Wavelet을 이용한 K-means clustering algorithm의 초기화

김국환, 장우진, 이준석

{lyn815, changw, akaSap}@snu.ac.kr
서울대학교 산업공학과
서울시 관악구 신림동 산56-1번지 서울대학교

Abstract

K-means clustering algorithm에서 주로 이루어지는 랜덤 초기화 (random initialization) 방법은 전역 최적화된 해(global minimum)를 찾아내기에 문제점을 지니고 있다. 즉, 여러 횟수의 알고리듬 반복(iteration)을 실행하더라도 전역 최적화된 해를 찾아내기가 매우 힘들며 주어진 자료의 크기(data size)가 큰 경우에 있어서 이는 거의 불가능하다. 본 논문은 이러한 문제점들을 극복하기 위한 방안으로, wavelet을 이용하여 최적의 초기 군집 중심점(initial clustering center)들을 선택하는 방법을 제시한다. 즉, 웨이블릿을 이용한 효과적인 초기화 (initialization)를 통해서 작은 알고리듬 반복 횟수만으로도 전역 최적화에 도달하는 초기화 방법을 기술한다. 이런 초기화 방법이 군집 알고리즘에 사용될 경우, 온라인상에서 실시간 이루어지는 군집 분석에 큰 도움이 된 수 있다.

Keyword : Clustering, Wavelets, K-means clustering algorithm

1.서론

1.1 기술 발전의 현황과 관련하여 결과값의 정확도의 중요성이 증가

최근 모 회사에서 웨이퍼에 구멍을 뚫는 새로운 방법을 사용하여 기존 제품보다 30% 가량 사이즈는 줄이면서도 성능은 30%이상 빨라진 반도체를 생산했다고 보도되었다. 갈수록 컴퓨터의 계산 처리 속도가 빨라지고 있기 때문에 computational time(실행할 때 걸리는 시간)보다 accuracy(정확도)에 중요성이 더해지고 있다. 일반적으로 computational time과 accuracy 사이에 trade-off가 존재한다. k-means clustering의 예를 들어보면 알고리즘을 한번 실행시키는 것보다 여러 번 실행시킬 때 좀 더 정확도가 높은 클러스터링 결과를 얻을 수가 있다.

1.2 문제 발생

데이터의 개수가 적고 feature의 수가 적은 경우, global optimum에 도달하는 것은 그리 어렵지 않다. 하지만 데이터의 개수와 feature 수가 늘어날수록 알고리즘을 실행할 경우 매번 local optimum값에 converge하여 아주 운이 좋지 않은 이상 한번에 global optimum에 도달하기 힘들어진다. 그리고 더 심각한 문제는 k-means를 여러 번 실행한다고 하더라도 정확도가 어느 수준 이상은 잘 개선되지 않는다는 점이다. 현재 나와 있는 초기화(initialization) 방법론이 대게 random selection을 쓰기 때문에 매번 실행할 때마다 random한 값으로 초기화되어 알고리즘이 돌아가기 때문에 매번 원점에서 다시 시작하는 것과 같다. 다만 매번 실행될 때, 더 좋은 클러스터링 결과가 나오면 current optimum cluster가 계속 업데이트되는 효과가 있지만 시간이 추가로 더 걸린다는 비용이 발생하고 이런 비용에 비하여 정확도 면에서 나오는 불만이 크다. 이를 극복하기 위한 노력으로 1998년 P.S. Bradley (University of Wisconsin, Madison)와 Usama M. Fayyad (Microsoft Research Center)가 "Refining Initial Points for K-Means Clustering"이라는 논문을 내놓았다. 이 논

문의 주된 관점은 일반적으로 k-means가 수많은 local minima중의 하나의 solution을 내놓는데, 반복적인 실행을 할 때 이전의 실행 후 결과 값들을 바탕으로 데이터의 분포를 추정하여 initial starting condition을 다시 정의하여 “better" local minimum을 얻겠다는 내용이다. 물론 이렇게 해서 실행하면 좋겠지만, 만약 분포가 굉장히 Uniform하게 나온다면 이 방법도 결코 해결책이 아니라는 것을 쉽게 알 수 있다. 이 논문에서 사용한 데이터를 살펴보면 2-D 이미지 데이터를 사용하여 분포가 2차원으로 쉽게 눈으로 확인할 수 있는 데이터이고, 그래서 feature의 차원이 높아지면 분포 추정하는 것이 매우 힘들어 진다.

1.3 연구의 목적

데이터 수도 많고, feature dimension도 큰 데이터를 온라인에서 실시간으로 클러스터링 해야 될 때, 최대한 빠른 시간 안에 정확하게 클러스터링 해야 하기 때문에, k-means 알고리즘을 무작정 여러 번 실행시킬 수는 없다. random subset selection을 하여 초기화하는 기존의 방법("kcentes"라 부름)으로 클러스터링을 한다고 가정하였을 때, 한 번을 실행하더라도 여러 번 실행했을 때보다 결과 값이 좋게 나오면 그 값이 "better" local minimum이 되는 것이다. 본 연구의 목적도 한번만에 좀 더 좋은 local minimum 값을 찾아내는 초기화 알고리즘을 만드는데 있고, 이 방법을 random selection에 의존하지 않는 deterministic한 방법으로 만드는데 초점을 두고 있다. 만약 새로운 초기화 알고리즘을 random하게 generating한다면 결국 비교하는 대상이나 비교 당하는 대상 둘 다 결과도 확률적으로 실행시킬 때마다 제 각각 변하기 때문에 비교하기가 굉장히 힘들어진다. 반면에 비교하는 알고리즘이 같은 데이터를 가지고 기본 설정(wavelet basis와 level 선택)을 동일하게 만들어 주면 항상 동일한 결과 값을 보여 준다면 비교가 더욱 객관적이고 쉬워질 것이다.

2. 본론

Delft University of Technology에서 만든 k-means algorithm을 기반으로 테스트를 해보았다. 먼저 이 알고리즘을 초기화하는 방법이 2가지가 있는데, 하나는 “rand"이고 다른 하나는 ”kcentres"이다. “rand"는 순전히 random하게 초기화한 것이고 "kcentres"는 다음과 같은 메커니즘으로 초기화가 작동된다. default는 kcentres로 설정되어 있고, rand보다 성능이 평균적으로 우수하게 나온다.

2.1 비교 대상이 되는 기존의 알고리즘 :

kcentres의 대략적인 알고리즘은 다음과 같다:

먼저 raw data를 symmetric한 distance matrix로 변환시킨다. 클러스터 개수를 k라 할 때M개의 데이터 중에서 random하게 k개를 선택하고 distance matrix를 참조하여 이 센터들의 distance를 계산한다. 이 작업을 N번 반복하면 k개의 센터 set이 N개 존재하게 되고 이 센터 candidate set들 중 모든 데이터 object들이 자신과 가장 가까운 센터와의 거리의 합을 최소화 하고 센터들 간의 거리를 최대화하는 최적의 센터 set을 return한다. 즉, 한마디로 요약하면 k개의 센터들을 N번 뽑아서 그 중 제일 나은 센터 set을 찾는 것이다.

2.2 wavelet을 이용한 초기화 방법

2.2.1 wavelet basis의 종류

다음과 같은 wavelet basis가 존재한다.

Wavelet Families	Wavelets
Daubechies	'db1' or 'haar', 'db2', ... ,'db10', ... , 'db45'
Coiflets	'coif1', ... , 'coif5'
Symlets	'sym2', ... , 'sym8', ... ,'sym45'
Discrete Meyer	'dmey'
Biorthogonal	'bior1.1', 'bior1.3', 'bior1.5' 'bior2.2', 'bior2.4', 'bior2.6', 'bior2.8' 'bior3.1', 'bior3.3', 'bior3.5', 'bior3.7' 'bior3.9', 'bior4.4', 'bior5.5', 'bior6.8'
Reverse Biorthogonal	'rbio1.1', 'rbio1.3', 'rbio1.5' 'rbio2.2', 'rbio2.4', 'rbio2.6', 'rbio2.8' 'rbio3.1', 'rbio3.3', 'rbio3.5', 'rbio3.7' 'rbio3.9', 'rbio4.4', 'rbio5.5', 'rbio6.8'

2.2.2 3가지 초기화 방법

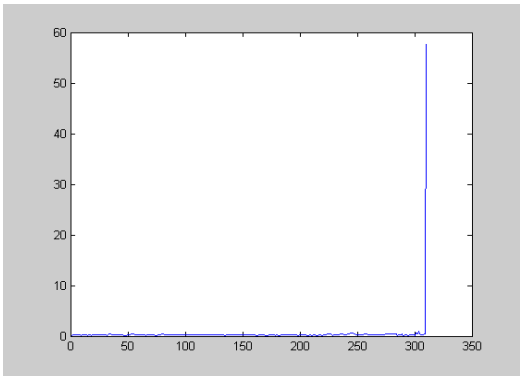
(Method 1) 먼저 raw data(A라고 하자)의 column별로 variance를 구한다음에 그 크기가 상위 50% 의 column만 떼어내어(이렇게 만들어진 매트릭스를 B라고 하자) 그것을 wavelet transform을 시켜서 wavelet data(C) 중에서 coarse level의 제일 앞 coefficient column을 kmeans에 집어 넣어서 돌려보았다. 이때 raw data A를 집어넣고 돌린게 아니라 C를 돌려보았다. (rbio4.4 이용 그리고 k=10 클러스터 개수를 10개로..)

(Method 2)kcentres로 초기화하여 kmeans로 돌려보았다.

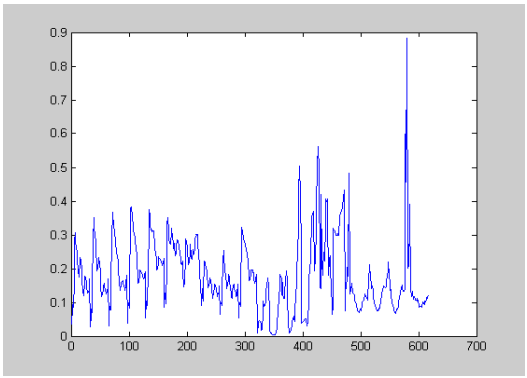
(Method 3) 먼저 raw data(A라고 하자)의 column별로 variance를 구한다음에 그 크기가 상위 50% 의 column만 떼어내어(이렇게 만들어진 매트릭스를 B라고 하자) kcentres를 이용하여 B를 kmeans에 넣고 돌려보았다.

2.3 사용한 데이터 설명

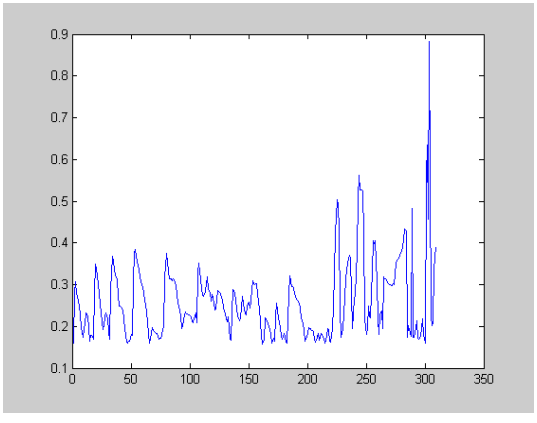
UCI Machine Learning Center에서 인터넷으로 제공하는 데이터 중에 데이터와 feature 수가 가장 큰 것을 사용하였다. (http://www.ics.uci.edu/~mlearn/MLRepository.html)
데이터의 이름은 “isolet5"이다.
다음은 feature들의 variance를 살펴보았다.



제일 마지막 feature의 scale이 나머지 feature 보다 현저히 커서 이것을 제외하고 다시 나머지 feature들에 대하여 variance를 plot해보았다.



이 중에서 variance를 크기 순으로 정렬하였을 때 상위 50% 안에 드는 feature들을 plot해보았다.



참고로 보기 편하게 제일 마지막 feature를

빼고 plot해봤는데, 실제 테스트를 할 때는 데이터를 있는 그대로 사용하였다. 물론 제일 마지막 column에 있는 숫자는 feature가 아니라 classification 후 인덱스로 1~26까지의 표시되어 있고, 이 column을 빼고 돌려보아도 결론적으로 말해서 본 연구에서 제시하는 Method 1이 Method 2나 Method 3보다 우수하게 나온다. 시간 나면 실제 Matlab code를 짜서 돌려보기 바란다.

2.3 basis의 선택

전체 종류는 6개이지만 각 level별로 또 다른 basis를 가지고 있기 때문에 다 합치면 수십 개가 된다. 이 중에서 어떤 basis를 사용해야 될지 고민이 된다. 저자가 여러 종류의 데이터를 서로 다른 basis를 사용하여 돌려본 결과 reverse biorthogonal wavelet을 사용하는게 가장 좋다고 판단하였고, 이 중에서도 rbio3.5, rbio4.4, rbio5.5, rbio6.8이 비슷하게 좋은 결과를 내놓았다. 하지만 어떤 basis가 절대적으로 항상 우수하다는 말은 하지 않고 넘어가겠다. 이 논문에서는 rbio4.4를 이용하였는데, 이 basis가 다른 basis에 비하여 항상 최적의 값을 제공한다는 말은 할 수 없으며 일단 기존의 알고리즘(kcentres)의 결과보다 나은 값을 도출해내는 것이 목적이기 때문에 크게 신경 쓰고 싶지 않다.

2.4 UCI 데이터 이용

본 연구에서는 distance를 측정하는 방법으로 Euclidean distance를 사용하였다. 다음은 UCI data 중 isolet5(1174x618)를 이용한 결과이다. 클러스터의 개수 k=10으로 하였다.

우선 rbio 4.4의 level 1,2만 보도록 하자. ([Method 1]과 [Method 2]를 비교하여 보았다.)

	rbio4.4	
level	distance	cpu time
1	1.3866e+004	2.2810
2	1.3802e+004	2.1720

위 표는 앞에서 언급한 바와 같이 rbio4.4로 실행시켜 level 1과 level 2의 결과이다. 위 표는 Method 2의 방법으로 연속해서 8번 실행시킨 후 얻은 결과이다. 정확도 면에서 Method 1이 Method 2보다 우수함을 알 수 있다. (distance는 작을 수록 좋다.)

trial	distance	cpu time
1	1.4910e+004	1.7190
2	1.4148e+004	3.5940
3	1.4297e+004	1.7350
4	1.4259e+004	1.9850
5	1.4334e+004	2.8750
6	1.4840e+004	2.6560
7	1.4291e+004	2.4380
8	1.5182e+004	2.4370

참고로 rbio의 다른 basis에 대해서도 level 1과 level 2의 결과를 잠시 살펴보고 넘어가자. 전부 표로 나타내지는 못했고 rbio1.1, rbio1.3, rbio1.5, rbio2.2, rbio2.4 이렇게 앞에서 5개만 살펴보다.

	rbio1.1	
level	distance	cpu time
1	1.4266e+004	1.9220
2	1.3866e+004	1.4530

	rbio1.3	
level	distance	cpu time
1	1.3866e+004	1.5310
2	1.3821e+004	1.1880

	rbio1.5	
level	distance	cpu time
1	1.3870e+004	1.3440
2	1.3823e+004	2.2810

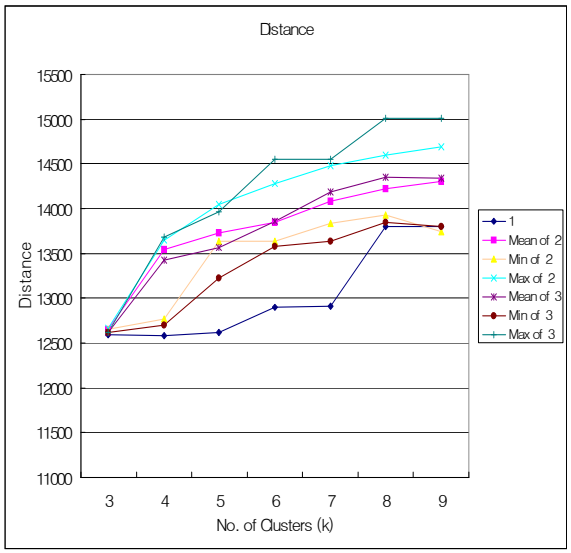
	rbio2.2	
level	distance	cpu time
1	1.3866e+004	1.3280
2	1.3802e+004	1.9840

	rbio2.4	
level	distance	cpu time
1	1.3870e+004	1.5310
2	1.3802e+004	1.1880

distance를 살펴보면 Method 2의 결과보다 다들 좋게 나온다.

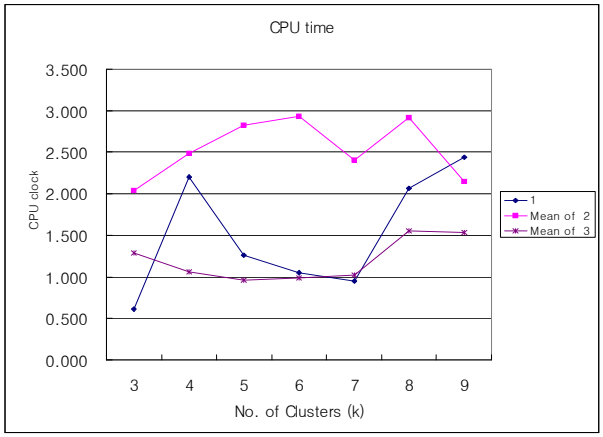
2.4 결과 분석

distance와 cpu time을 Method 1 (rbio4.4의 level 3)을 기준으로 다른 두 가지 방법으로부터 얻어진 결과의 Mean, Min, Max값들과 비교하여 아래와 같이 그림으로 나타내었다.



Method 1이 제일 좋은 performance를 보여주고 그 다음으로는 Method 3 그리고 Method 2가 가장 성능이 떨어진다.

다음은 cpu time을 서로 비교하였는데, Method 2와 Method 3에서 실행 시간이 짧다고 해서 가장 좋은 결과(제일 작은 distance)값을 가진다고 말할 수는 없다. 오히려 빨리 converge할 때 우리가 추구하는 “better" local min에서 동떨어진 결과를 얻을 수가 있다. 그래서 Min, Max 값을 제외하고 나머지 두 Method들에 관해서 Mean 값을 plot하여 아래와 같이 나타내었다.



일반적으로 Method 2 보다 Method 1의 cpu time이 더 짧다. 셋 중에서 Method 3가 제일 실행시간이 짧은데, 앞서 봤던 distance 그래프와 같이 빨리 converge한다고 해서 distance 값이 작은 것은 아니다. 단지 Method 3가 Method 1보다 빠르다는 것을 말해준다. 전체적으로 저자가 하고 싶은 말은 Method 1이 Method 2보다 빨리 실행되면서 더 정확한 값을 구해준다는 것이고 여기서 Method 1의 cpu time을 계산할 때 raw data를 wavelet으로 transform하는 시간을 제외시켜주었는데, 그 justification을 다음 섹션에서 설명하겠다.

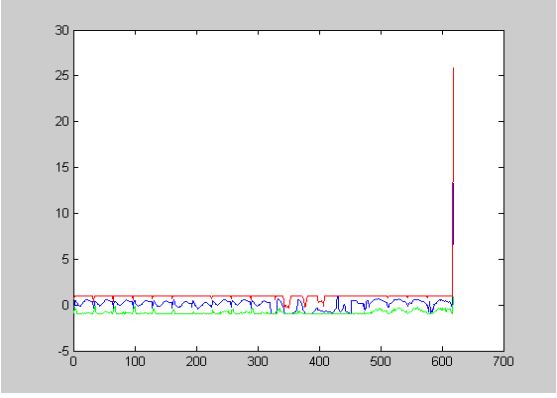
2.5 cpu time에 대한 justification

본 연구에서 사용된 데이터인 isolet5를 wavelet transform 시키는데 걸리는 시간은 4~9초이다. (그 정도는 실행하는 컴퓨터의 환경에 따라 차이가 날 것이다.)
kcentres로 초기화된 Method 2를 2번 실행시킬 시간에 wavelet을 이용한 Method 1을 사용하는 편이 더 좋을 이유가 justification의 첫 번째 사안이고, 둘째로 만약 온라인에서 데이터를 얻는다면, 데이터를 얻을 때, 각 feature의 variance를 고

려하여 얼마든지 k-means clustering algorithm을 실행시키기 전에 wavelet transform을 시킬 수 있다는 것이 두 번째 사안이다. 만약 시간이 무한대로 주어져 있다면 굳이 이런 방법을 쓰지 않고도 얼마든지 정교하게 clustering을 할 수 있다. 하지만, 본 연구에서의 가정은 시간이 제한되어 있다는 것이고, 제한된 시간 안에 최대한 빨리 converge하면서 좀 더 정확한 clustering을 추구하고자 하는데 의미를 두기 때문에 상황을 굳이 말로 표현하자면 온라인에서 실시간으로 feature 수가 큰 데이터를 clustering한다고 생각하면 이해하기 편할 것 같다. 일단 실시간으로 다량의 데이터가 들어오는데, 데이터가 들어오는 즉시 wavelet transform을 시켜주고 초기화를 위해 이용해야 할 coarse level의 제일 앞에서 있는 coefficient만 저장시키고 나머지는 버리는 식으로 wavelet transform 시간을 줄이는 것이 충분히 가능한 일이다. 이러한 coarse level coefficient를 저장하면서 가지고 있게 되면 매번 clustering을 할 때마다 random하게 초기화하지 않고 이 column vector를 uniform하게 잘라서 k-means algorithm에 넣어버리면 초기화 시간을 최대한 단축시킬 수 있는 장점이 있다.

2.6 사용한 데이터에 대한 justification

데이터의 평균 값 및 최소, 최대 값을 plot 해보면 다음과 같다.



거의 모든 feature의 값들이 -2와 2 사이에 적당히 Uniform하게 분포되어 있다. 만약 특정 몇몇 feature의 절대 값이 극단적으로 크다면 clustering의 초기화를 할 때, 이에 해당되는 상위 p%만 선택하여 k-means를 실행시키면, 만약 이 데이터처럼 그러한 feature의 수가 하나라면, histogram을 이용하여 크기 순서대로 Uniform하게 잘라서 초기화한 방법과 비슷한 효과를 볼 수 있으므로 Method 1의 초기화의 과정인 feature selection에 이미 포함된다고 말 할 수 있다. 그리고 본인은 실제 제일 마지막 feature 하나만 제외하고 위에 주어진 똑같은 방식으로 k-means를 실행시켜보았는데, 제일 마지막 feature를 고려한 것과 제외시킨 것들 사이에 큰 차이는 발견되지 않았다. 오히려 Uniform에 가까운 데이터일수록 Method 1의 정확도가 Method 2의 정확도보다 상대적으로 더 나아진다는 사실을 발견하였다.

2.7 future work

"Refining Initial Points for K-Means Clustering"의 내용처럼 K-Means Algorithm을 2번 이상 반복해서 실행시킬 때 어떤 방식으로 초기화를 해야 하는지 본 연구에서 사용한 방법을 토대로 기존 연구에 대해 개선책을 찾도록 해야 할 것이다.

3. 결론

본 연구는 K-Means algorithm의 초기화를 wavelet을 활용하여 좀 더 정확한 clustering 결과를 도출하였다는 점에서 그 의의를 두고 있다.

<Reference>

P. Bradley,Usama Fayyad, Refining Initial Points for K-Means Clustering, 1998

(그림 1) rbio4.4 사용

k=3	Method 1	
trial	distance	cpu time
1	1.2598e+004	1.5790
2	1.2613e+004	0.8120
3	1.2598e+004	0.6090
4	1.2598e+004	0.5000
5	1.2598e+004	2.1870
6	1.2579e+004	0.9850
7	1.2598e+004	1.7500
8	1.2598e+004	1.9220
9		

k=3	Method 2	
trial	distance	cpu time
1	1.2648e+004	1.9380
2	1.2648e+004	3.0620
3	1.2664e+004	2.4690
4	1.2664e+004	1.3900
5	1.2648e+004	1.0320
6	1.2664e+004	2.2820
7	1.2664e+004	2.4840
8	1.2664e+004	1.5780
9	1.2648e+004	2.0930

k=3	Method 3	
trial	distance	cpu time
1	1.2614e+004	0.6410
2	1.2614e+004	0.7340
3	1.2614e+004	2.6250
4	1.2614e+004	0.7350
5	1.2614e+004	2.7180
6	1.2614e+004	0.6100
7	1.2614e+004	0.7340
8	1.2614e+004	0.7350
9	1.2614e+004	2.0940

k=4	Method 1	
trial	distance	cpu time
1	1.3685e+004	0.8440
2	1.2579e+004	0.6090
3	1.2584e+004	2.2030
4	1.2613e+004	1.3590
5	1.2613e+004	1
6	1.2613e+004	0.9060
7	1.2579e+004	1.0630
8	1.2579e+004	1.2030
9		

k=4	Method 2	
trial	distance	cpu time
1	1.3643e+004	3.2500
2	1.3643e+004	3.9060
3	1.3643e+004	4.1410
4	1.3643e+004	1.0470
5	1.3643e+004	2.7180
6	1.2771e+004	2.0470
7	1.3643e+004	1.6250
8	1.3643e+004	1.4530
9	1.3643e+004	2.2030

k=4	Method 3	
trial	distance	cpu time
1	1.3687e+004	0.9840
2	1.3326e+004	0.5470
3	1.3687e+004	0.5310
4	1.3687e+004	1.3280
5	1.3687e+004	1.9060
6	1.2701e+004	0.7500
7	1.3687e+004	1.6720
8	1.3687e+004	1.2190
9	1.2701e+004	0.6250

k=5	Method 1	
trial	distance	cpu time
1	1.3685e+004	0.8750
2	1.3516e+004	1.0160
3	1.2613e+004	1.2650
4	1.2612e+004	0.7810
5	1.2598e+004	1.3440
6	1.2598e+004	0.6720
7	1.2598e+004	0.6880
8	1.2598e+004	0.5630
9		

k=5	Method 2	
trial	distance	cpu time
1	1.3754e+004	1.7500
2	1.3906e+004	2.9840
3	1.3641e+004	1.8280
4	1.3641e+004	4.3750
5	1.3641e+004	5.4370
6	1.3641e+004	1.6250
7	1.3641e+004	2.5780
8	1.4047e+004	2.4060
9	1.3702e+004	2.4070

k=5	Method 3	
trial	distance	cpu time
1	1.3929e+004	1.7350
2	1.3576e+004	0.7820
3	1.3689e+004	1.2660
4	1.3576e+004	1.6250
5	1.3576e+004	0.6400
6	1.3576e+004	0.4220
7	1.3576e+004	0.7970
8	1.3576e+004	0.7810
9	1.3964e+004	1.2810

k=6	Method 1	
trial	distance	cpu time
1	1.3802e+004	2.0620
2	1.3516e+004	0.9380
3	1.2901e+004	1.0470
4	1.3521e+004	1.7660
5	1.3521e+004	2.4530
6	1.3516e+004	1.7960
7	1.3523e+004	1.7030
8	1.3546e+004	1.4380
9		

k=6	Method 2	
trial	distance	cpu time
1	1.3906e+004	2.5780
2	1.3699e+004	3.5470
3	1.3641e+004	4.7190
4	1.3906e+004	3.3290
5	1.3702e+004	2.3600
6	1.3906e+004	2.3120
7	1.3641e+004	2.1250
8	1.3906e+004	2.5310
9	1.4278e+004	2.9380

k=6	Method 3	
trial	distance	cpu time
1	1.3991e+004	0.9530
2	1.3584e+004	0.7040
3	1.3635e+004	0.7030
4	1.3810e+004	1.4220
5	1.4547e+004	1.0470
6	1.4164e+004	1.3130
7	1.3635e+004	0.9530
8	1.3576e+004	0.5630
9	1.3802e+004	1.2190

k=7	Method 1	
trial	distance	cpu time
1	1.3802e+004	1.4380
2	1.3523e+004	2.1090
3	1.2914e+004	0.9530
4	1.3533e+004	0.9690
5	1.3520e+004	0.5620
6	1.3520e+004	0.7190
7	1.3516e+004	1.9060
8	1.3516e+004	1.6250
9		

k=7	Method 2	
trial	distance	cpu time
1	1.4277e+004	1.3600
2	1.3833e+004	3.1720
3	1.3876e+004	4.4840
4	1.4090e+004	2.3900
5	1.4207e+004	1.5630
6	1.3932e+004	1.7500
7	1.4480e+004	2.1400
8	1.3910e+004	2.1720
9	1.4090e+004	2.5780

k=7	Method 3	
trial	distance	cpu time
1	1.4101e+004	0.5940
2	1.4220e+004	0.7180
3	1.4220e+004	0.9680
4	1.4370e+004	0.9680
5	1.3641e+004	0.7180
6	1.4164e+004	1.2030
7	1.4547e+004	1.2030
8	1.4547e+004	0.8280
9	1.3849e+004	1.9840

k=8	Method 1	
level	distance	cpu time
level 1	1.3866e+004	1.1100
level 2	1.3821e+004	2.0470
level 3	1.3802e+004	2.0630
level 4	1.3576e+004	0.9840
level 5	1.3521e+004	1.2810
level 6	1.4148e+004	2.0310
level 7	1.4038e+004	2.0630
level 8	1.4075e+004	2.3750
level 9		

k=8	Method 2	
trial	distance	cpu time
1	1.4047e+004	3.2660
2	1.4338e+004	2.8280
3	1.4602e+004	2.3910
4	1.4051e+004	4.7350
5	1.3932e+004	1.1560
6	1.4437e+004	4.4530
7	1.4329e+004	2.3900
8	1.4182e+004	2.0120
9	1.4133e+004	3.0320

k=8	Method 3	
trial	distance	cpu time
1	1.4090e+004	1.4220
2	1.4194e+004	0.8590
3	1.4164e+004	1.0210
4	1.4386e+004	1.8750
5	1.5009e+004	2.5470
6	1.3849e+004	1.1250
7	1.4613e+004	2.0160
8	1.4066e+004	2.4220
9	1.4746e+004	0.7340

k=9	Method 1	
trial	distance	cpu time
1	1.3834e+004	1.2810
2	1.3523e+004	1
3	1.3802e+004	2.4370
4	1.3526e+004	1.3280
5	1.4148e+004	1.6560
6	1.4146e+004	1.6560
7	1.4100e+004	2.2810
8	1.4077e+004	2.3130
9		

k=9	Method 2	
trial	distance	cpu time
1	1.4323e+004	3.6870
2	1.4697e+004	1.6090
3	1.3740e+004	3.8120
4	1.4359e+004	1.6090
5	1.4232e+004	1.4070
6	1.4277e+004	1.6560
7	1.4342e+004	1.4070
8	1.4520e+004	2.0470
9	1.4298e+004	2.0790

k=9	Method 3	
trial	distance	cpu time
1	1.5006e+004	1.7340
2	1.4526e+004	1.0940
3	1.4185e+004	1.4840
4	1.4082e+004	1.1720
5	1.4280e+004	1.3130
6	1.3806e+004	0.7660
7	1.3982e+004	1.1560
8	1.4739e+004	1.9850
9	1.4424e+004	3.1400

(Method 1에서 왼쪽에 trial 1~9로 나와 있는데, 잘못 표기되었고 level 1 ~ level 8로 정정되어야 된다. 예, k=8 일 때 표 참조)