Fibonacci saatinde ilgili kombinasyonları yaratmak için bir alternatif yol olabileceği ile ilgili sizin yanınıza gelmiştim. Maalesef ki Ankara’ya dönmemiz gerektiği için uygun zamanı da yaratamamış oldum. Kusura bakmayın bu konuyla ilgili. Bu muhabbeti yaptığımız için, bu konuda bir açıklama yapmayı borç bilip bunu anlatmak istedim. Aslında mevcuttaki kombinasyon bulma bir lineer optimizasyon problemi olarak ele alınabilir. Elbette mülakat sırasında böyle bir opsiyon söz konusu olmayabilir. Ancak amaç zaten gelişim ve farklı yönlerden bakmak ise bunun konuşulması güzel olacaktır.  
Lineer optimizasyon aslında bir çözüm uzayında optimal opsiyonu bulmamızı sağlayan bir disiplin. Bir amaç fonksiyonu yaratılır ve maksimize veya minimize edilerek o amaç fonksiyonunu amacın yönüne doğru optimal değişken setini ortaya çıkartan bir işlem. Bunun için C tabanlı Cplex tool’u var. Fikir edinmenizde yardımcı olacaktır. Keza Python’da da benzer toollar var.  
Normalde bu problemler çok uzun zaman alan çözüm sürelerine ulaşabiliyor ancak Fibonacci saatinde opsiyonlar sınırlı ve üstel şekilde artmadığı için çok hızlı çözümler alınabilir.

Değişken şeması;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| x\_vars | [0] | Equals 1 if square 1 is red, 0 otherwise |  | 3 | 1 | 5 |
| x\_vars | [1] | Equals 1 if square 1 is green, 0 otherwise |  |
| x\_vars | [2] | Equals 1 if square 2 is red, 0 otherwise |  | 2 |
| x\_vars | [3] | Equals 1 if square 2 is green, 0 otherwise |  |
| x\_vars | [4] | Equals 1 if square 3 is red, 0 otherwise |  | 4 | |
| x\_vars | [5] | Equals 1 if square 3 is green, 0 otherwise |  |
| x\_vars | [6] | Equals 1 if square 4 is red, 0 otherwise |  |
| x\_vars | [7] | Equals 1 if square 4 is green, 0 otherwise |  |
| x\_vars | [8] | Equals 1 if square 5 is red, 0 otherwise |  |
| x\_vars | [9] | Equals 1 if square 5 is green, 0 otherwise |  |

Bu değişkenler hangi karenin ne şekilde yanması gerektiğini amaç fonksiyonu ve eklenen kısıtlar altında karar verecek. Örneğin x\_vars[0] 1 değeri alırken x\_vars[1] 0 değerini alıyorsa bu durumda 1. Kare kırmızı yanacak. Tam tersinde yeşil yanacak. Bunların ikisinin de 1 olduğu durumda mavi yanacak. İkisi de 0 ise beyaz olarak kalacaktır.

Değişkenleri bu şekilde tanımladıktan sonra optimizasyon probleminin kabaca şöyle bir yapısı olur.  
Yapı başlar --  
Min ya da Max (Amaç fonksiyonu)  
Subject to  
Değişkenler üzerinden kısıtların belirtilmesi  
Değişkenlerin Binary olduğunun belirtilmesi  
Yapı biter –  
Buraya kadar bir yapı kurmuş olduk ancak amaç fonksiyonunu tanımlarken önemli olan aldığımız inputu (saati) tamamıyla dakika birimine çevirerek yansıtmak.  
Bu yüzden de amaç fonksiyonunu yazarken kırmızı renkli değişkenleri 60 ile yeşil olanları ise normalde de yaptığımız 5 ile çarpacağız.  
  
Amaç fonksiyonunu da bu değişkenlere uygun şekilde tanımlamaktayız.

Amaç fonksiyonu:

(x\_vars[0]+x\_vars[2]+2\*x\_vars[4]+3\*x\_vars[6]+5\*x\_vars[8])\*60+ (x\_vars[1]+x\_vars[3]+2\*x\_vars[5]+3\*x\_vars[7]+5\*x\_vars[9])\*5

Verilen bu amaç fonksiyonunda değişkenler ait olduğu karelerin Fibonacci seri değeri ile çarpılırken kırmızıları temsil edenler 1 saate tekabül eden 60 ve yeşilleri temsil edenler ise aynı şekilde Fibonacci seri değeri ve dakika değeri olan 5 ile çarpılıyor. Bu şekilde karelerin toplam gösterdiği saatin dakika cinsinden değeri bulunmuş oluyor.

Problemi çözme şeklimiz amaç fonksiyonunun ne yönde optimize edileceğini önemsememektedir. Maximizasyon ya da minimizasyon yapılabilir. Burada amaç fonksiyonu aslında kısıt olarak kullanıcının girdiği dakikaya eşitlenerek eklenecektir. Böylelikle solver bu dakikayı toplam olarak ortaya çıkartan optimal çözüm bir karar değişkeni kombinasyonu ve aynı zamanda da saatte bir gösterim şekli oluşturacak. Eğer ilk optimal çözümü bulamaz ise user’ın girdiği dakika değeri saatin gösterimine uygun değil demektir. Bu durumda program uyarısını yapıp kendini kapatır.

Model bir başlangıç çözümü bulduktan sonra aynı modeli tekrar çözdürmeyi planlıyoruz, ancak bu sonucu vermeyecek şekilde manipüle ederek. Bunu da çözümü oluşturan karar değişkenlerine göre bir kısıt oluşturup bunu modele ekleyerek yapmamız gerek. Bu kısıt ise şu şekildedir;

Basitçe bu kısıt 1’e eşit olan karar değişkenleri, tüm karar değişkenleri toplamına (toplamdaki 1 almış karar değişkeni sayısı) eşit olamaz. Bu şekilde bu kombinasyonun tamamının aynı anda 1 alma olasılığını kaldırmış oluyoruz. Eğer bu toplam ilerideki iterasyonlarda kullanılan bir altküme olsaydı bu alternatif çözümleri kaçırmamıza sebep olurdu, ancak elimizdeki problemin yapısı böyle bir yapıyı desteklemediği için kullanımda sakınca yoktur.

Ancak şöyle bir durum var ki, eşitsizlik sembolü maalesef ki lineer optimizasyonda direkt olarak kullanılabilen bir konsept değil. Detaylıca anlatmak yerine bir örnek üzerinden anlatmak isterim. Bu kısıtı!= yerine >= ve <= ile ekleyeceğiz.

Örneğin son iterasyonda bulunan sonuç şu şekilde olsun; x\_vars[0]=1, x\_vars[2]=1, x\_vars[3]=1

Bu durumda modele eklenmesi gereken kısıt; x\_vars[0] + x\_vars[2] + x\_vars[3] !=3

Lineer yapıyı bozmamak adına eklenmesi gereken iki kısıt;

x\_vars[0] + x\_vars[2] + x\_vars[3] - smal\_m >= 3 – y\_var\*big\_m

x\_vars[0] + x\_vars[2] + x\_vars[3] + smal\_m <= 3 + (1-y\_var)\*big\_m

Bu kısıtlarda big\_m ve small\_m olarak belirtilen parametreler aslında mevcut modelde aktif olması gereken kısıtı seçmek için eklenmiş elemanlardır. Small\_m parametresi çok küçük bir sayıyı temsil eder. Model integer ve tam sayılardan oluştuğu için 0.00001 değeri almıştır. Bu değer toplam değer olan (3)’e tekrar eşit olmamasını sağlamak için eklenmiştir. Bu durumda karar değişkenlerinin toplamı 3.00001’den büyük ya da 2.9999’dan küçük olabilir ama kesinlikle 3 olamaz. Big\_m parametresi ise modele yapay bir değişken bağlanmıştır(y\_var) ve çok büyük bir sayıyı temsil etmektedir. Kodumuzda 1000000 değere tekabül eder. Eğer y\_vars değişkeni 1 olursa üstteki (>=) kısıtının bir anlamı kalmıyor. Tam tersi durumda ise alttaki (<=) kısıtının anlamı kalmıyor. Bu değişken amaç fonksiyonunun yönüne göre hareket etmektedir. Bu şekilde eşit değildir kısıtı lineer yapıyı bozmadan eklenmiş olur.

Her çözümde bu yapıyı ekleyerek sonuçları kaydettikten sonra, kara değişkenleri her sonuç için tek tek değerlendirilir ve buna göre de her sonuçtaki karelerin rengi bir 2’li for döngüsü ve if mekanizması ile bulunur ve daha sonra da animasyona uyacak şekle getirilerek yansıtılmıştır. Bu konuda biraz gpt’den yardım aldım. Çok elimin hâkim olduğu bir konu değildi, o yüzden hatalar olabilir ama output gözlemlenebiliyor.