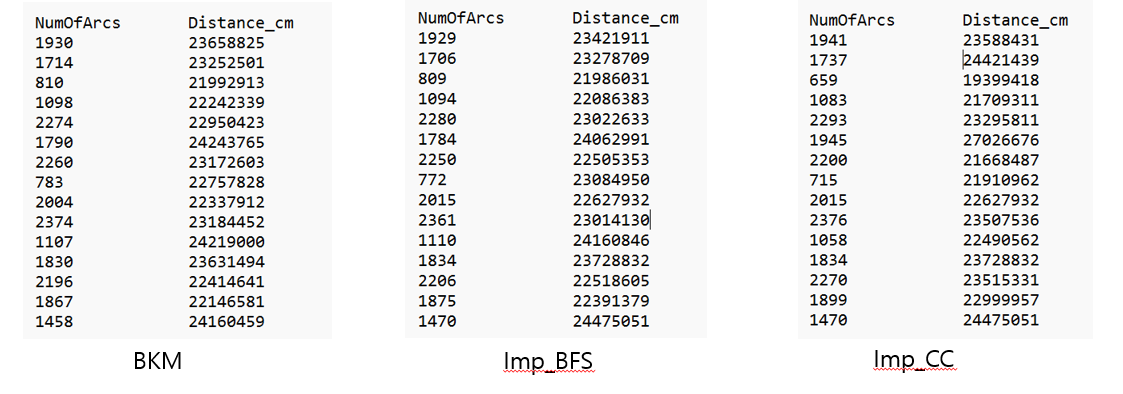


**그림 6 기존 알고리즘과 개선 알고리즘 결과 비교1**

****

**그림 7 기존 알고리즘과 개선 알고리즘 결과 비교2**

개선한 알고리즘의 결과와 비교한 결과 2가지로는 Balanced K-means 알고리즘만 사용한 결과(BKM)와 너비우선탐색 알고리즘까지 적용한 결과(Imp\_BFS)가 있다. 기존 알고리즘 결과를 시각화 하였을 때 개선해야할 부분으로는 그림 6의 8번 index와 14번 index, 그림 7의 3번 index와 6번 index가 있었다. 두 부분 모두 위 그림에서 확인할 수 있듯이 문제상황이 의도한 대로 개선이 되었음을 확인할 수 있다.

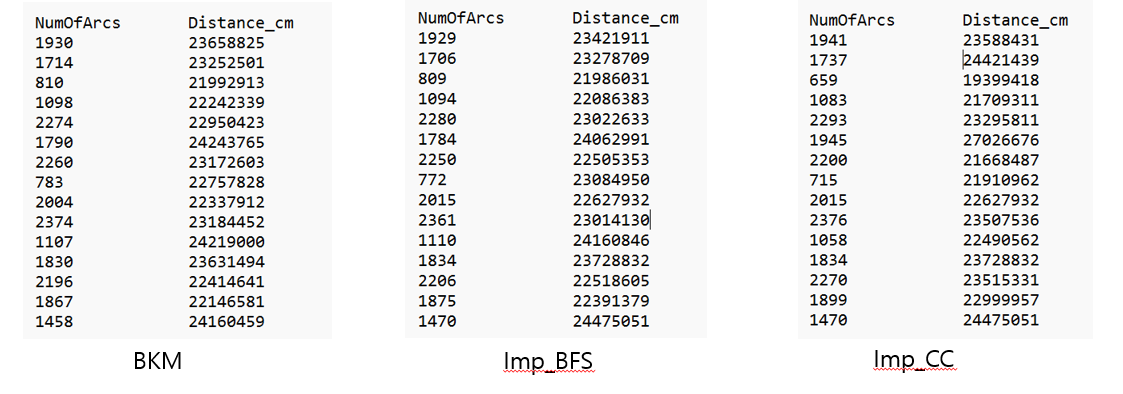


**그림 8 사용된 data값**

그림 8은 각각의 알고리즘을 적용한 후 확인할 수 있는 data값이다. 사용된 총 Arc의 개수는 25,495개이고 알고리즘의 결과마다 index별 Arc 개수, 각 index의 arc distance를 확인할 수 있다.

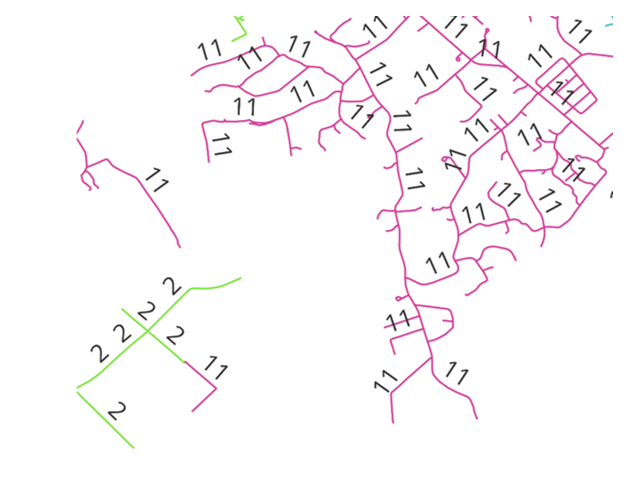
1. **결론**

위 연구결과에서 확인할 수 있듯이 문제 상황이라고 생각했던 2개의 부분에 대해서 의도했던 대로 개선이 되었음을 확인할 수 있다.



**그림 9 Imp\_CC의 data값**

다만 cluster간의 balance를 고려하지 못해 CC를 다른 index로 넘겨준 후 그만큼에 해당하는 Arc를 할당 받지 못해 cluster마다 arc distance 차이가 큰 것을 확인할 수 있다. CC를 넘겨준 후 용량에 맞춰 다른 arc를 할당 받는 코드를 구현한다면 해결할 수 있는 문제일 것 같다.



**그림 10 주변에 다른 index의 arc가 존재하지 않는 CC**

그림 10의 상황은 따로 배정 되어있는 2번 index의 CC가 실제로는 주변에 다른 index의 arc가 존재하지 않아 index가 바뀌지 않은 상황이다. 개선 알고리즘의 logic으로는 이와 같은 문제상황을 해결할 수 없으므로 특정 CC에 연결된 arc가 존재하지 않는 경우에는 기존 cluster의 centroid를 제외하고 가장 가까운 centroid의 cluster를 찾아 배정해주는 방식의 알고리즘이 필요할 것 같다.

1. **후기**

포스텍에 들어온 이후 학업을 놓은 학기도 있었고 나름대로 학업에 많은 관심을 쏟은 학기도 있었지만 대체적으로 성적이 좋지 않았습니다. 그로 인해 스스로에게 많은 실망을 느끼고 있었습니다. 이를 해결하기 위해선 나 자신에게 무언가를 성취하는 경험을 주어야한다고 생각하여 학업적으로 성취를 느낄 수 있는 연구참여를 이번 여름 학기에 해보기로 마음을 먹었습니다.

연구참여를 하기로 마음먹은 후 새내기 연구참여 때 여러 흥미로운 주제들을 접할 수 있었던 김병인 교수님의 랩이 가장 먼저 떠올랐고 늦게 컨택을 시도했음에도 김병인 교수님께서 흔쾌히 받아주셔서 연구참여를 시작할 수 있었습니다.

연구참여를 하면서 느꼈던 점 중 가장 컸던 것은 스스로에 대한 반성이었습니다. 호기롭게 시작한 첫 연구참여였지만 나 자신의 능력 부족으로 계획했던 타임라인이 계속 밀리고 이를 해결하기 위해 이승엽 사수님께 계속 의존하는 모습을 돌아보면서 객관적으로 나 자신이 많이 부족하다는 것을 깨달을 수 있었고 학업적으로 보다 많은 노력을 쏟아야겠다고 느꼈습니다.

또한 느꼈던 점은 어떤 분야에 대해 연구참여를 진행할 때 관련된 전공 과목을 듣거나 관련 지식들을 미리 알아가는 것이 필요하다는 것입니다. C++이라는 언어를 접해보지 못한 상태로 연구참여를 시작하고 연구참여 기간 동안 C++을 공부하고 응용하려 하니 힘든 점이 많았던 것 같습니다. 관련 과목을 듣고 연구참여를 진행했다면 연구과정이 보다 수월했겠다는 생각이 들었습니다.

우선 이번 연구 참여를 흔쾌히 허락해주시고 매주 미팅 때마다 격려해주시고 연구에 대해서도 계속 피드백 해 주신 김병인 교수님께 감사드립니다. 또한 연구참여 마무리 단계에서 헤매고 있던 저에게 가이드라인을 제시해주시고 알고리즘 구현 과정에서 많은 도움을 주시고 다소 귀찮을 수 있는 질문에도 친절히 답변해 주신 이승엽 사수님께도 감사드립니다.