Gebze Technical University
Computer Engineering

CSE424 – 2019

HOMEWORK REPORT

Serkan Sorman
151044057

1) Brute Force

Yarı çapları liste şeklinde verilen çemberlerin minimum width ile 2D bir box içerisine yerleştirilmesi için çemberlerin hangi sırada dizili olması gerektiği problemine çözüm yöntemlerinden biridir.

Argüman olarak verilen çember listesinin tüm permütasyonları oluşturulur. Tüm bu permütasyonların sahip olduğu width hesaplanır ve birbirleri ile karşılaştırılarak minimum widthe sahip circle sırası elde edilir. Tüm permütasyonları oluşturma işlemi O(n!) zaman alır burada n circle sayısıdır.

2)Branch & Bound

TSP de kullanılan B&B çözümü Circle Packing problemine uygulanmıştır. TSP probleminde başlangıç noktası sabittir ve başlangıç noktasına tekrar dönülmesi gerekir fakat circle packing probleminde optimal çözüme göre başlangıç noktası değişebilir ve bu noktaya geri dönüş gibi bir durum söz konusu değildir. Bu sebeple input olarak circle yarıçapları listesindeki her bir circle ilk eleman olacak şekilde ayarlanır ve TSP problemindeki B&B çözümü bu listelere uygulanır. Böylece her branchdeki local optimal çözümler elde edilir ve karşılaştırılarak global optimal çözüme ulaşılır. Worst case time complexitysi Brute-force ile aynıdır yani O(n!) zaman alır burada n circle sayısıdır.

3) Greedy

Brute Force yaklaşımı sonucunda elde edilen optimal circle sıralamalarında, ard arda gelen circleların yarıçapları arasındaki farkın maksimum olduğu görülmüştür. Buradan yapılan çıkarım sonucu ilk olarak alınan circle listesi sıralanmıştır. Ardından minimum yarıçapa sahip circle yeni oluşturulan listenin ortasına set edilmiştir. Eğer var ise bir önceki indexine en büyük circle ve bir sonraki indexine ikinci en büyük circle set edilmiştir. Daha sonra listedeki tüm elemanlar en küçük ve en büyük elemanlar yan yana gelecek şekilde listeye eklenmiştir. Javada kullanılan Array.sort() referanslar için Merge Sort kullandığından algoritmanın time complexitysi O(nlogn) dir.

4) Iterated Local Search

Aşağıdaki pseudo code circle packing problemine uygulanmıştır.

```
procedure Iterated-Local-Search(n_max : IN) : S;
    begin
        Create starting solution s;
        s := Local-Search(s);
        n := 0;
        repeat
            s' := Mutate(s);
            s' := Local-Search(s');
            if f(s') < f(s) then s := s';
            n := n + 1;
        until n = n_max;
        return s;
    end;
```

Başlangıç olarak initial bir circle listesi oluşturulup width hesaplanır. Local search için argüman olarak alınan populasyon sayısı kadar neighborhood oluşturulup neighborhood içerisinden local optima elde edilir. Initial solution mutate edilerek yeni circle sıralaması elde edilir. Yapılan tüm bu işlemler iterasyon sayısı kadar tekrar edilir ve en az width değerini sağlayan çözüm elde edilmeye çalışılır. Ayrıca argüman olarak alınan Target değere ulaşılırsa döngü sonlandırılır. Time complexity O(i*n) dir. Burada i arguman olarak alınan iterasyon sayısı ve n populasyon boyutudur.

5) Simulated Annealing

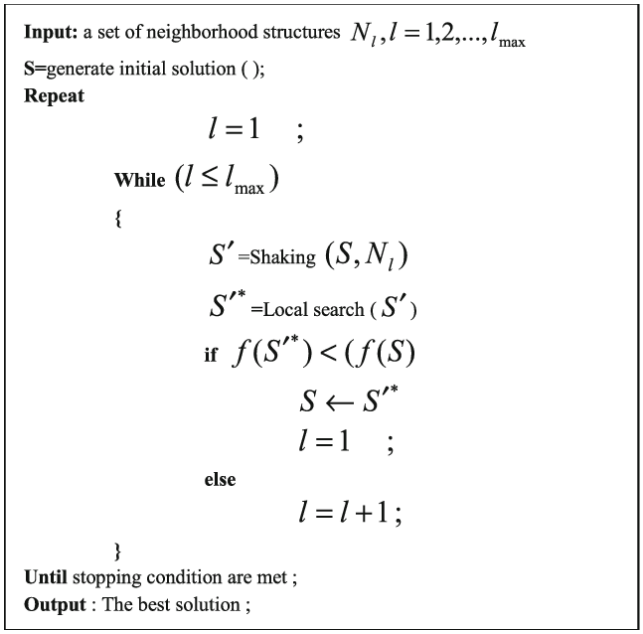
Aşağıdaki pseudo code circle packing problemine uygulanmıştır.

Başlangıç sıcaklık değeri olan T 1.0 olarak set edilmiştir. Sonrasında T değeri her iterasyonda 0.9 oranında azaltılmıştır. İstenen sıcaklık olan 0.001 e ulaşılana kadar bu işlem devam edilmiştir. İçteki döngü argüman olarak alınan iterasyon sayısına göre loop edilmiştir. Daha küçük widthi sağlayan solution güncel solutionun yerini almıştır. Ayrıca belirtilen olasılık sağlanırsa da aynı işlem gerçekleştirilmiştir. Böylece seçilen neighborlar ile current solution karşılaştırılarak optimal çözüme ulaşılmaya çalışılmıştır. Argüman olarak alınan Target değere ulaşılırsa döngü sonlandırılır. Time complexitysi O(i*p * logn) dir. Sıcaklık logn olarak azalırken local search populasyon boyutu olan p kadar ve iç döngü iterasyon sayısı olan i kadar döner.

1: Generate initial configuration $\mathbf{z}^{(0)}$ and initial temperature T_0.
2: **while** convergence criteria are not met **do**
3: Fix temperature $T_k = \text{annealingSchedule}(T_{k-1})$.
4: **for** $l = 1, \ldots, L$ **do**
5: Choose randomly an element $\mathbf{z}_l \in \mathcal{N}(\mathbf{z}^{(k)})$.
6: If $f(\mathbf{z}_l) < f(\mathbf{z}^{(k)})$, then $\mathbf{z}^{(k)} \leftarrow \mathbf{z}_l$.
7: Else, with probability $P_{\text{accept}}(\mathbf{z}_l, \mathbf{z}^{(k)}; T_k)$, then $\mathbf{z}^{(k)} \leftarrow \mathbf{z}_l$
8: **end for**
9: **end while**
10:
11: **return** the best solution found.

6) VNS

Aşağıdaki pseudo code circle packing problemine uygulanmıştır.



Shaking işleminde, neighborhoodlar arasından random olarak index l de bulunan solution alınır. Initial solutionla karşılaştırılarak initial solutionun yerini alır. Daha sonra bu solutiona neighborhood içerisinde local search yapılarak local optima elde edilir. Daha küçük widthe sahip solutionlar current solutionun yerini alarak index eski haline getirilir ve devam edilir. Aksi durumda iterasyona devam edilir. Argüman olarak alınan Target değere ulaşılırsa döngü sonlandırılır. Time complexitysi O(i * kmax * p) dür. Burada i arguman olarak alınan iterasyon sayısıdır. Local search populasyon boyutu olan p, dış döngü iterasyon sayısı olan i ve iç döngü kmax değeri kadar döner.

7)Particle Swarm Optimization

Aşağıdaki pseudo code circle packing problemine uygulanmıştır.

```
FOR each particle i
    FOR each dimension d
      Initialize position x_id randomly within permissible range
      Initialize velocity v_id randomly within permissible range
    End FOR
END FOR
Iteration k=1
DO
   FOR each particle i
       Calculate fitness value
       IF the fitness value is better than p_best_id in history
         Set current fitness value as the p_best_id
      END IF
   END FOR
   Choose the particle having the best fitness value as the g_best_d
   FOR each particle i
       FOR each dimension d
                Calculate velocity according to the equation
                v_id(k+1)= w v_id(k)+c1 rand1(p_id-x_id)+c2 rand2 (p_gd-x_id)
                Update particle position according to the equation
            x_id(k+1)= x_id(k)+v_id(k+1)
       END FOR
   END FOR
k=k+1
WHILE maximum iterations or minimum error criteria are not attained
```

Başlangıçta random solution içeren particlelar oluşturulmuştur. Algoritmada belirtilen formüllere göre her particleın velocitysi ve positionu update edilir. Positionu fazla olan particle sürüden aykırı hareket ediyordur ve değiştirilmesi gerekir. Positionu kadar loop edilerek random indexleri swap edilir. Ana döngüde her particleın şu ana kadar sahip olduğu en optimal çözüm yani local besti (pBest) update edilir. Aynı şekilde tüm particleların pBestleri karışılaştırılarak global optima elde edilir

8)Ant Colony Optimization

İlk olarak farklı solutionlara sahip karıncalar oluşturulur. Solutionların fitness değerlerine göre feromon tablosundaki pathler update edilir. Feromon miktarı fazla olan path yani solution gBest olarak kabul edilir. %80 oranla karıncaların her biri bu pathe yönlendirilir yani solutionlarına gBest set edilir. %20 ihtimalle ise karınca farklı bir pathe yönlendirilir. Tüm karıncaların feromon miktarı diğer pathlere göre çok olan pathe sürekli yönelmemesi için evoporate oranına göre her iterasyonda feromon değerleri belli bir oranda evoporate edilir. Feromon miktarı fazla olan pathe göre gBest update edilir ve optimal çözüme ulaşılmaya çalışılır.

Farklı Size ve Range için yapılan Testler ve sonuçları

Test 1)

```
Double [] circles = {99.0,5.0,2.0,2.0,500.0,650.0,4.0,100.0,121.0,987.0,7.0};
showExecutionTime(new BruteForce( TARGET: 913.7),circles);
showExecutionTime(new Greedy( TARGET: 913.7),circles);
showExecutionTime(new BranchAndBound( TARGET: 913.7),circles);
showExecutionTime(new SimulatedAnnealing( iteration: 50000, populationSize: 0, t: 1.0, tMin: .0001, alpha: 0.9, target: 913.7)
showExecutionTime(new IteratedLocalSearch( iteration: 3000, populationSize: 1000, target: 913.7),circles);
showExecutionTime(new VNS( iteration: 500, populationSize: 1000, kMax: 500, target: 913.7),circles);
showExecutionTime(new PSO( iteration: 1000, particleSize: 1000, dimension: 3, target: 913.7),circles);
showExecutionTime(new ACO( iteration: 7500, populationSize: 2000, evaporationRate: 0.7, target: 913.7),circles);
```

11 size ve belirtilen parametrelerle yapılan test sonucu.

Minimum width bulunurken oluşabilecek sapmaları gözle görünür kılmak için farklı frekanslarda ve rangelerde circle yarıçapları verilmiştir (2,987,100,650 vs.). Target değer bulunduğunda algoritmalar durdurulduğu için brute-force,greedy ve B&B dışındaki algoritmalarda çalışma zamanını parametrelere (iterasyon, populasyon boyutu vs.)göre yorumlamamız doğru olmayabilir.

Brute-force bu size için çalıştırıldığında heap boyutu aşıldığından test tamamlanamadı. Branch and Boundun bulduğu sonuca göre bu problem inputu için optimal çözüm 913,634 dür.

En hızlı optimal sonuca ulaşan algoritma Greedydir çünkü nlogn time complexitye sahip. Greedy,SimulatedAnneiling ve Iterated Local Searchte optimal çözüme ulaşılmıştır. VNS, PSO ve ACO da optimal çözüme çok yakın değerler elde edilmiştir. Yapılacak parametre değişiklikleriyle optimal çözüm tam olarak elde edilebilir fakat bunun karşılığında çalışma zamanında artış meydana gelebilir.

PSO ve ACO nun çalışma zamanlarının diğer algoritmalara göre fazla olmasının sebebi iterasyon sayılarının ve populasyon sizelarının yüksek olmasının yanı sıra local ve global best değerlerinin hesaplanmasıdır. VNS algoritmasında daha uygun bir çözüm bulunduğunda k değerinin sıfırlanmasından kaynaklı olarak uzun çalışma zamanları gözlenebiliyordu. Fakat Target değere daha önce ulaşılmasından kaynaklı beklenenden daha kısa süreli bir çalışma zamanı gözlenmiştir.

```
######### Greedy #########
Minimum width: 913,634 Circle Order: 100.0 5.0 500.0 2.0 987.0 2.0 650.0 4.0 121.0 7.0 99.0  Total time: 35ms
######### BranchAndBound #########
Minimum width: 913,634 Circle Order: 100.0 5.0 500.0 2.0 987.0 2.0 650.0 4.0 121.0 7.0 99.0 Total time: 2282ms
######### SimulatedAnnealing #########
Minimum width: 913,634 Circle Order: 99.0 7.0 121.0 4.0 650.0 2.0 987.0 2.0 500.0 5.0 100.0  Total time: 205ms
######### IteratedLocalSearch #########
Minimum width: 913,634 Circle Order: 99.0 7.0 121.0 4.0 650.0 2.0 987.0 2.0 500.0 5.0 100.0  Total time: 917ms
######### VNS #########
Minimum width: 913,675 Circle Order: 99.0 5.0 500.0 2.0 987.0 2.0 650.0 4.0 121.0 7.0 100.0  Total time: 729ms
######### PSO #########
Minimum width: 914,106 Circle Order: 100.0 4.0 650.0 2.0 987.0 2.0 500.0 5.0 121.0 7.0 99.0  Total time: 6169ms
######### ACO #########
Minimum width: 914,106 Circle Order: 99.0 7.0 121.0 5.0 500.0 2.0 987.0 2.0 650.0 4.0 100.0  Total time: 4104ms
```

Test 2)

```
Double [] circles = {99.0,5.0,2.0,500.0,650.0,4.0,100.0,32.0};
showExecutionTime(new BruteForce( TARGET: 665.972),circles);
showExecutionTime(new Greedy( TARGET: 665.972),circles);
showExecutionTime(new BranchAndBound( TARGET: 665.972),circles);
showExecutionTime(new SimulatedAnnealing( iteration: 100, populationSize: 0, t: 1.0, tMin: .0001, alpha: 0.9, target: 665.972)
showExecutionTime(new IteratedLocalSearch( iteration: 200, populationSize: 200, target: 665.972),circles);
showExecutionTime(new VNS( iteration: 100, populationSize: 200, kMax: 100, target: 665.972),circles);
showExecutionTime(new PSO( iteration: 200, particleSize: 200, dimension: 3, target: 665.972),circles);
showExecutionTime(new ACO( iteration: 500, populationSize: 200, evaporationRate: 0.7, target: 665.972),circles);
```

8 size ve belirtilen parametrelerle yapılan test sonucu.

Brute-force'nin asıl çalışma zamanı 32-37 msdir.

Brute-force algoritmasının bazı algoritmalardan hızlı çalışmasının sebebi parametrelerdeki iterasyon ve populasyon boyutlarıdır. Daha düşük parametre değerleriyle de optimal sonuca ulaşılabilir. Görüldüğü gibi en hızlı olan Greedydir çünkü çalışma zamanı nlogn.

```
######### BruteForce #########
Minimum width: 665,971 Circle Order: 99.0 5.0 500.0 2.0 650.0 4.0 100.0 32.0  Total time: 72ms
######### Greedy #########
Minimum width: 665,971 Circle Order: 32.0 100.0 4.0 650.0 2.0 500.0 5.0 99.0  Total time: 1ms
######### BranchAndBound #########
Minimum width: 665,971 Circle Order: 99.0 5.0 500.0 2.0 650.0 4.0 100.0 32.0 Total time: 21ms
######### SimulatedAnnealing #########
Minimum width: 665,971 Circle Order: 99.0 5.0 500.0 2.0 650.0 4.0 100.0 32.0  Total time: 15ms
######### IteratedLocalSearch #########
Minimum width: 665,971 Circle Order: 32.0 100.0 4.0 650.0 2.0 500.0 5.0 99.0  Total time: 131ms
######### VNS #########
Minimum width: 665,971 Circle Order: 99.0 5.0 500.0 2.0 650.0 4.0 100.0 32.0  Total time: 31ms
######### PSO #########
Minimum width: 665,971 Circle Order: 32.0 100.0 4.0 650.0 2.0 500.0 5.0 99.0  Total time: 73ms
######### ACO #########
Minimum width: 665,971 Circle Order: 32.0 100.0 4.0 650.0 2.0 500.0 5.0 99.0  Total time: 162ms
```

Test 3)

```
Double[] circles = {5.0, 5.0, 7.0, 47.0, 341.0, 918.0, 7.0, 4.0, 251.0, 1.0,
        77.0, 341.0, 5.0};
showExecutionTime(new Greedy( TARGET: 766.071),circles);
showExecutionTime(new BranchAndBound( TARGET: 772.099),circles);
showExecutionTime(new IteratedLocalSearch( iteration: 10000, populationSize: 1500, target: 766.071),circles);
showExecutionTime(new VNS( iteration: 1000, populationSize: 1000, kMax: 500, target: 766.071),circles);
showExecutionTime(new PSO( iteration: 2000, particleSize: 1500, dimension: 3, target: 766.071),circles);
showExecutionTime(new ACO( iteration: 20000, populationSize: 2000, evaporationRate: 0.7, target: 766.071),circles);
showExecutionTime(new SimulatedAnnealing( iteration: 100000, populationSize: 0, t: 1.0, tMin: .0001, alpha: 0.9, target: 766.071)
```

13 size ve belirtilen parametrelerle yapılan test sonucu.

Görüldüğü gibi en hızlı çalışan yine Greedydir ve 0-5 ms arasında optimal sonuca ulaşmıştır. Brute-force algoritmasının çalışma zamanı boyunca heap aşımından dolayı testi tamamlanamamıştır. B&B algoritması 3 dakika 51 saniyede optimal çözümü bulmuştur. Optimal çözüme ulaşamayan ACO 2 dakika 6 saniyede yakın bir çözüm bulmuştur. Iterated Local Search ve Simulated Annealing algoritmaları Target değere erken bir zamanda ulaştıklarından dolayı 3.7 ve 7.3 saniye gibi bir zamanda optimal çözüme ulaşmışlardır.

```
######### Greedy #########
Minimum width: 766,070 Circle Order: 7.0 77.0 5.0 341.0 4.0 918.0 1.0 341.0 5.0 251.0 5.0 47.0 7.0  Total time: 40ms
######### BranchAndBound #########
Minimum width: 766,070 Circle Order: 7.0 77.0 5.0 341.0 4.0 918.0 1.0 341.0 5.0 251.0 5.0 47.0 7.0 Total time: 231341ms
######### IteratedLocalSearch #########
Minimum width: 766,070 Circle Order: 7.0 77.0 5.0 341.0 4.0 918.0 1.0 341.0 5.0 251.0 5.0 47.0 7.0  Total time: 3778ms
######### VNS #########
Minimum width: 766,070 Circle Order: 7.0 77.0 5.0 341.0 4.0 918.0 1.0 341.0 5.0 251.0 5.0 47.0 7.0  Total time: 23052ms
######### PSO #########
Minimum width: 766,070 Circle Order: 7.0 77.0 5.0 251.0 5.0 341.0 1.0 918.0 4.0 341.0 5.0 47.0 7.0  Total time: 27957ms
######### ACO #########
Minimum width: 769,688 Circle Order: 5.0 341.0 4.0 918.0 1.0 341.0 5.0 251.0 5.0 77.0 7.0 47.0 7.0  Total time: 12646ms
######### SimulatedAnnealing #########
Minimum width: 766,070 Circle Order: 7.0 47.0 5.0 251.0 5.0 341.0 4.0 918.0 1.0 341.0 5.0 77.0 7.0  Total time: 7319ms
```

Test 4)

Aynı circlelardan oluşan bir liste verilmesi durumu kontrol edilip tüm algoritmalar için O(n) zamanda minimum width hesaplanmıştır.

```
Double[] circles = {5.0 ,5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0};
```

Normalde tüm algoritmalar 0ms sürede çalışıyor fakat ilk sırada hangi algoritma olursa olsun ekran görüntüsünde bulunan Brute-force algoritması gibi normal Total time ın üstüne 35-40 ms değer ekli halde gözüküyor.

```
######### BruteForce #########
Minimum width: 110,000 Circle Order: 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0  Total time: 40ms
######### Greedy #########
Minimum width: 110,000 Circle Order: 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0  Total time: 0ms
######### BranchAndBound #########
Minimum width: 110,000 Circle Order: 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0  Total time: 0ms
######### SimulatedAnnealing #########
Minimum width: 110,000 Circle Order: 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0  Total time: 0ms
######### IteratedLocalSearch #########
Minimum width: 110,000 Circle Order: 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0  Total time: 0ms
######### VNS #########
Minimum width: 110,000 Circle Order: 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0  Total time: 0ms
######### PSO #########
Minimum width: 110,000 Circle Order: 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0  Total time: 0ms
######### ACO #########
Minimum width: 110,000 Circle Order: 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0  Total time: 0ms
```