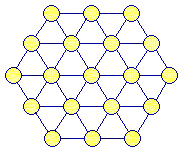
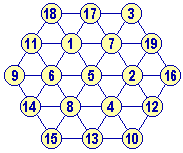
**2015-2016 Eğitim Yılı Yapay Zeka Dersi 2. Ödev Raporu**

Ödev konusu:

Ödevde 1’den 19’a kadar olan sayıların aşağıdaki altıgene yerleştirilmesi istenmektedir. Yerleştirme işlemi paralel tepe tırmanma yöntemiyle yapılacaktır.



Koşullar: her bir satırın, sol ve sağ yönlerdeki çapraz toplamların hepsinin eşit ve 38 olması istenmektedir. Örnek çözüm;



**Problem çözümü**

Neden Paralel Tepe Tırmanma?

Problem çözülürken soruda da belirtildiği gibi paralel tepe tırmanma algoritması kullanılmıştır. Bu algoritma kullanıcının girdiği K değeri kadar noktadan başlayıp çözümü bulmaya çalışmaktadır. Tek bir başlangıç noktasını kullanmak yerini birden fazla noktadan başlamak kesin çözüm yolunda bizi başarıya taşıyacaktır. Çünkü tek bir noktadan başlarsak lokal bir tepeye varmış olabiliriz. Bu yüzden paralel tepe tırmanma yöntemi bizi bu lokal tepelerden kurtarmayı sağlamaya çalışır.

Kullanılan Programlama Dili ve IDE

Bu problem JAVA programlama dilinde yazılmıştır. Bu dilde yazılmasının sebebi nesneye yönelimli bir dil olması ve esnek bir yapıda olmasıdır. IDE olarak ise ECLIPSE kullanılmıştır.

Çözüm

İlk olarak girilen K değerine bağlı olarak rastgele yukarıdaki tabloyu sağlayacak matrislerin yaratıldığı bir Matris sınıfı oluşturulmuştur. Bu sınıf içerisinde iki boyutlu matrisi belirten matrix[5][9] matrisi ve bu matrisleri başlangıçta rastgele oluşturmaya yarayan getMatris() metodu bulunmaktadır.

RandomSayi sınıfı da rastgele sayılar üretmektedir. Fakat bu üretme işlemi 1’den 19’a kadar olan sayılardan sadece birini rastgele oluşturmaya çalışmaktadır. Yani o anda oluşturulan matris için 1 değerinin sadece bir defa gelmesini sağlamaktadır. Bu sınıf Matris sınıfı içerisinde kullanılmıştır.

YeniMtr sınıfı da parent’tan child oluştururken kullanılan secimArray arrraylistini oluşturmada kullanınıyor.

Main sınıfı da bütün geri kalan işlemlerin yapıldığı sınıftır. Bu sınıfta kullanılan 6 metot vardır. Bu metotlar;

1. Oluştur() metodu: bu metot her bir parent’ tan derinlik sayısı(bu sayı kaç tane child üretileceği bilgisini tutuyor) kadar child matrisler üretmek için kullanılıyor. Üretilen child matrisler olusturulan2 adlı arraylist’ e ekleniyor.
2. rastgeleSec() metodu: bu metot bir sonraki durumda kullanılacak olan parent’ ları üretmeye yarıyor. Bu işlemi yaparken parent’ların ve childların tutulduğu listeleri tek bir arraylistte birleştiriyor ve bu arraylistteki en iyi k-2 elemanı parent listesine ekliyor. Eklenen parentlar birleşik arraylistten siliniyor. Ardından da kalan 2 parentı da rastgele birleşik arraylistten seçiyor.
3. Secim() metodu: bu metot child matrisleri oluştururken hangi değerlerin yerinin değişeceğini rastgele üretiyor. Üretilen değerler oluştur() metodunda kullanılıyor.
4. İyilik\_degeri\_hesapla(arraylist<Matris> olusturulan5) metodu: metoda gelen oluşturulan5 matris listesindeki her bir elemanın çözüme olan iyiliğini hesaplayıp iyilikm listesine bu değerleri yerleştiriyor.
5. En\_iyi\_matris(arraylist<Matris> matrisBul) metodu: bu metot ise en son durumda oluşturulan matrislerden en iyi iyilik değerine sahip matrisi bulmaya yaramaktadır.

Kodun aşağıdaki tablodaki K değerlerine göre çalıştırılması sonucu elde edilen çözüme olan en yakın uzaklık ve ortalama uzaklık değerleri verilmiştir. Bu değerler her bir ka değeri için 100 defa çalıştırılarak elde edilmiştir.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K değerleri | İterasyon sayısı | Çözüme olan uzaklık | Çözüme olan ortalama uzaklık | Çözümün bulunma yüzdesi |
| 1 | 10000 | 6 | 6 | 0 |
| 5 | 10000 | 4 | 22 | 0 |
| 10 | 10000 | 4 | 8 | 0 |
| 25 | 10000 | 4 | 4 | 0 |
| 50 | 10000 | 4 | 4 | 0 |
| 100 | 10000 | 4 | 4 | 0 |