Bölüm 7: Yakınsama Algoritmaları

Algoritmalar

Yakınsama Algoritmaları

- Bazı problemlerin çözümlerini bulmak çok zor veya imkansız olabilir.
- Yakınsama algoritmaları,
  - Mükemmel çözümü bulmak yerine,
  - Orijinal çözüme mümkün olduğunca yakın çözümler üretir.
  - Karmaşık problemler makul bir süre içerisinde çözülebilir.

Yakınsama Algoritmaları

- Polinom zaman karmaşıklığına sahiptir.
- Orijinal çözüme belli bir oran veya yüzde ile yakın çözümler üretir.
- Hesaplama kaynakları kısıtlı olduğunda tercih edilir.
- Terazinin bir kefesine bir ağırlık koyulduğunda,
  - diğer kefeye tam olarak aynı ağırlığı koymak zor olabilir.
  - Bu durumda, çeşitli küçük ağırlıklar kullanılarak,
  - büyük ağırlığa olabildiğince yakın bir denge kurulabilir.

Yakınsama Algoritmalarının Uygulamaları

- Seyir satıcısı (Traveling salesman) problemi:
  - Seyyar satıcının en kısa sürede en fazla müşteriye ulaşmasını sağlayan rotaya karar vermek.
- Sırt çantası (Knapsack) problemi:
  - Çantaya sığacak, maksimum fayda sağlayacak eşyaların seçilmesi.
- İşlemci (Job scheduling) planlama:
  - Bilgisayarda birden fazla sürecin en verimli şekilde yürütülmesi.

Yakınsama Algoritmaları Türleri

- Yakınsama:
  - Orijinal problemin daha basit bir versiyonunu çözülür.
- Açgözlü (Greedy):
  - Her adımda en iyi görünen seçenek seçilerek ilerlenir.
- Yerel Arama (Local search):
  - Mevcut çözüm küçük değişikliklerle iyileştirilir, daha iyi çözüm bulunur.

Seyahat Eden Satıcı Problemi

- Optimizasyon Problemi:
  - Satıcı, bir dizi şehri en kısa mesafede ziyaret etmek istiyor.
- Kombinatoriyel Problemler:
  - Tüm şehirlerin gezilme sırasını belirlemek.
- NP-Zor Problemi:
  - Pratikte çok sayıda şehir için en iyi rotayı bulmak zor olabilir.

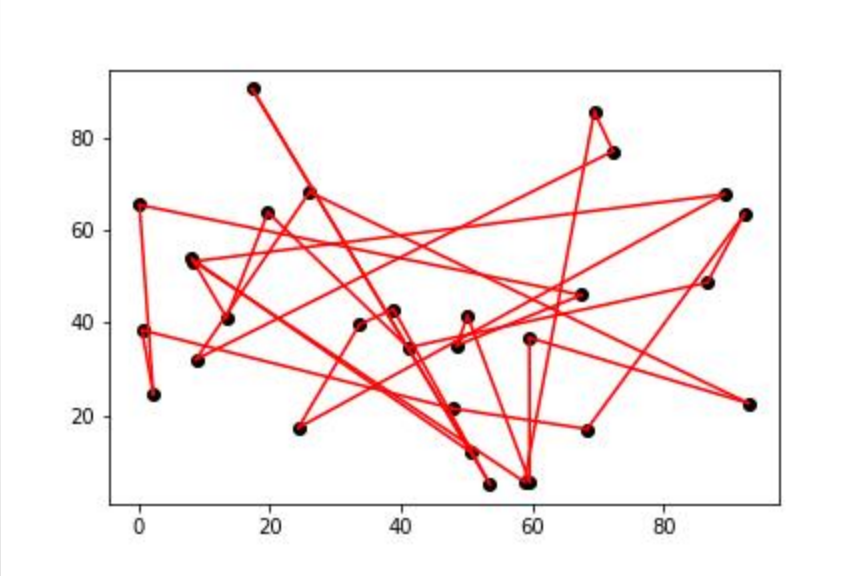
Seyahat Eden Satıcı Problemi

- Şehirler:
  - *n* şehir (örneğin, A, B, C, ...)
- Mesafeler:
  - d_{ij}: i ve j şehirleri arasındaki mesafe

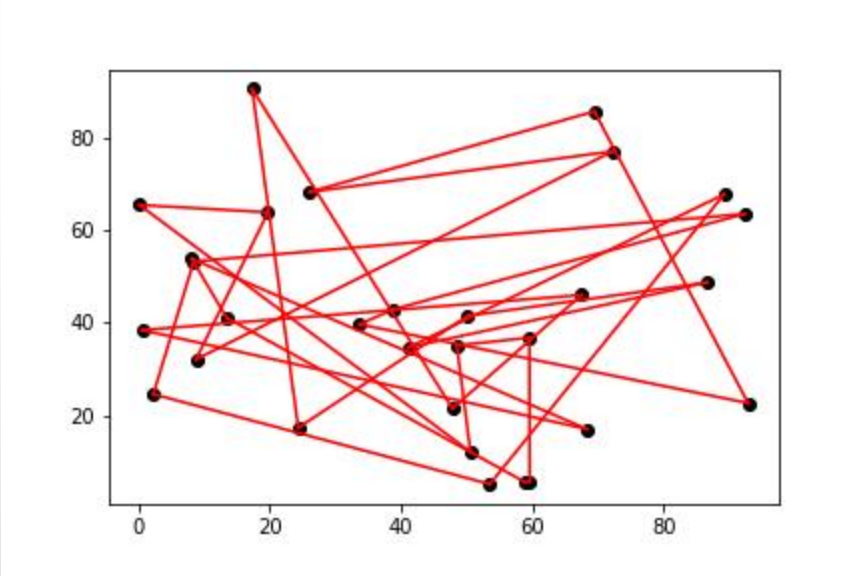
Problemin Çözümü

- Tam Çözüm:
  - Tüm olası rotaları kontrol etmek
- Yaklaşık Çözümler:
  - Heuristik algoritmalar (örneğin, Nearest Neighbor, Genetic Algorithms)
- Optimizasyon Araçları:
  - Simulated Annealing, Ant Colony Optimization..

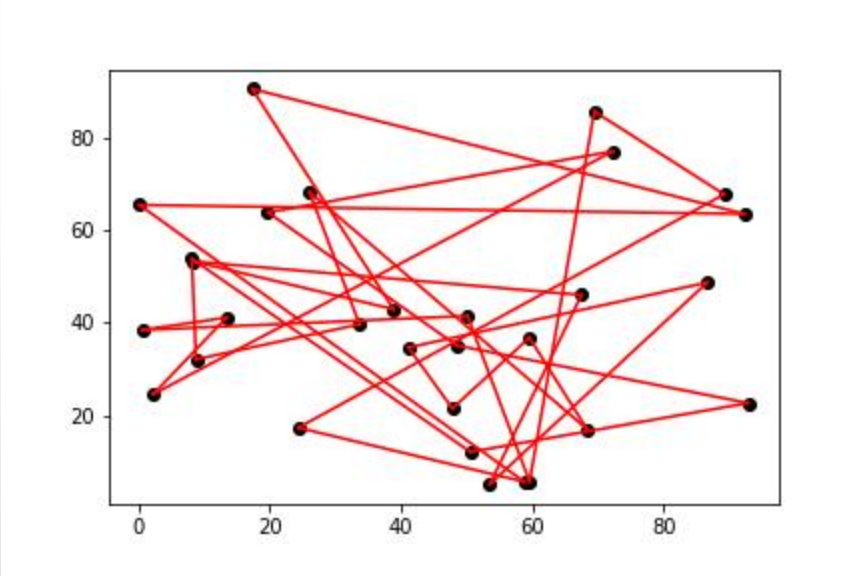
Traveling Salesman



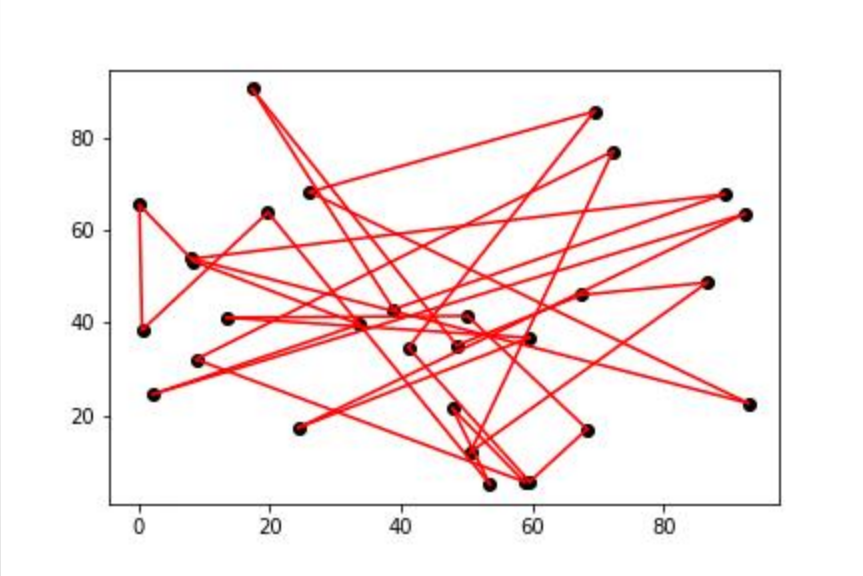
Traveling Salesman



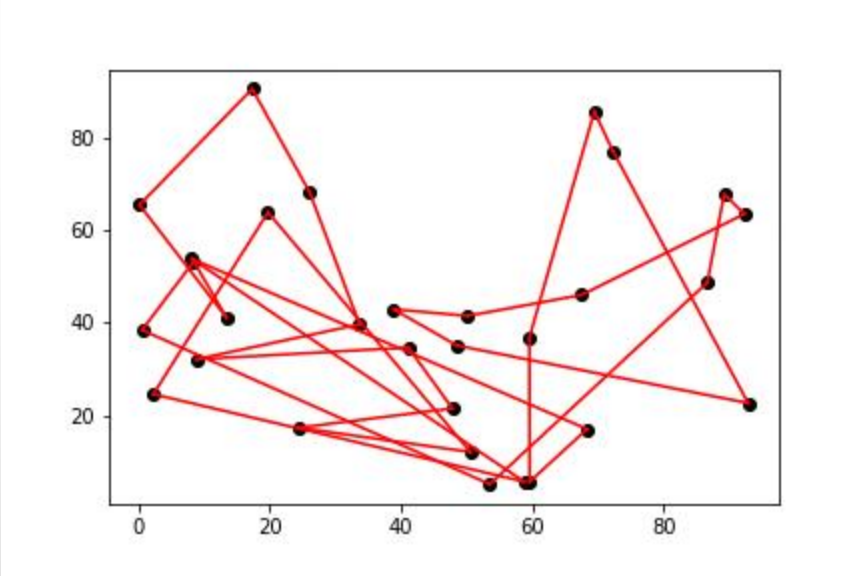
Traveling Salesman



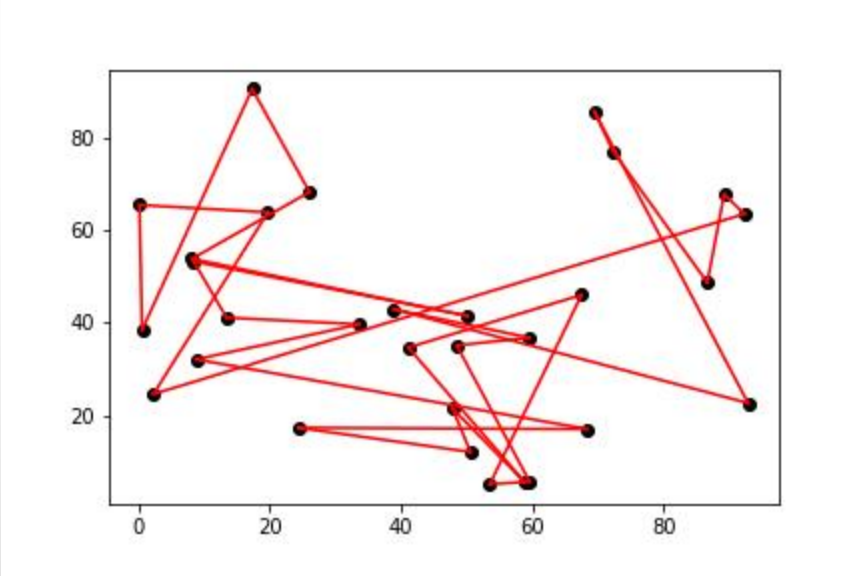
Traveling Salesman



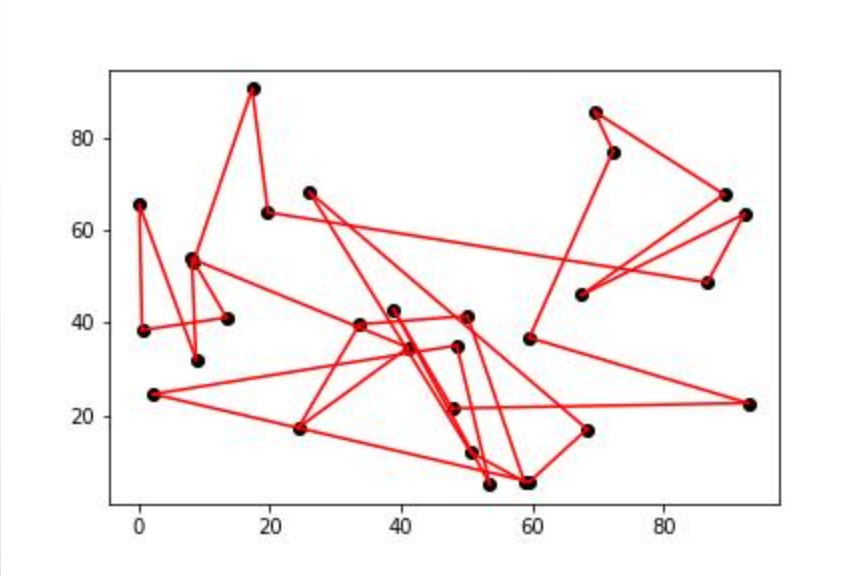
Traveling Salesman



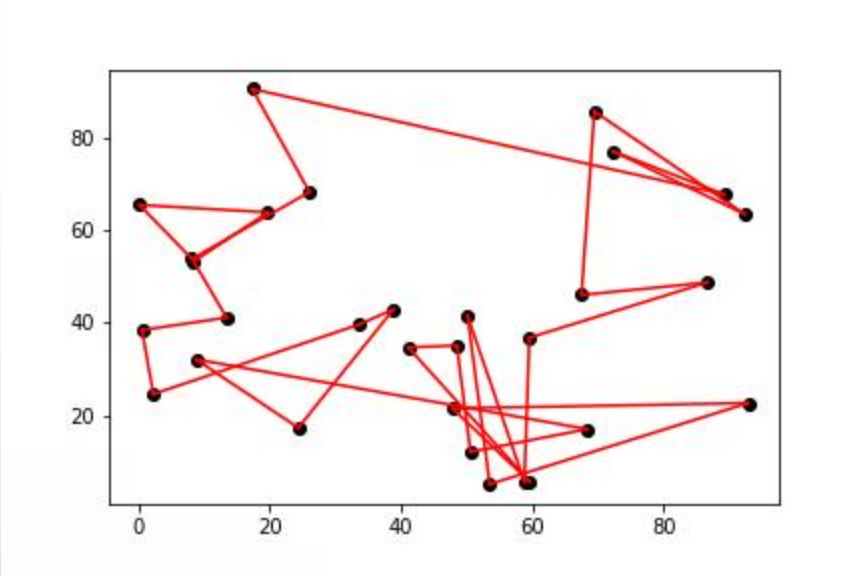
Traveling Salesman



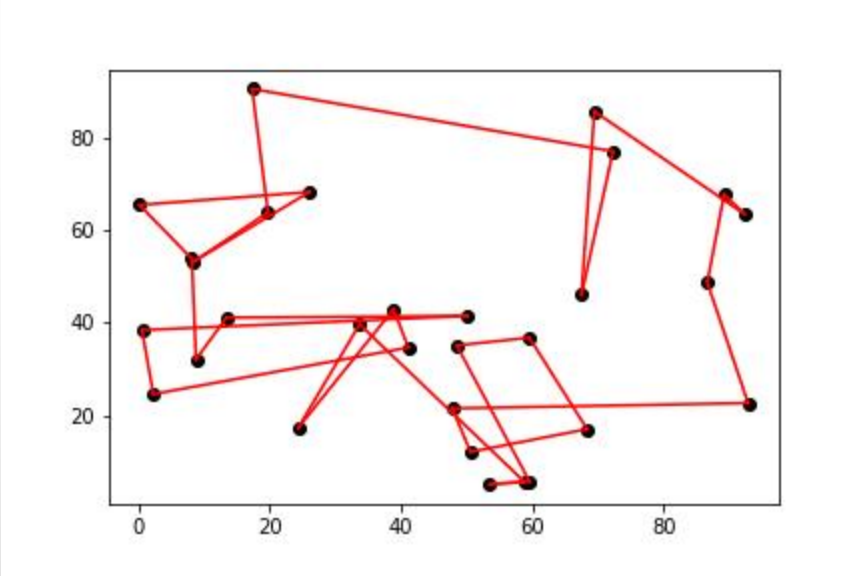
Traveling Salesman



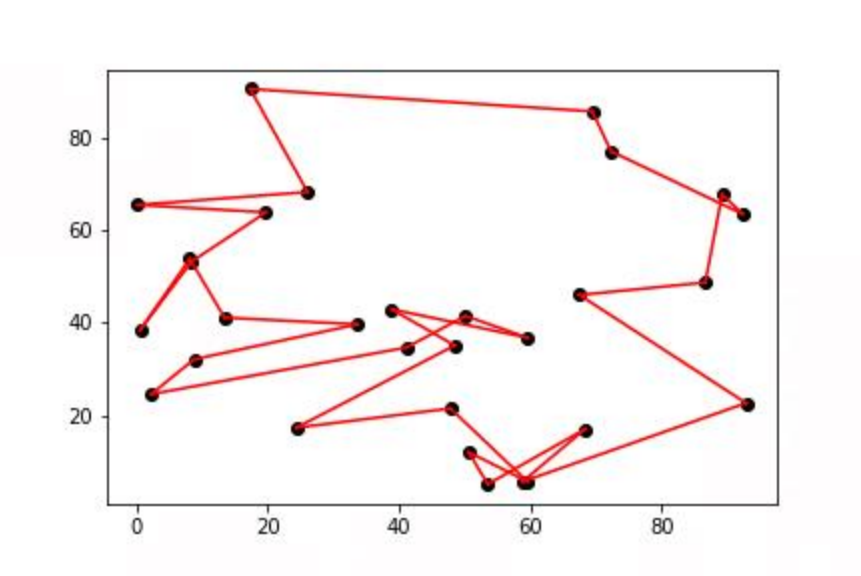
Traveling Salesman



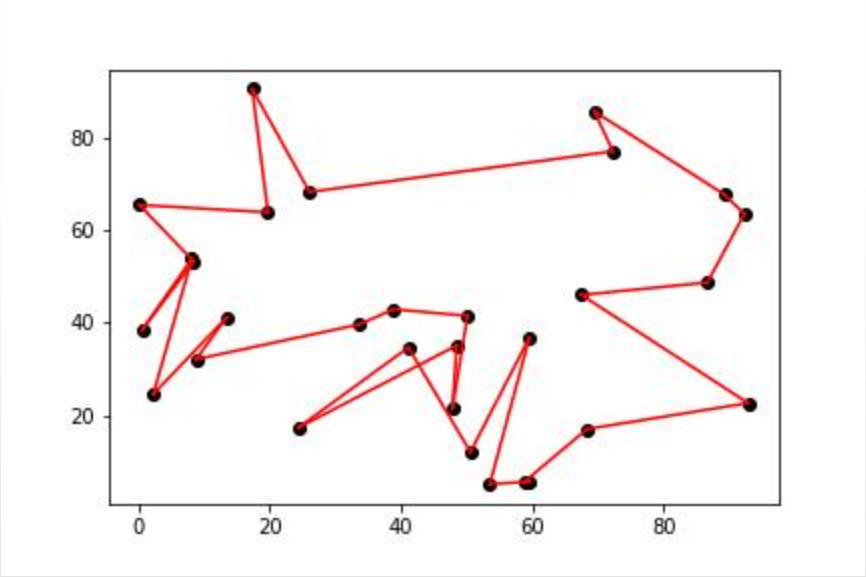
Traveling Salesman



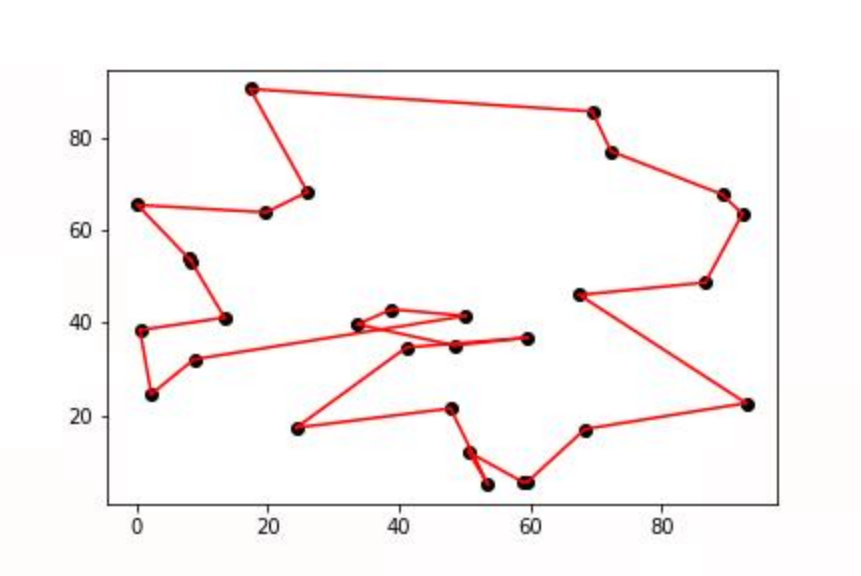
Traveling Salesman



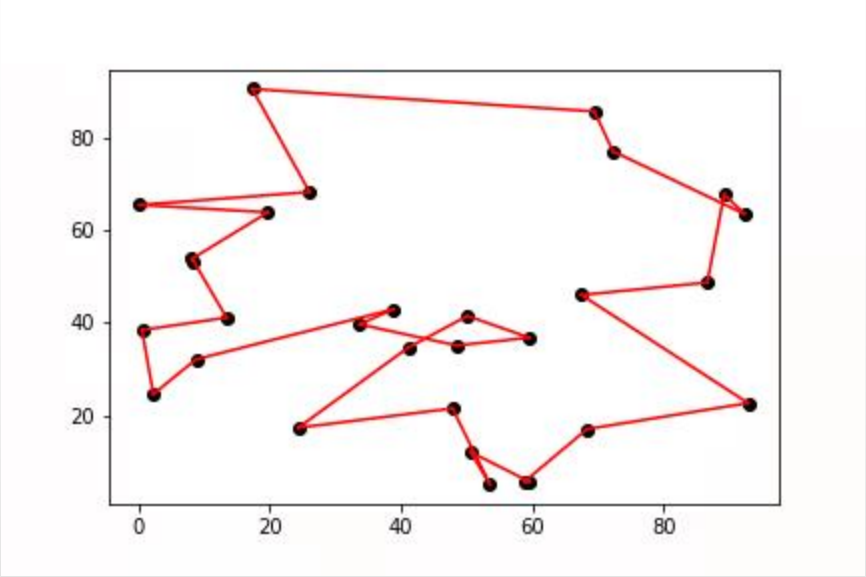
Traveling Salesman



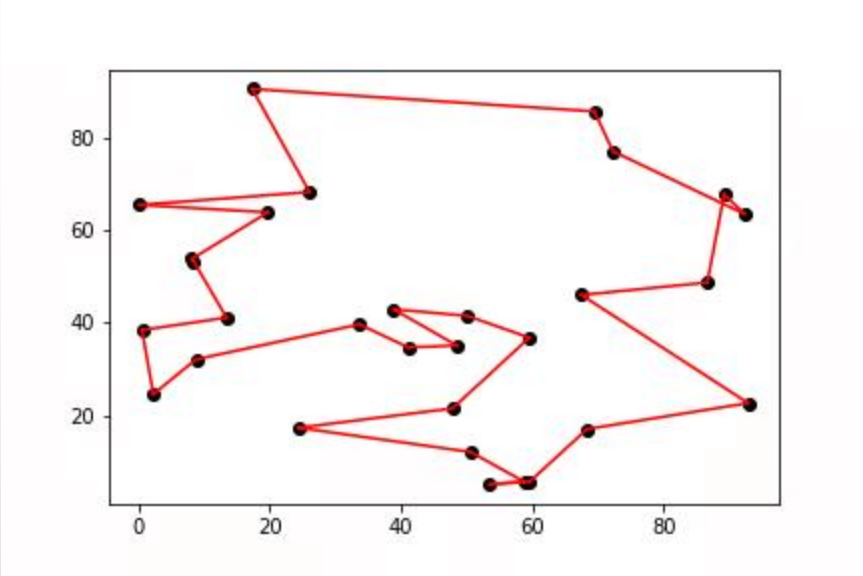
Traveling Salesman



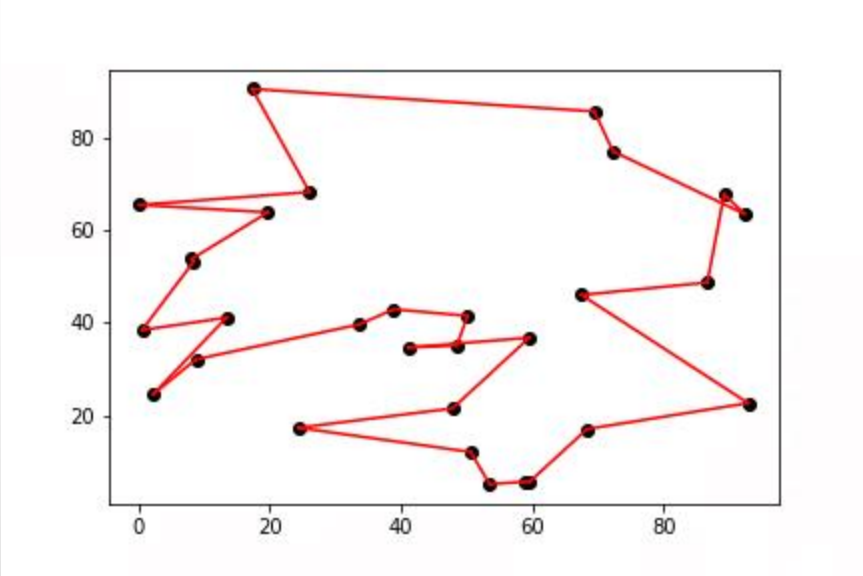
Traveling Salesman



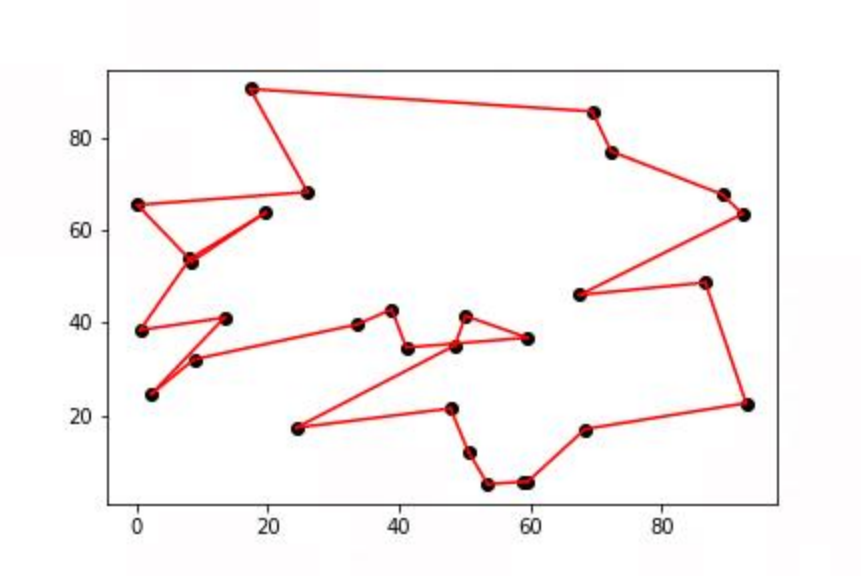
Traveling Salesman



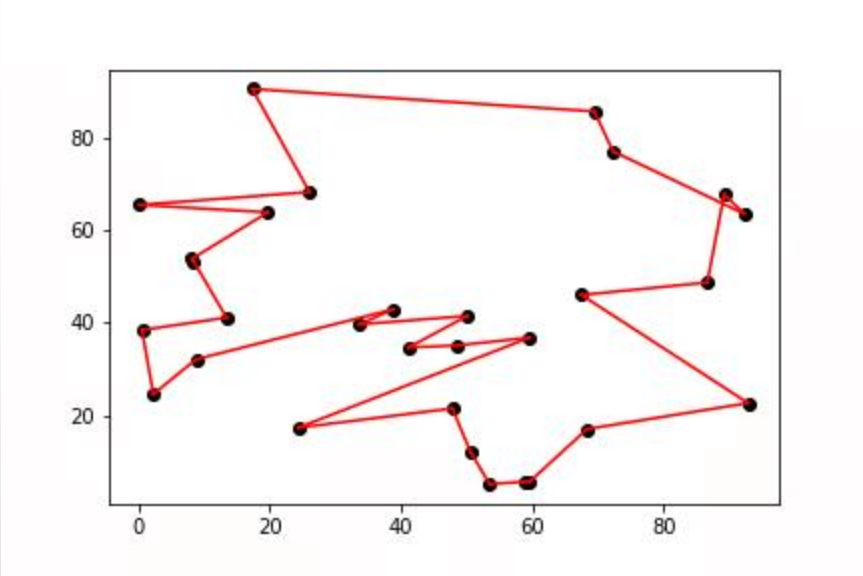
Traveling Salesman



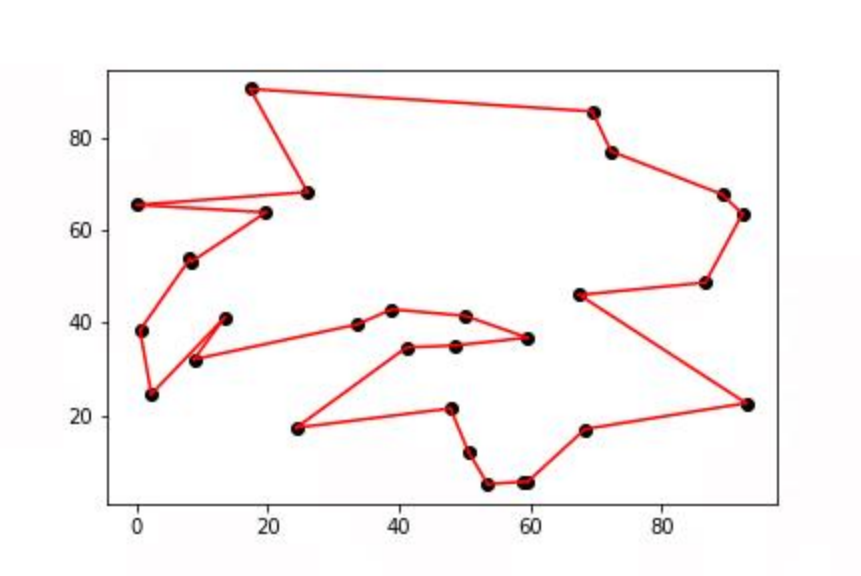
Traveling Salesman



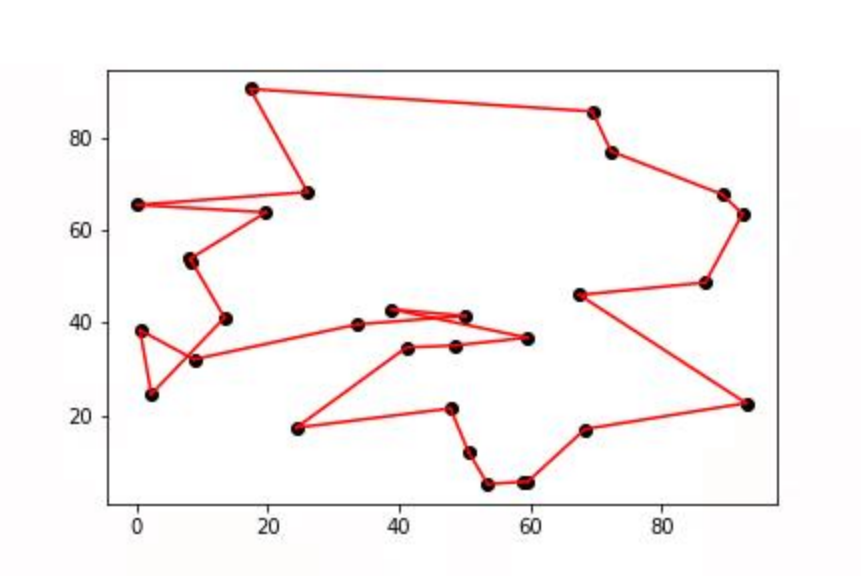
Traveling Salesman



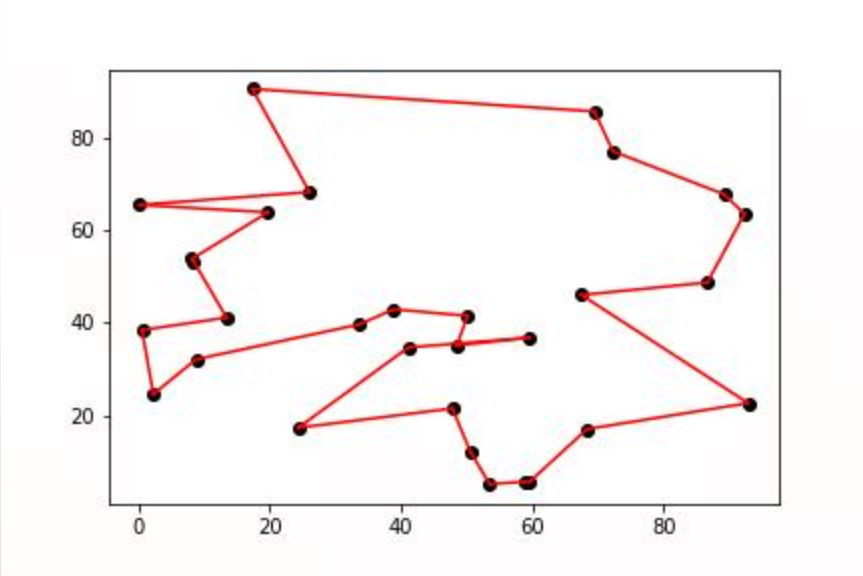
Traveling Salesman



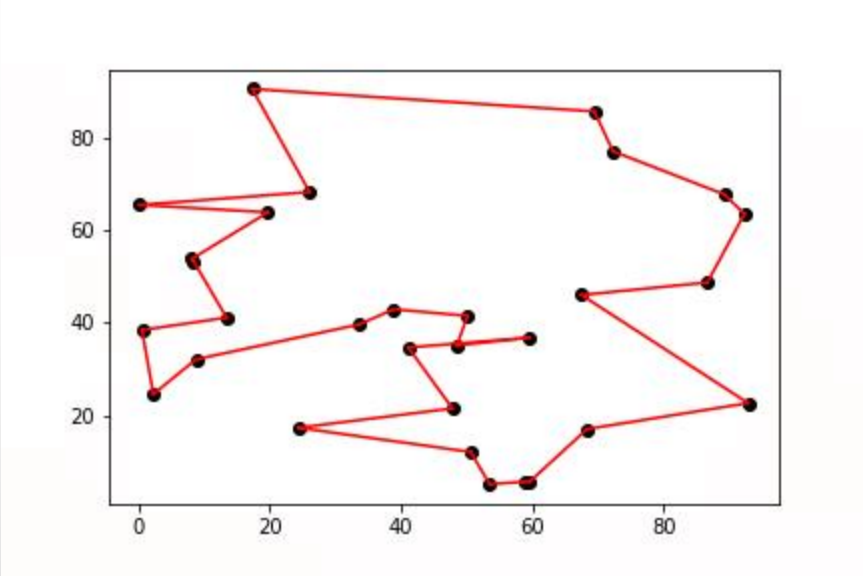
Traveling Salesman



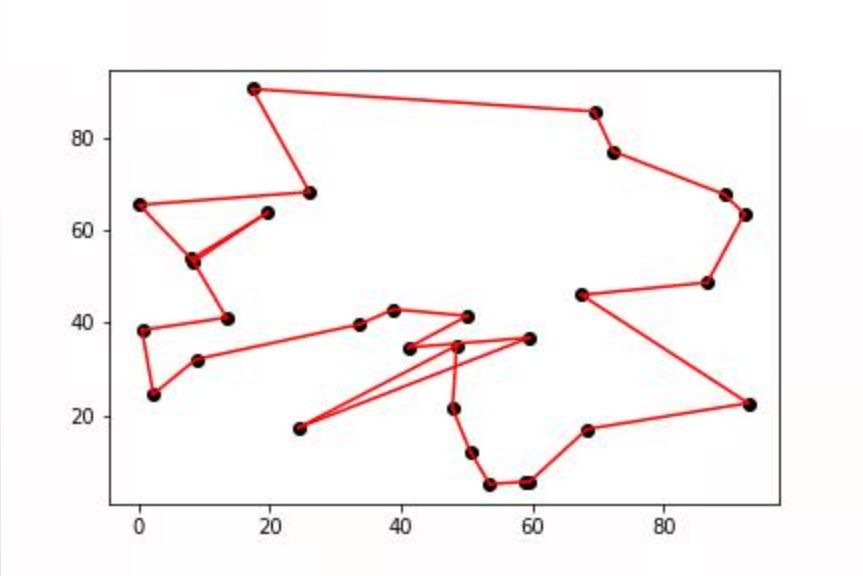
Traveling Salesman



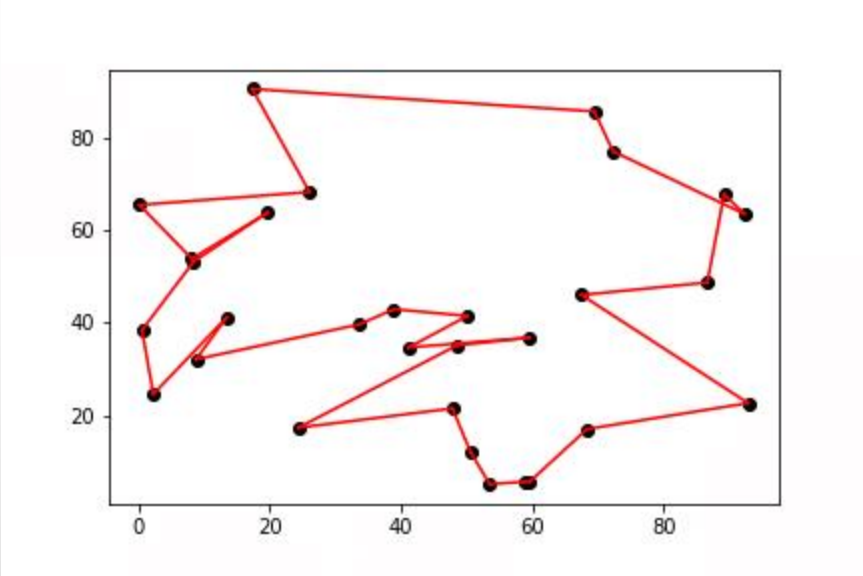
Traveling Salesman



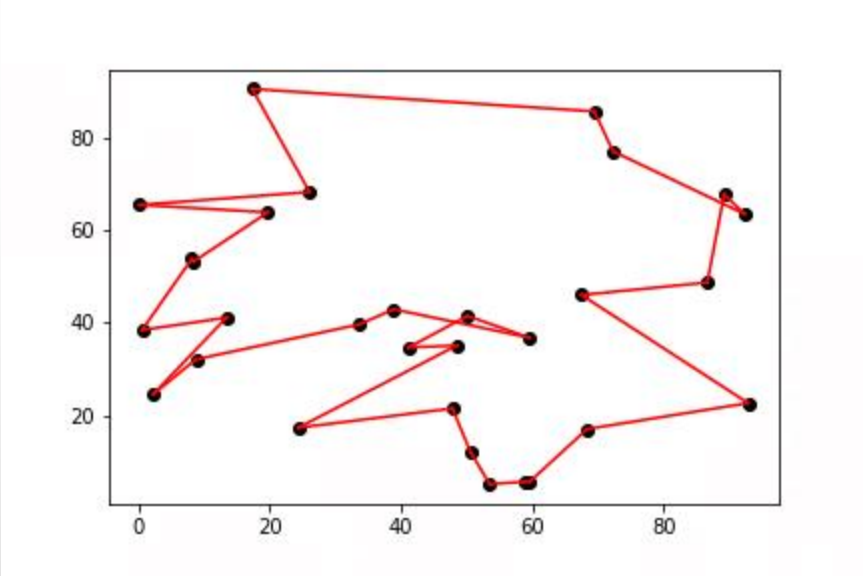
Traveling Salesman



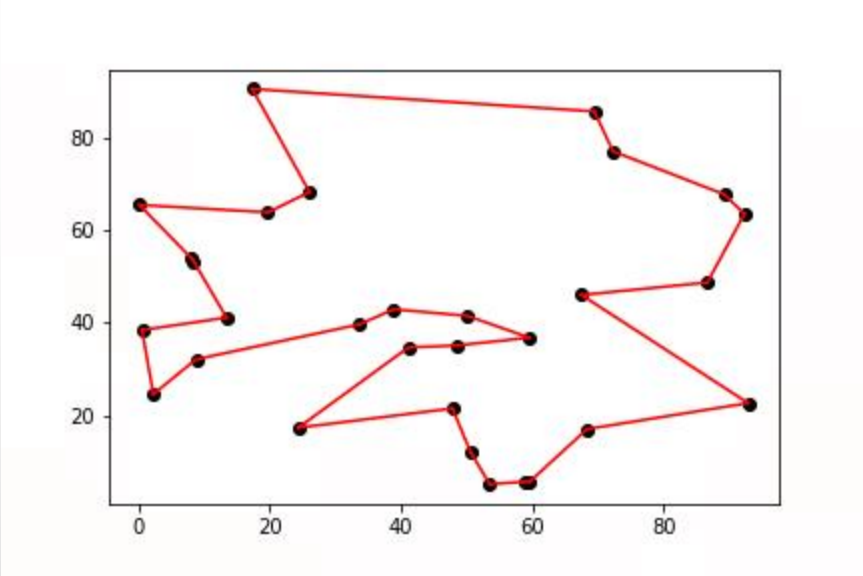
Traveling Salesman



Traveling Salesman



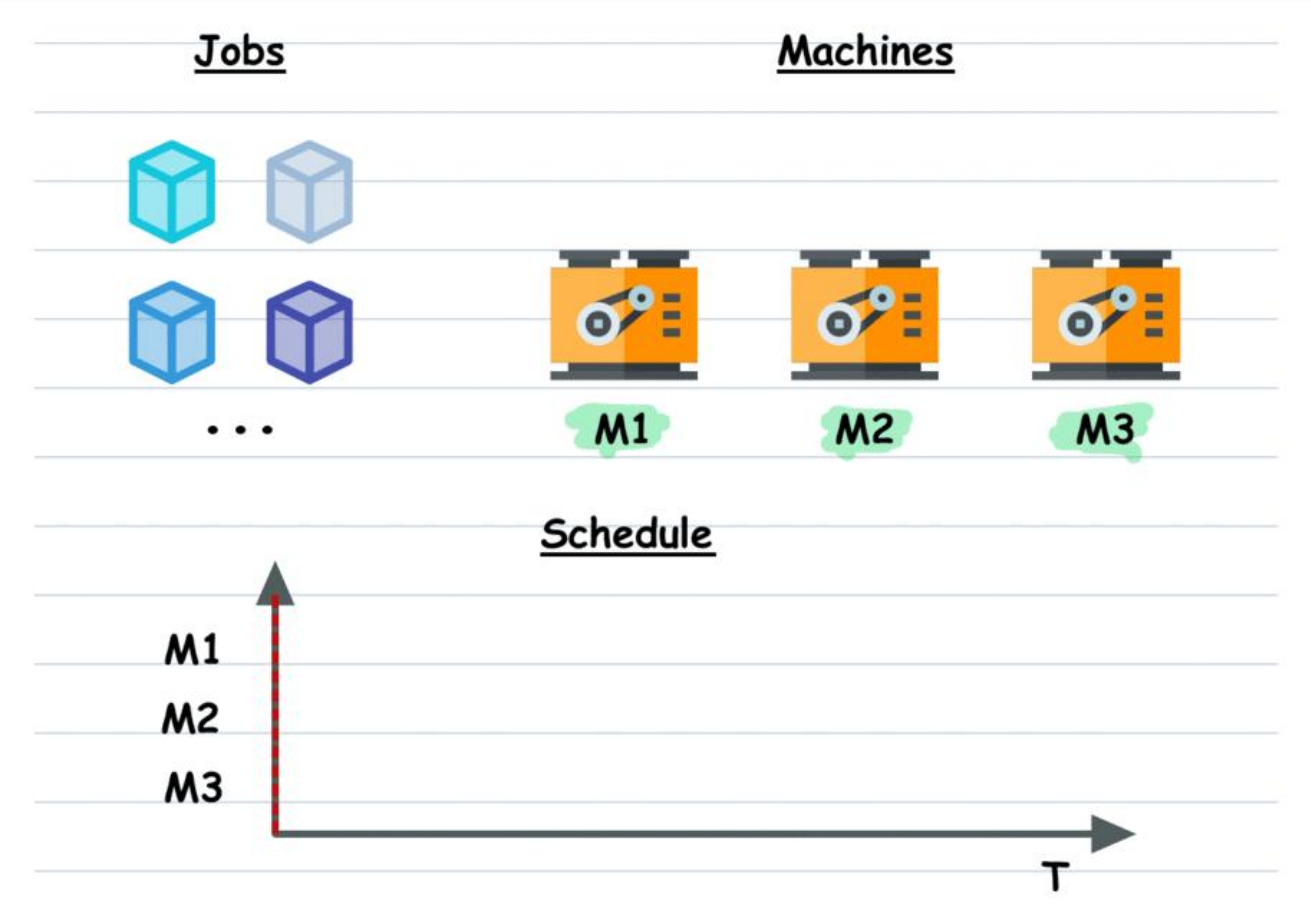
Traveling Salesman



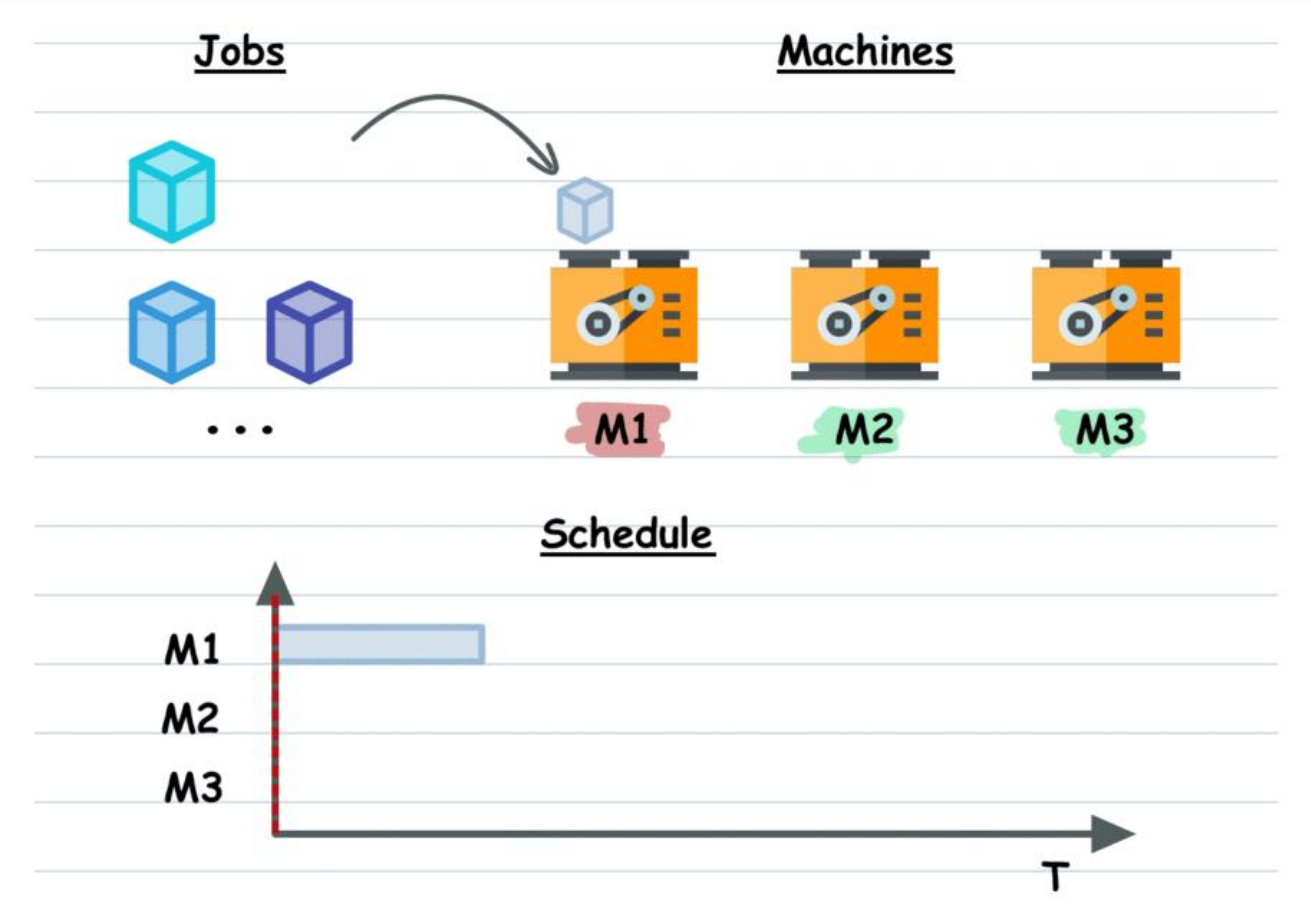
İş Çizelgeleme

- Optimizasyon Problemi:
  - Belirli kaynaklarla (örneğin, işçiler, makineler) işlerin en verimli şekilde planlanması.
- Amaç:
  - İşlerin tamamlanma süresini minimize etmek veya
  - Belirli kriterlere göre optimize etmek.

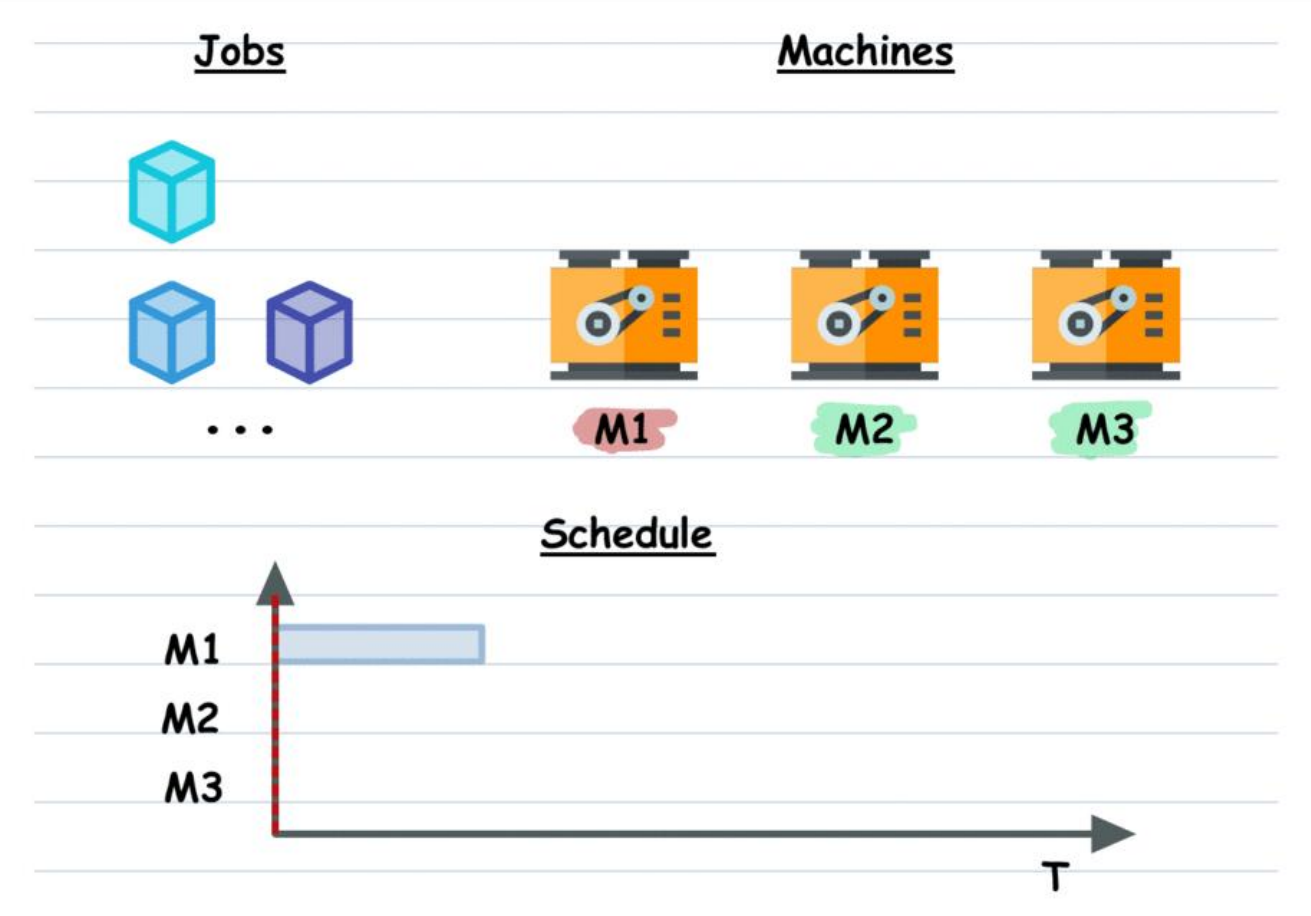
Job Scheduling



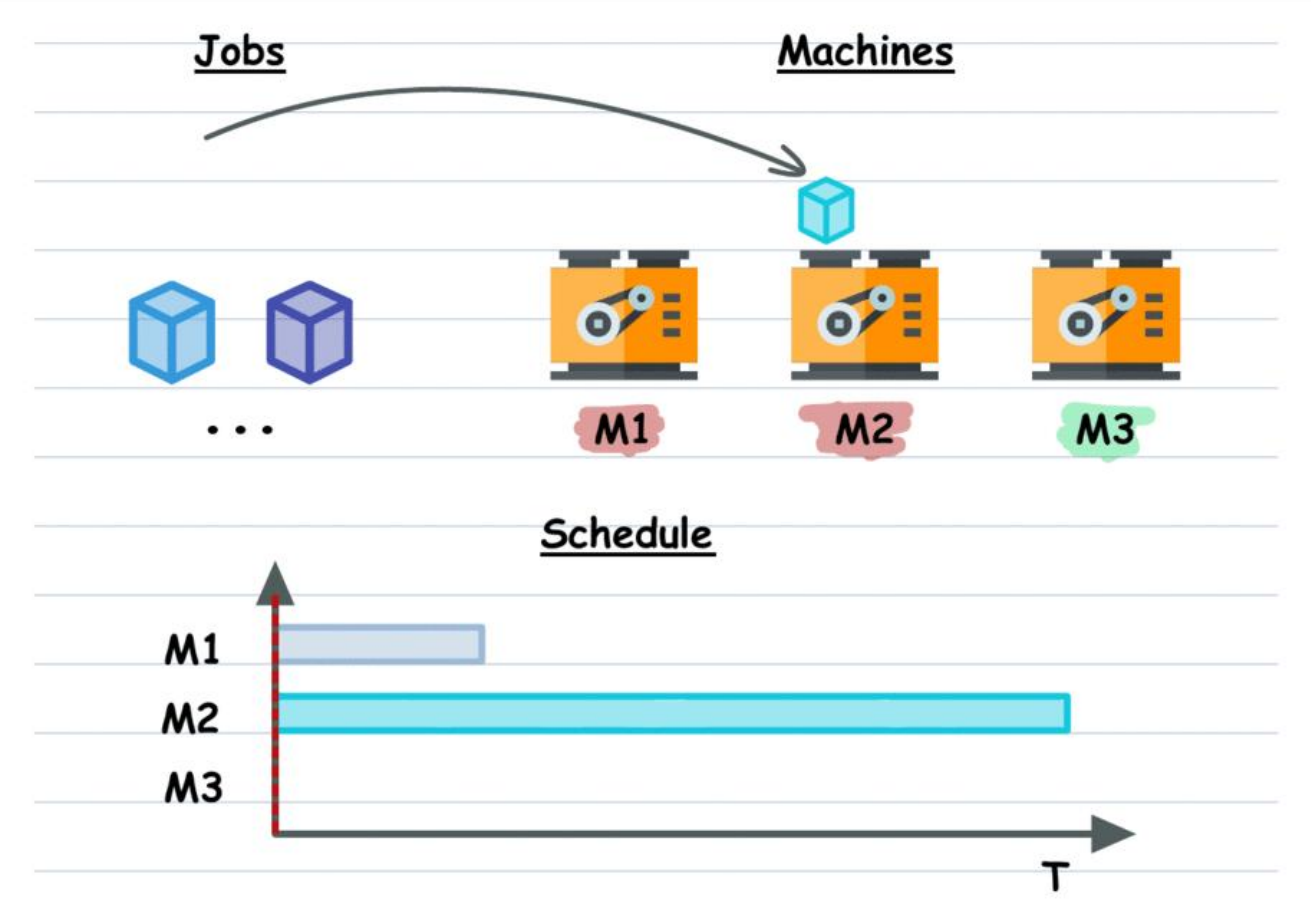
Job Scheduling



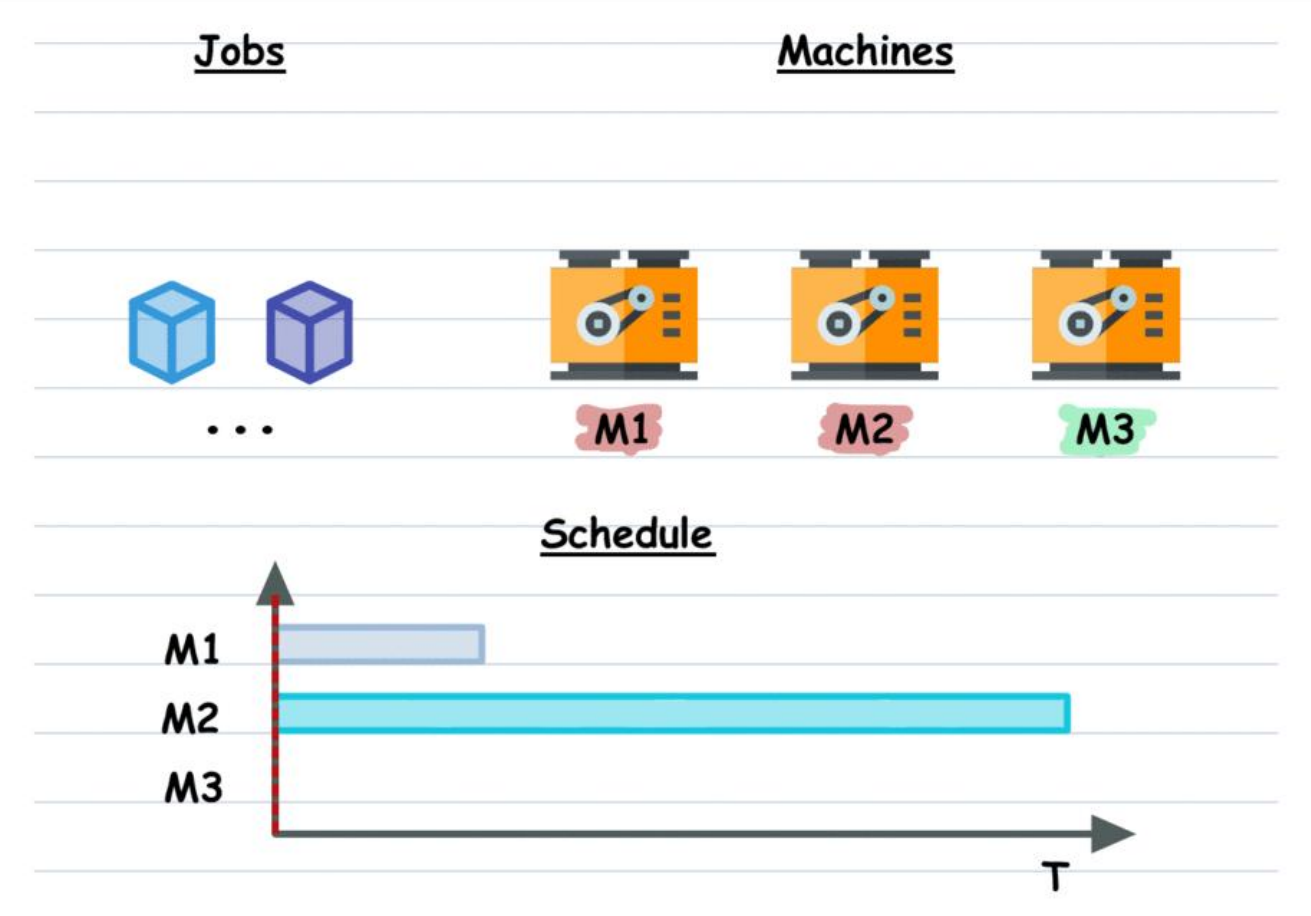
Job Scheduling



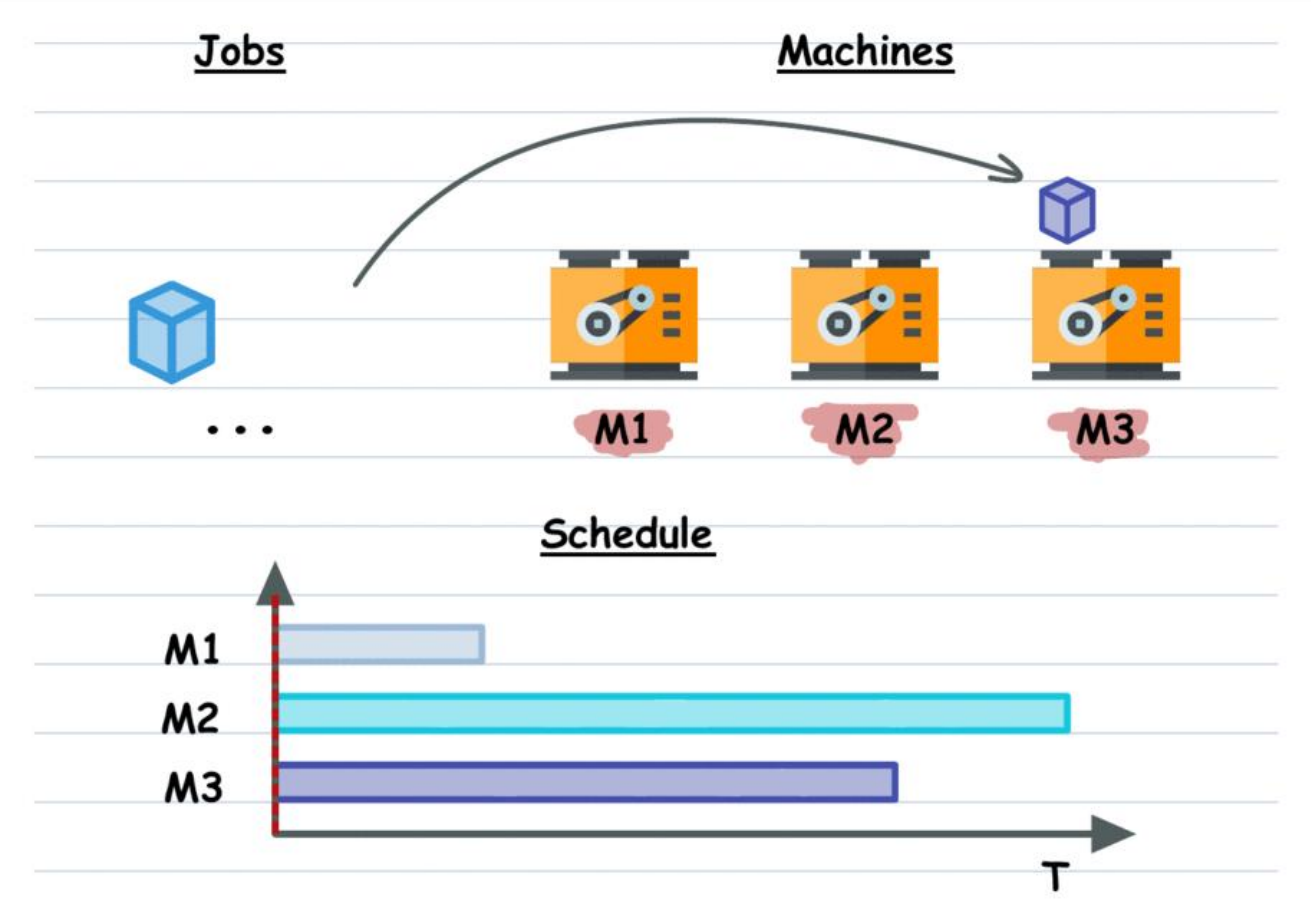
Job Scheduling



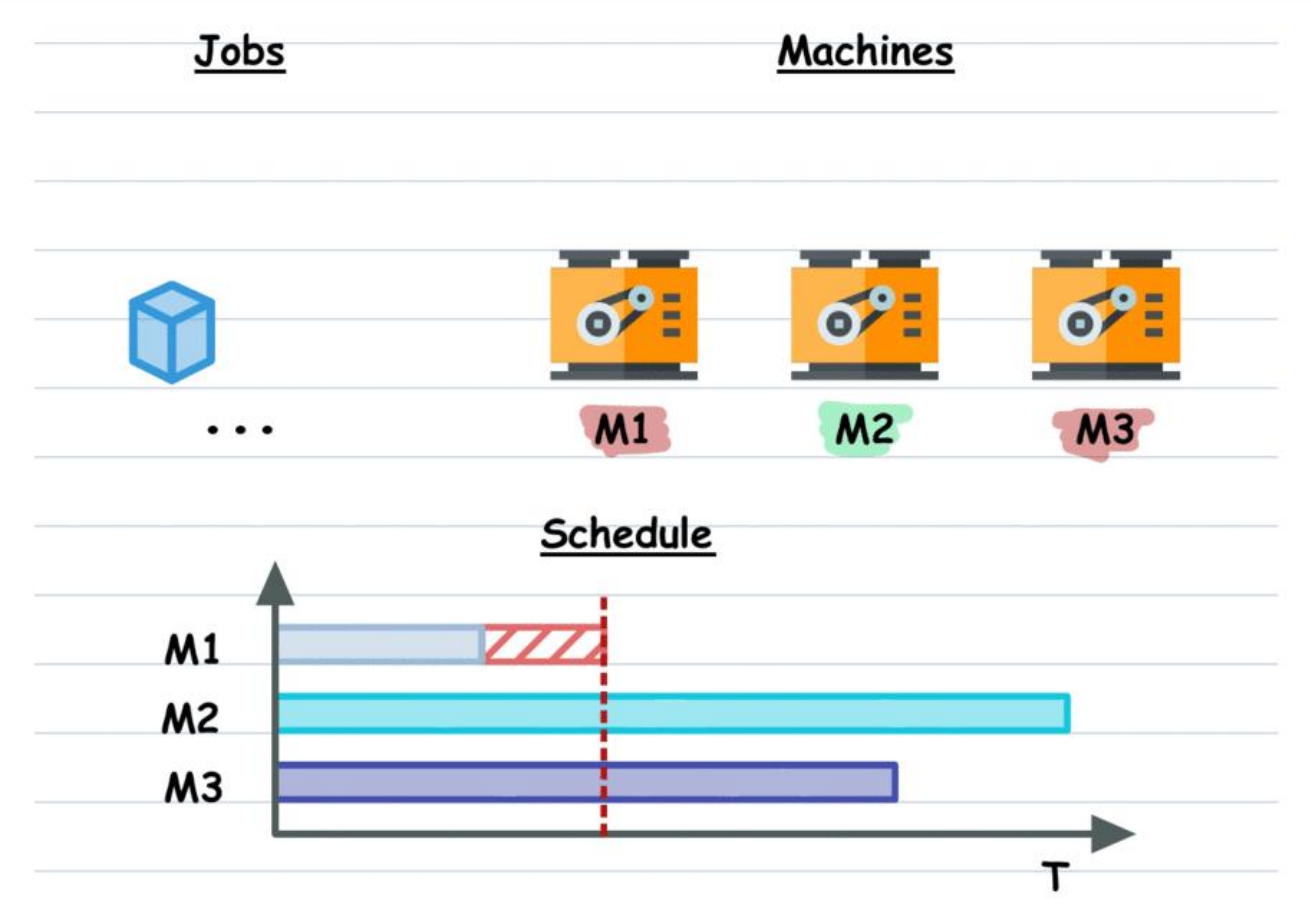
Job Scheduling



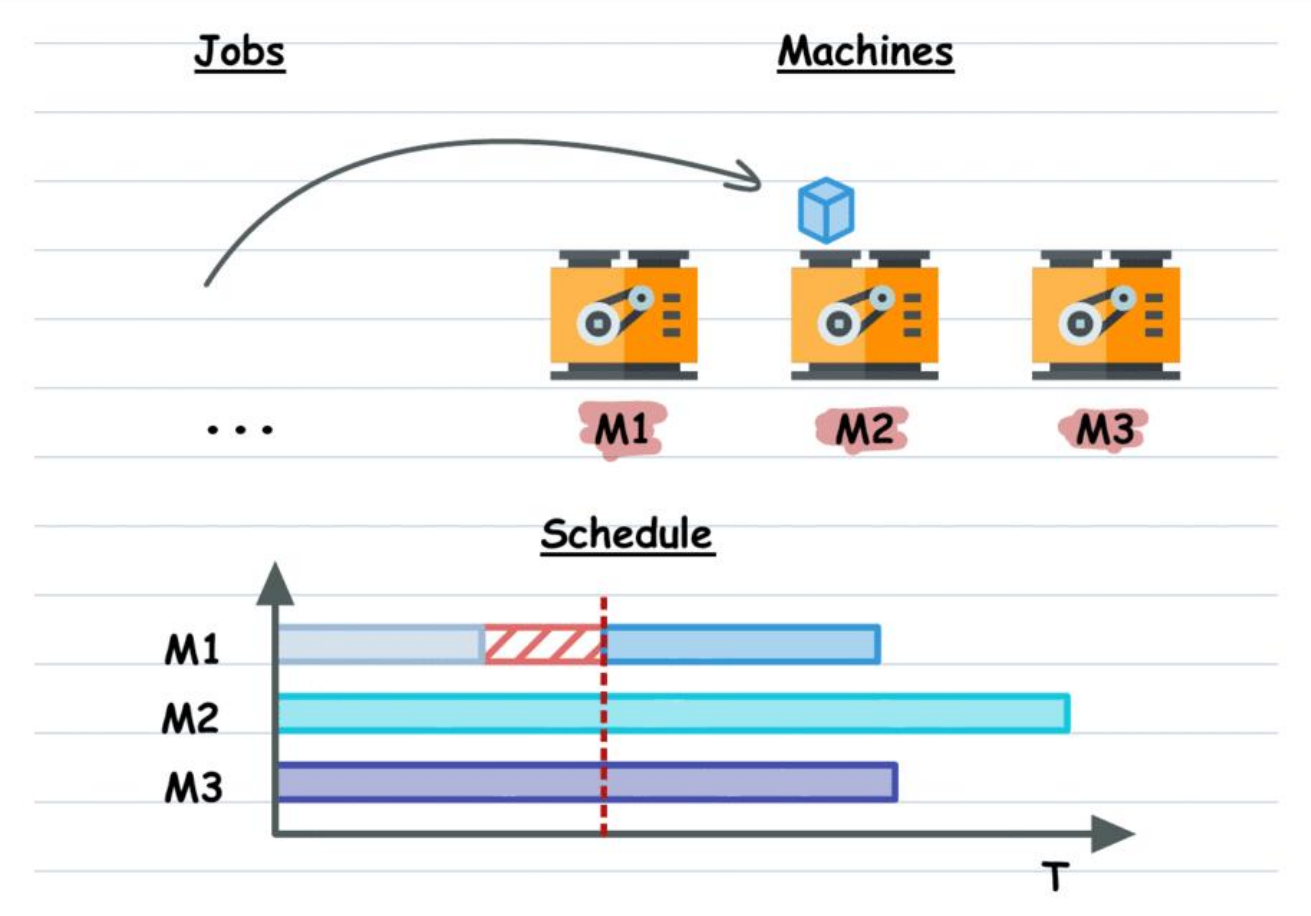
Job Scheduling



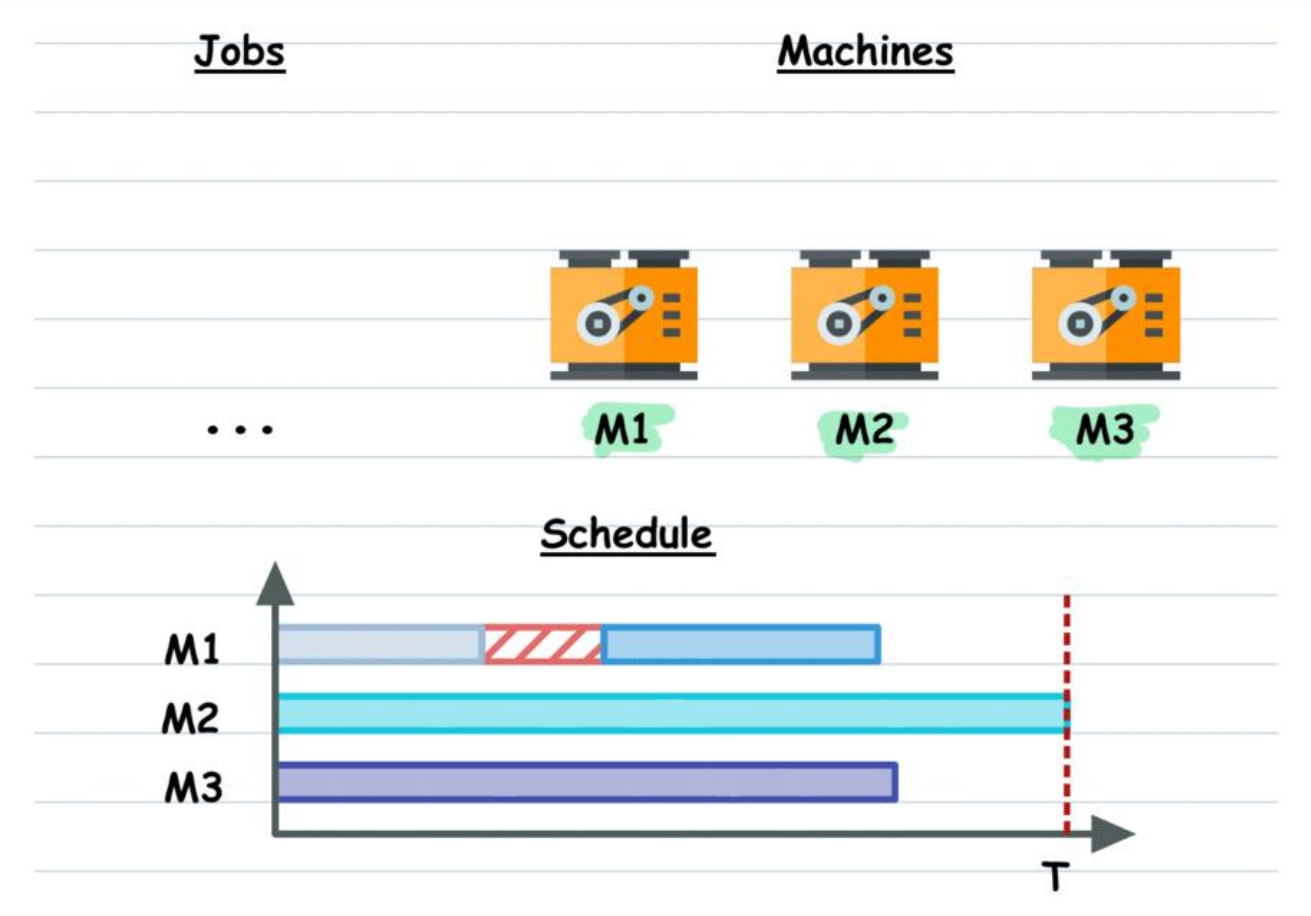
Job Scheduling



Job Scheduling



Job Scheduling



Job Scheduling

Job ID	Profit	Deadline
3	40	2
4	30	2
1	20	4
2	10	1

-1	-1	-1	-1

Job Scheduling

Job ID	Profit	Deadline
3	40	2
4	30	2
1	20	4
2	10	1

-1	3	-1	-1

Profit = 40

Job Scheduling

Job ID	Profit	Deadline
3	40	2
4	30	2
1	20	4
2	10	1

4	3	-1	-1

Profit = 40 + 30

Job Scheduling

Job ID	Profit	Deadline
3	40	2
4	30	2
1	20	4
2	10	1

4	3	-1	1

Profit = 40 + 30 + 20

Job Scheduling

Job ID	Profit	Deadline
3	40	2
4	30	2
1	20	4
2	10	1

4	3	-1	1

Profit = 40 + 30 + 20 + 0*

*cannot be performed

Düğüm Kapsama

- Kapsama Problemi:
  - Bir çizgenin tüm kenarlarını kapsayacak en az sayıda düğümü seçme.
- Amaç:
  - Tüm kenarları en az düğümle kapsayan kümeyi bulmak.

Düğüm Kapsama

- Çizge:
  - Düğümler: A, B, C, D
  - Kenarlar: (A, B), (A, C), (B, C), (B, D), (C, D)
- Düğüm Kapsama Kümesi:
  - {A, B, C}

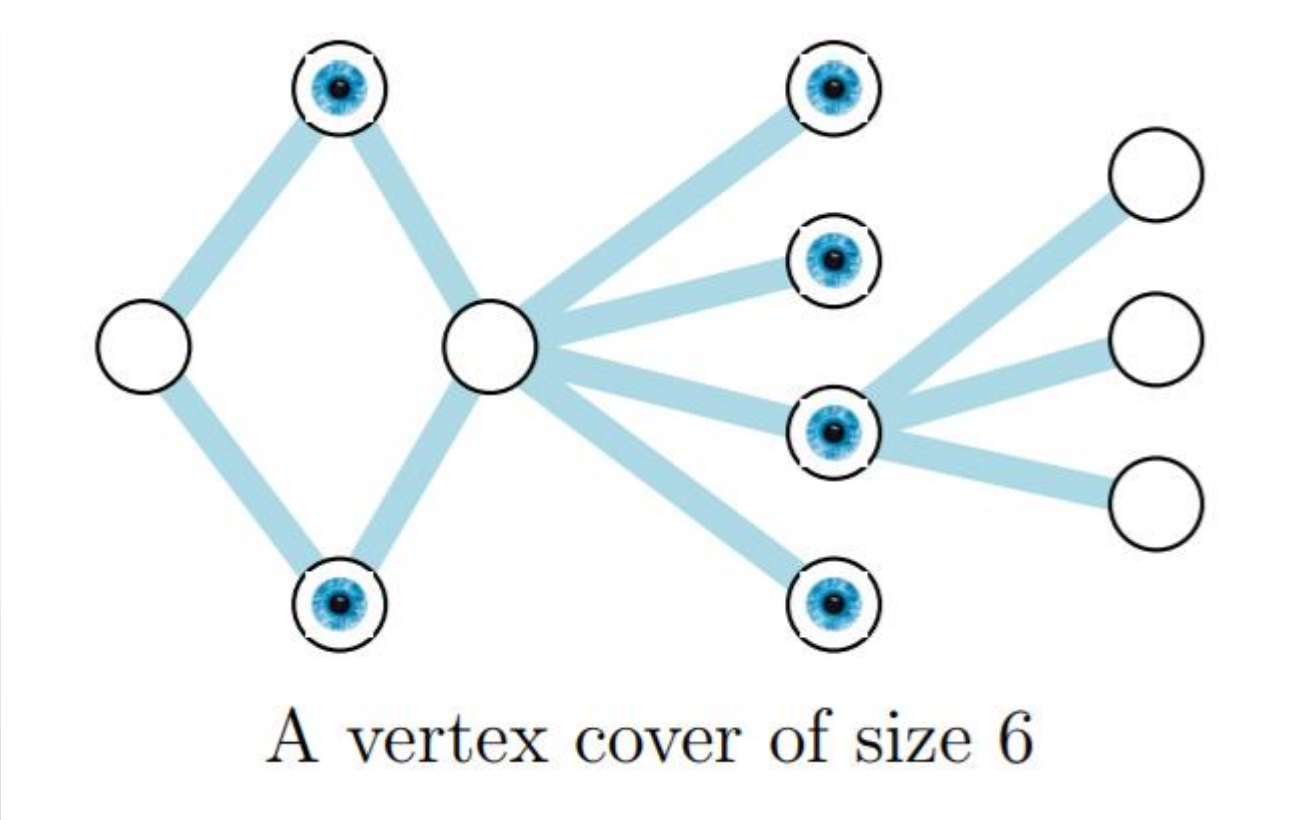
Çözüm

- Tam Çözüm:
  - Tüm olası düğüm kümelerini kontrol etmek.
- Yaklaşık Çözümler:
  - Greedy algoritmalar, Approximation algoritmaları, ...
- Optimizasyon Araçları:
  - Lineer Programlama, Branch and Bound, ...

Düğüm Kapsama Algoritması

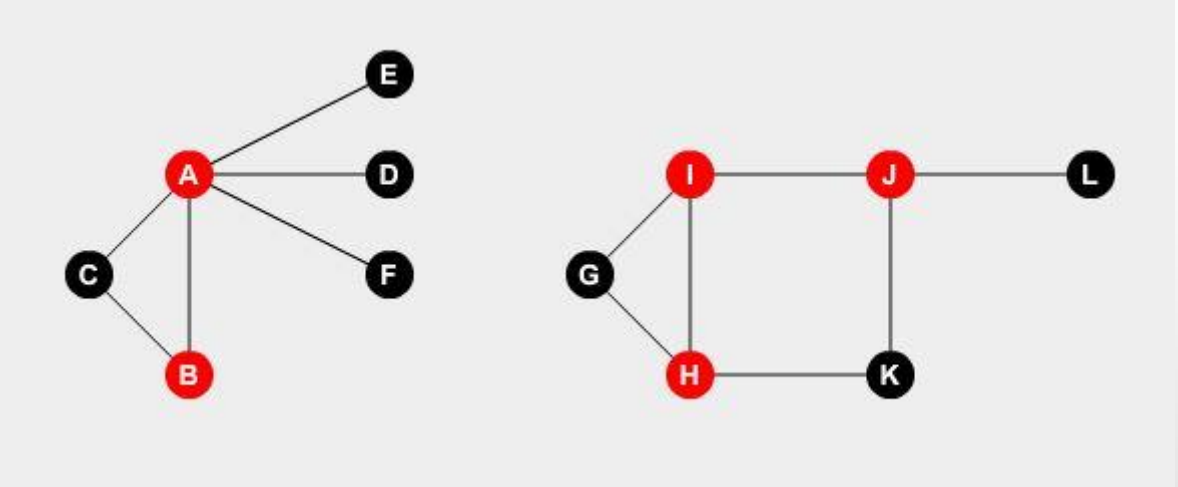
- Başlangıçta boş bir kapsama kümesi oluştur.
- Bir kenarı kapsayan düğümlerden birini seç ve kapsama kümesine ekle.
- Tüm kenarlar kapsayana kadar tekrarla.

Vertex Cover

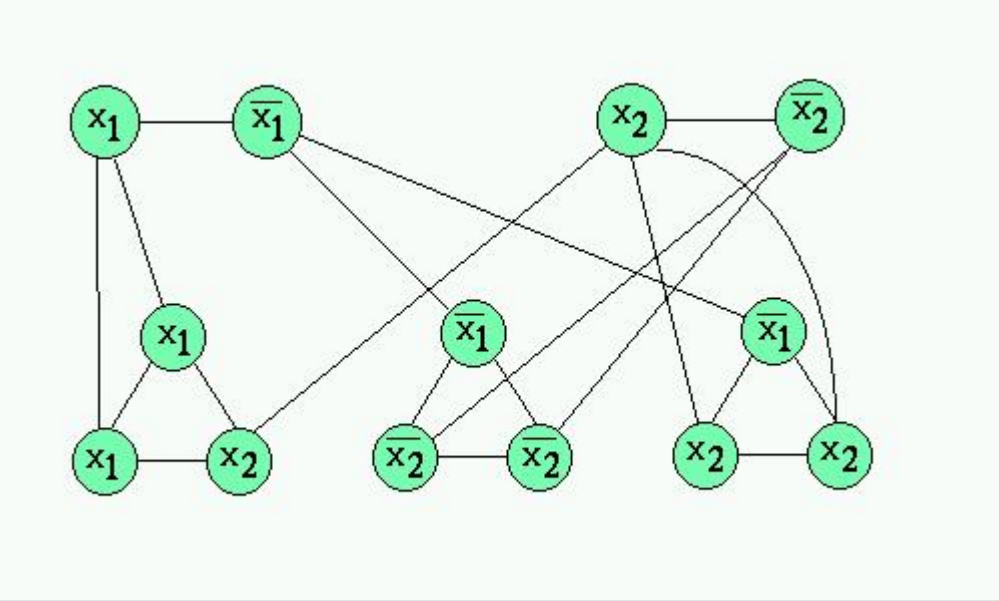


A vertex cover of size 6

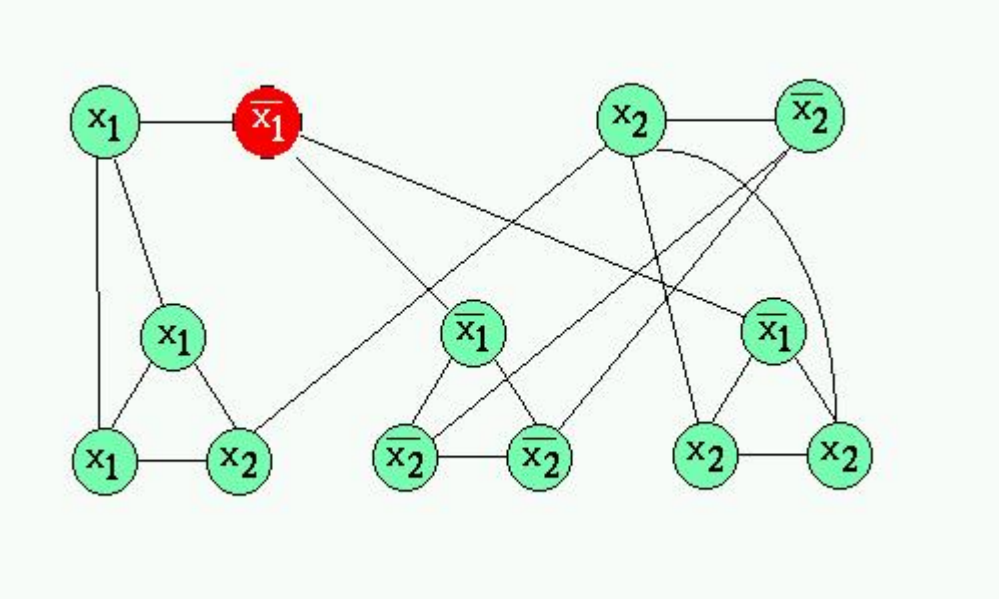
Vertex Cover



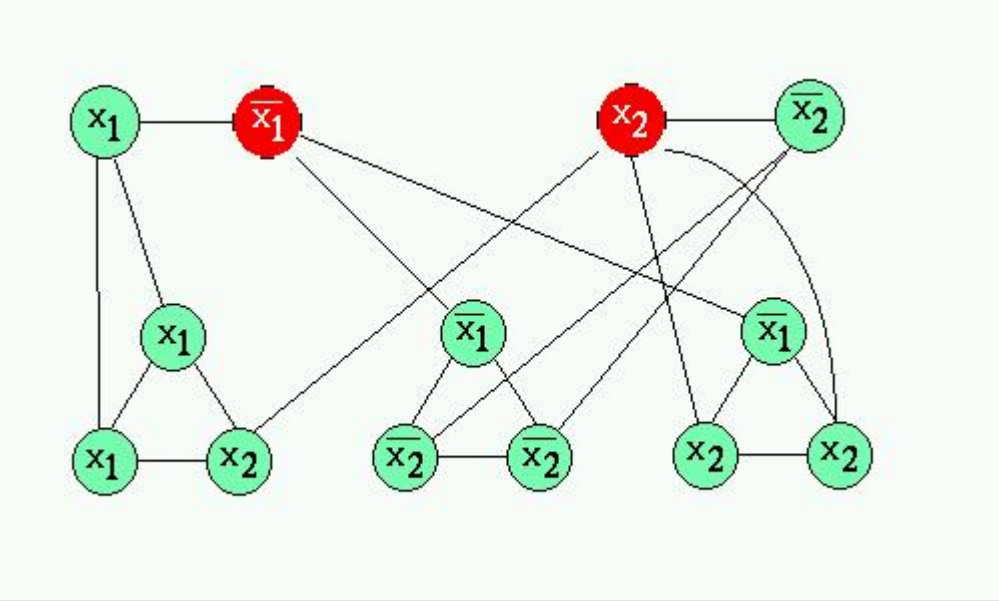
Vertex Cover



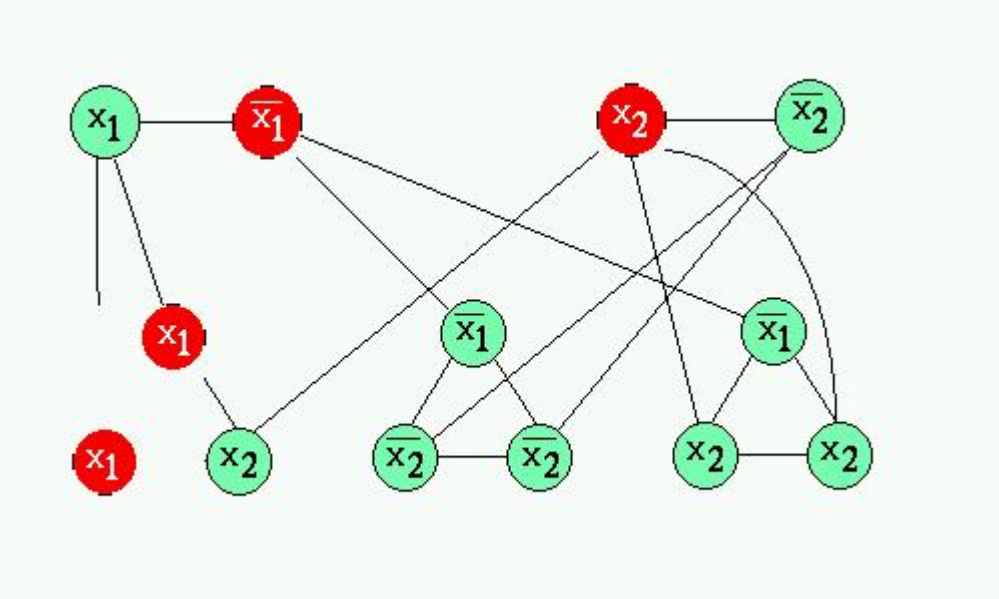
Vertex Cover



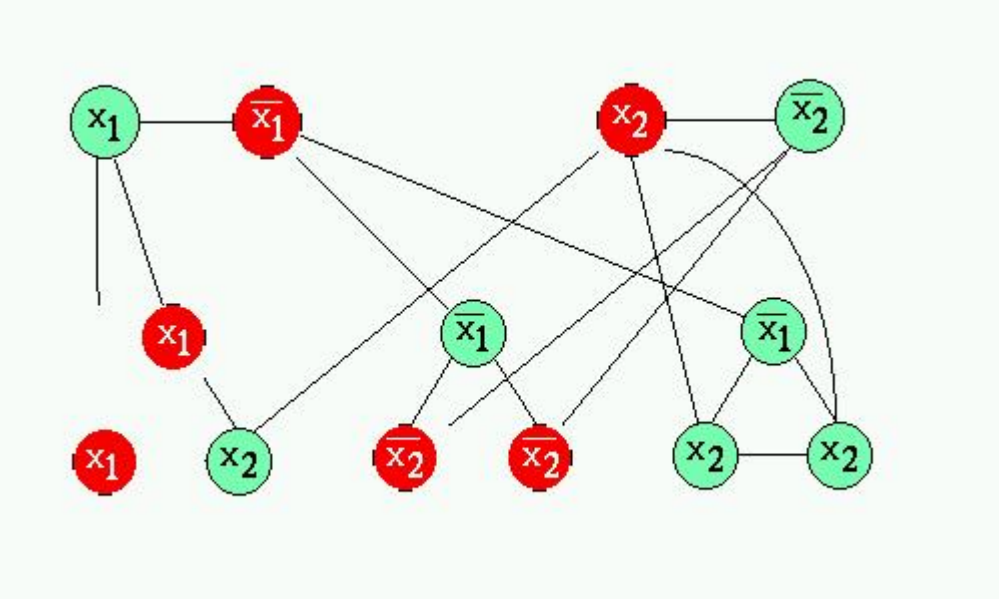
Vertex Cover



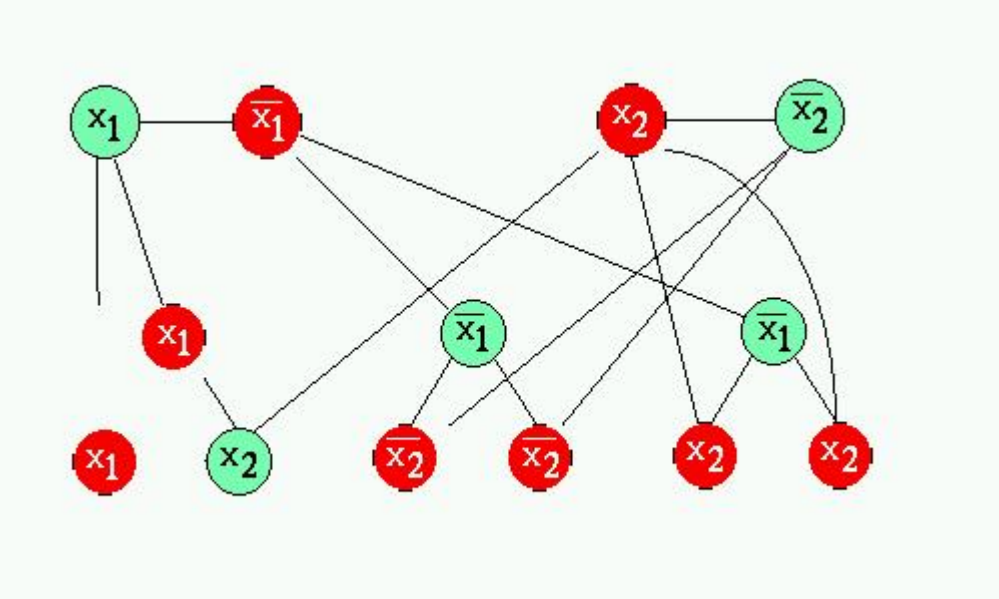
Vertex Cover



Vertex Cover



Vertex Cover



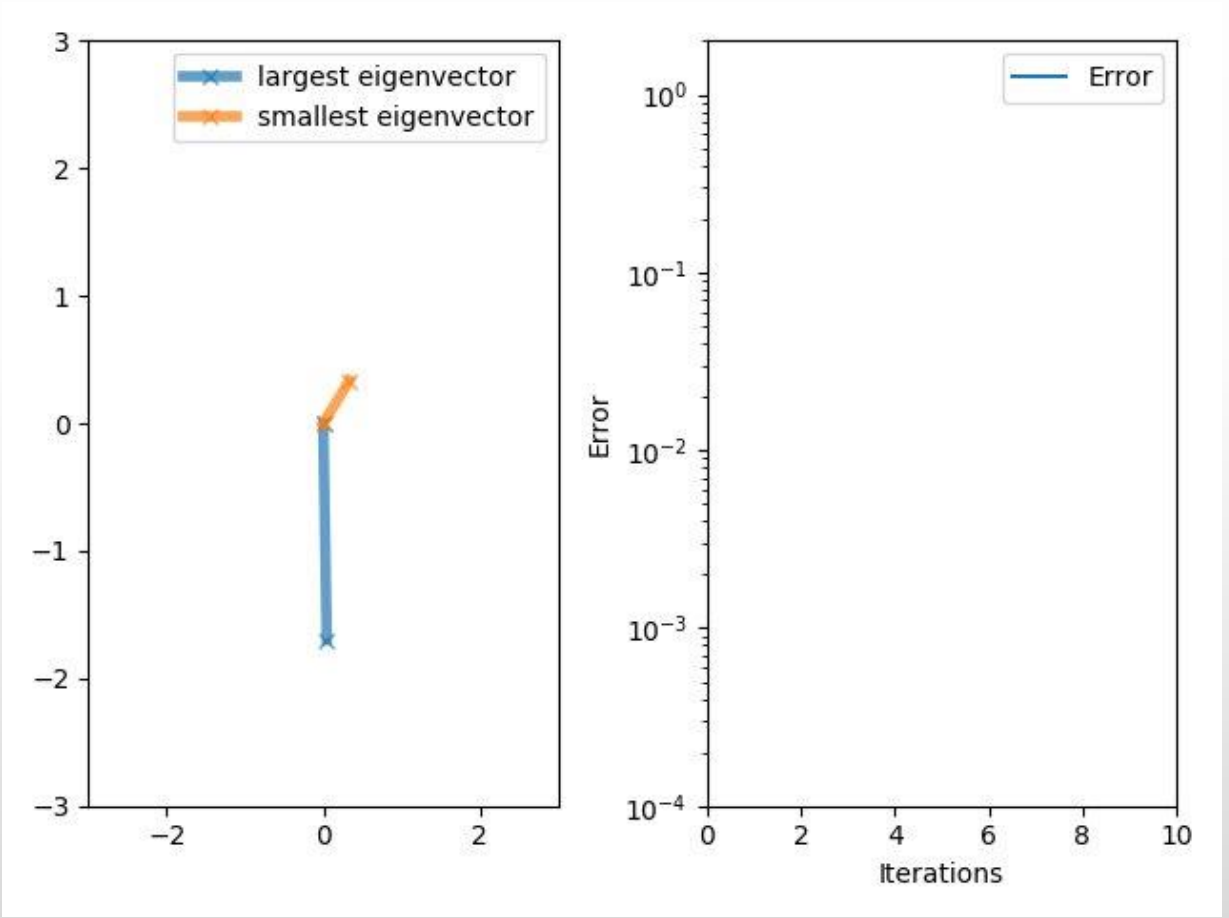
Power Iteration Algoritması

- Bir matrisin en büyük (mutlak değeri en büyük) özdeğerini ve buna ait bir özvektörü yaklaşık olarak hesaplamak için kullanılan iteratif bir yöntem.
- Bir matrisin özdeğerlerinin kesin bir şekilde hesaplanması yerine, sadece en büyük özdeğer ve buna ait bir özvektörün yaklaşık bir tahminini sağlar.
- Büyük boyutlu matrislerde etkilidir.

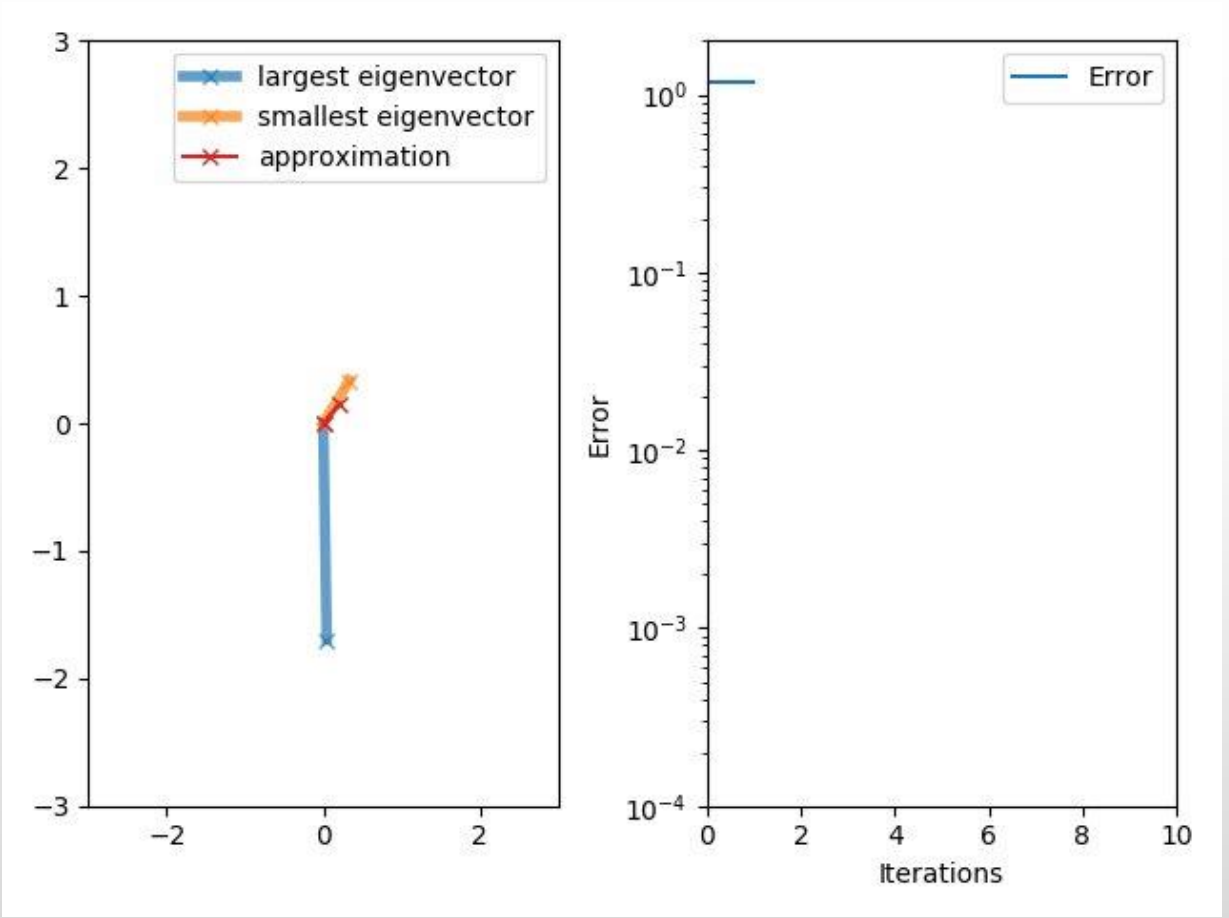
Power Iteration Algoritması

- İlk olarak, bir matris belirlenir ve rastgele bir başlangıç özvektörü seçilir.
- Başlangıç özvektörü, matrisle çarpılarak yeni bir özvektör elde edilir.
- Yeni vektör, normu (uzunluğu) 1'e normalleştirilir.
- Bu işlem, istenilen değer kez tekrarlanır.
- Matrisin en büyük özdeğeri ve bu özdeğere ait özvektör bulunmuş olur.

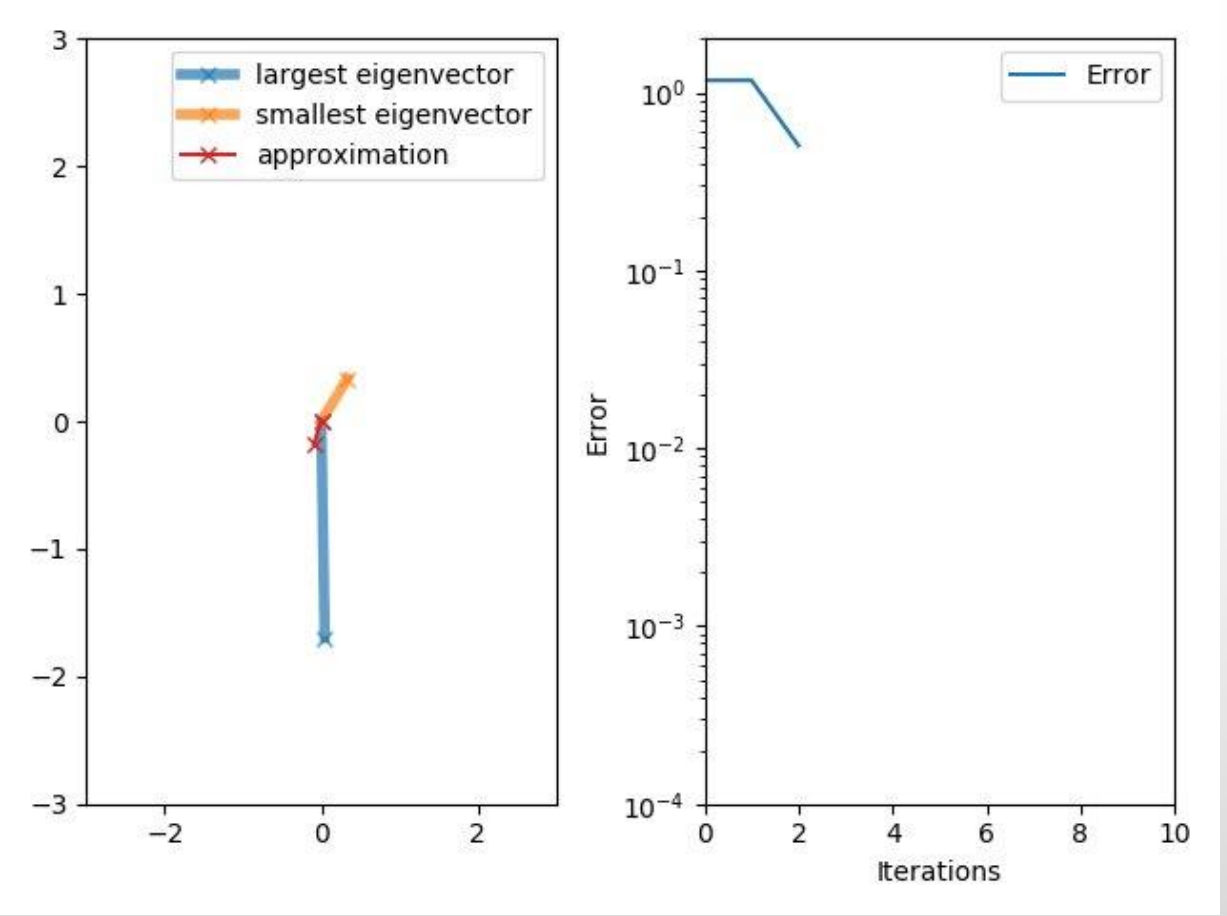
Power Iteration Algoritması



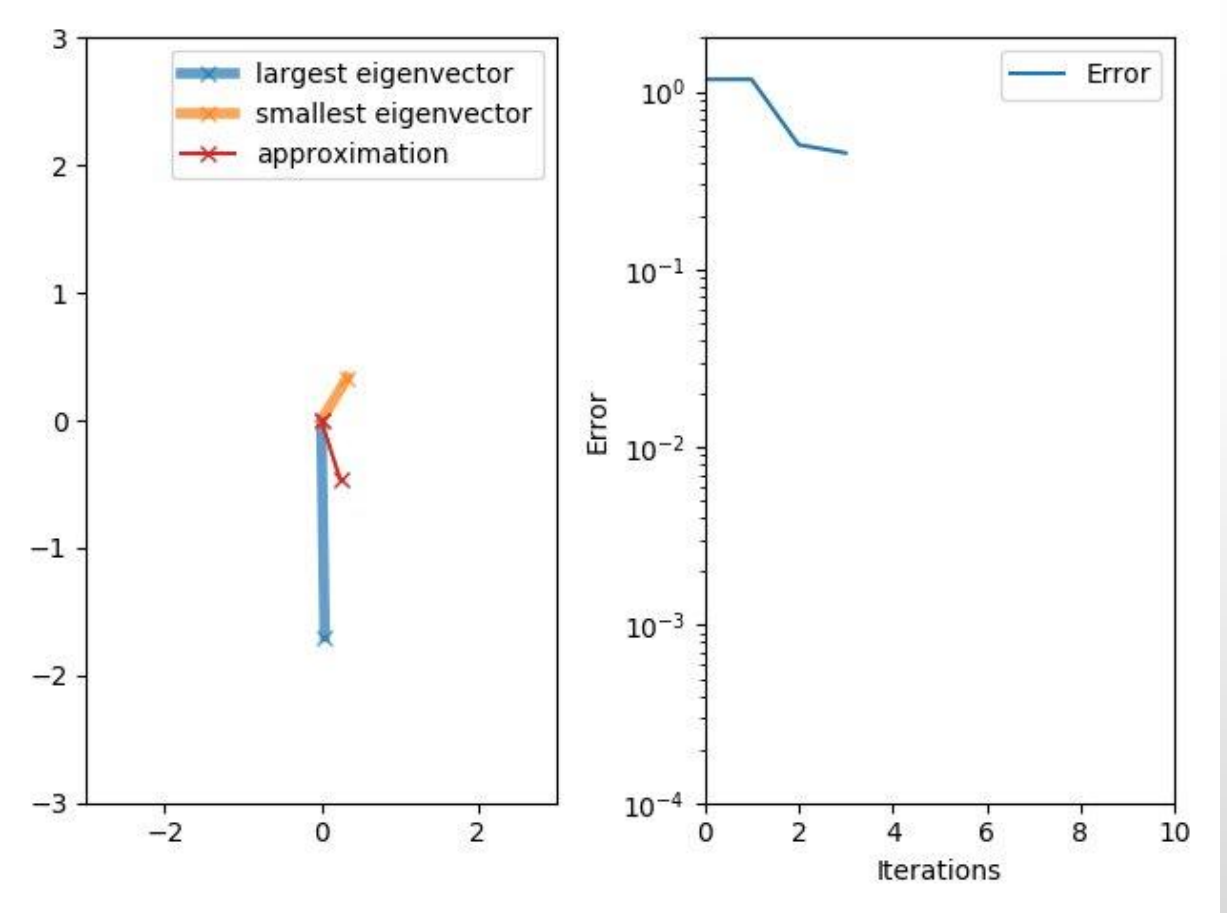
Power Iteration Algoritması



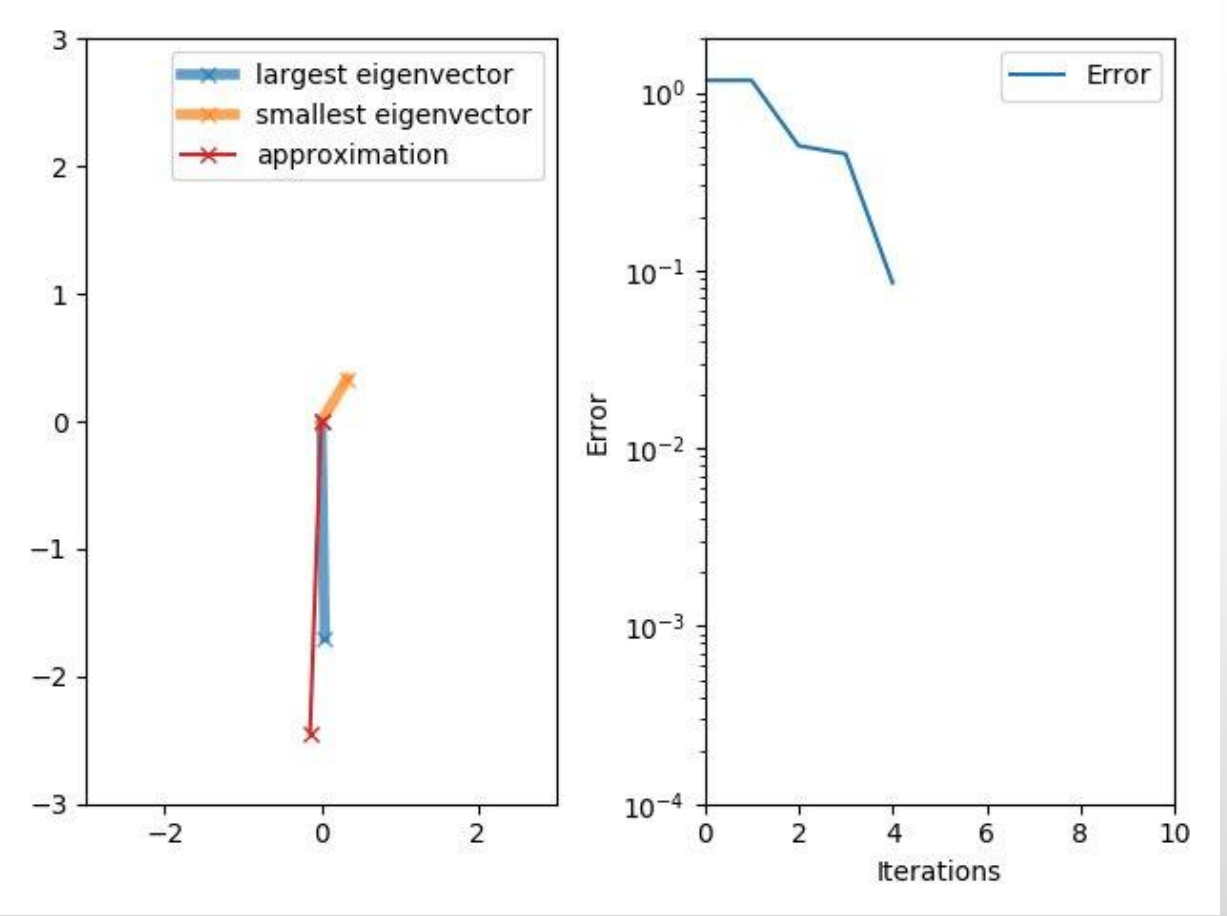
Power Iteration Algoritması



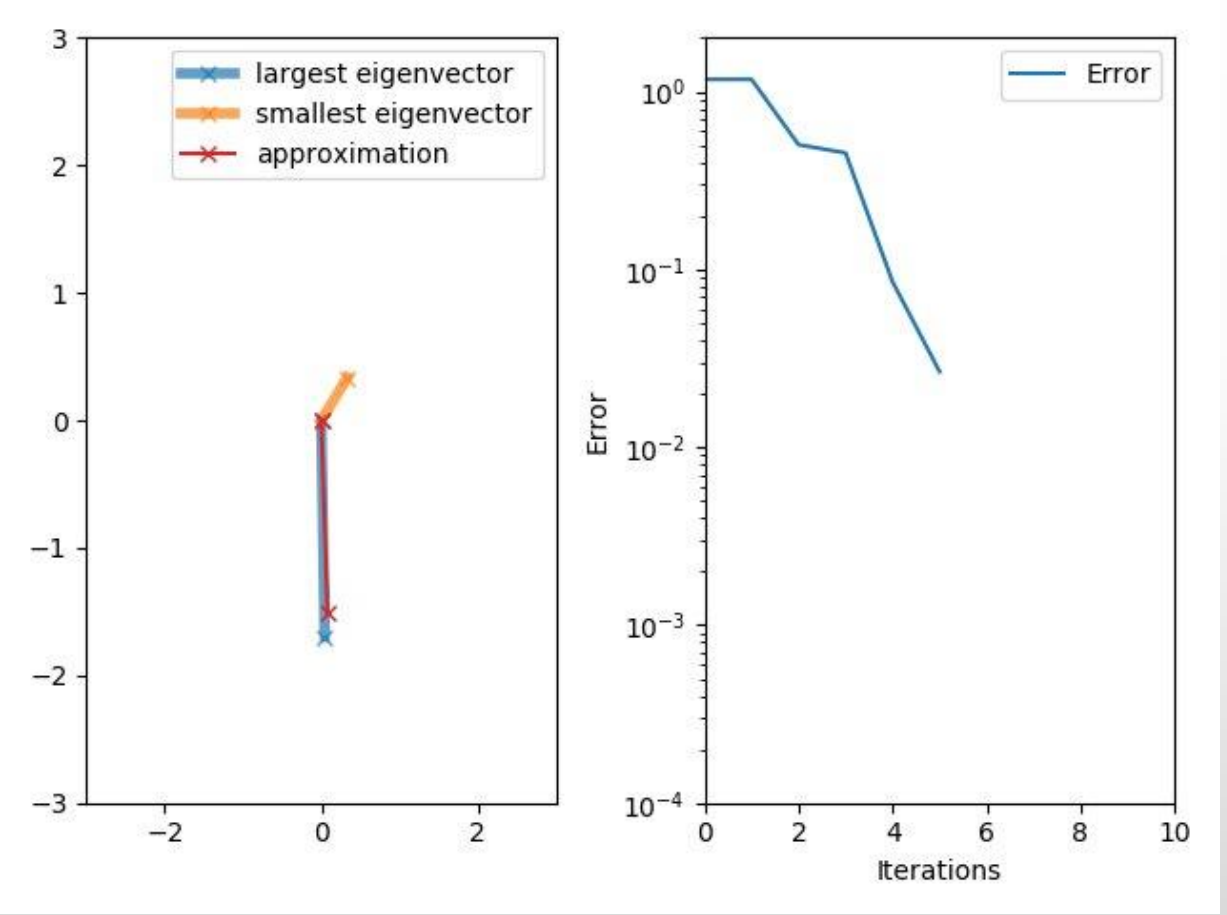
Power Iteration Algoritması



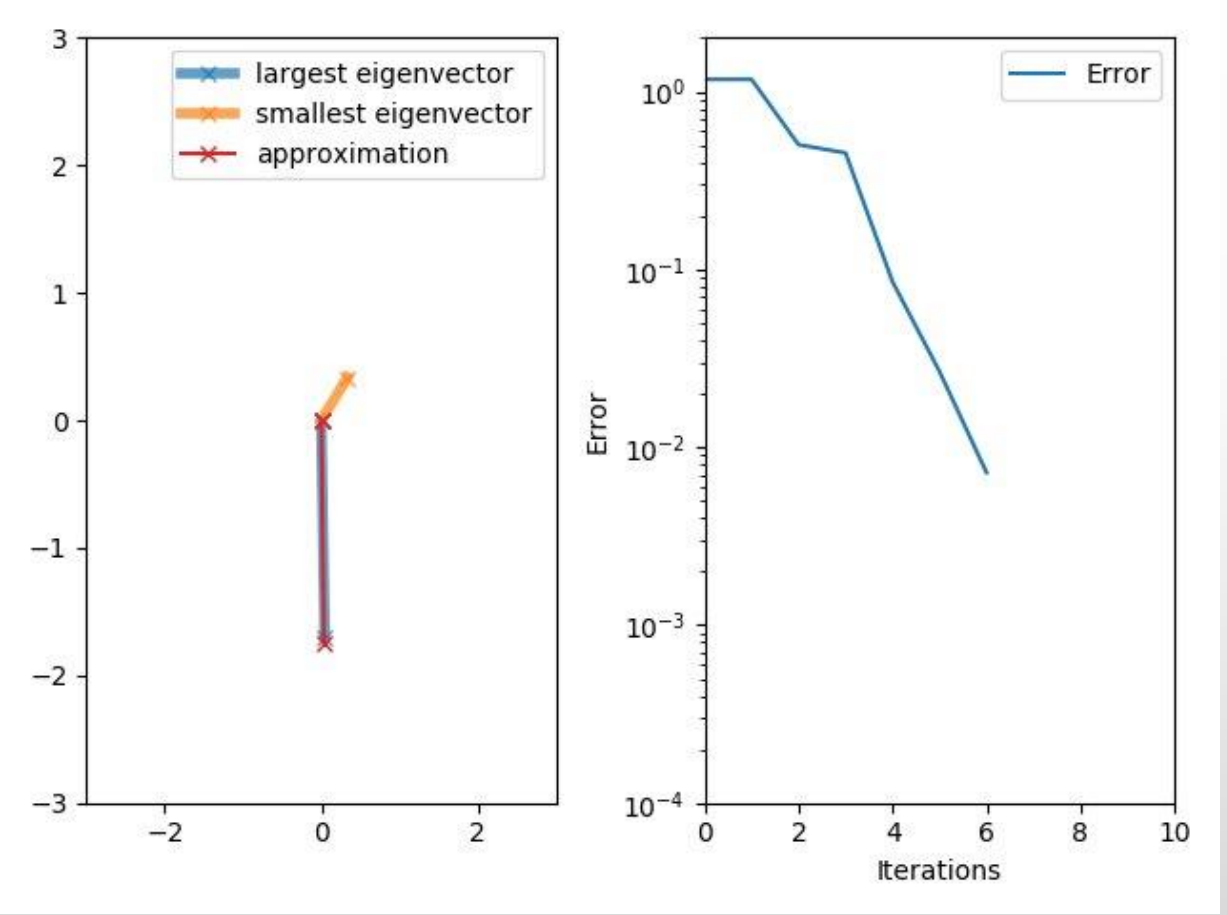
Power Iteration Algoritması



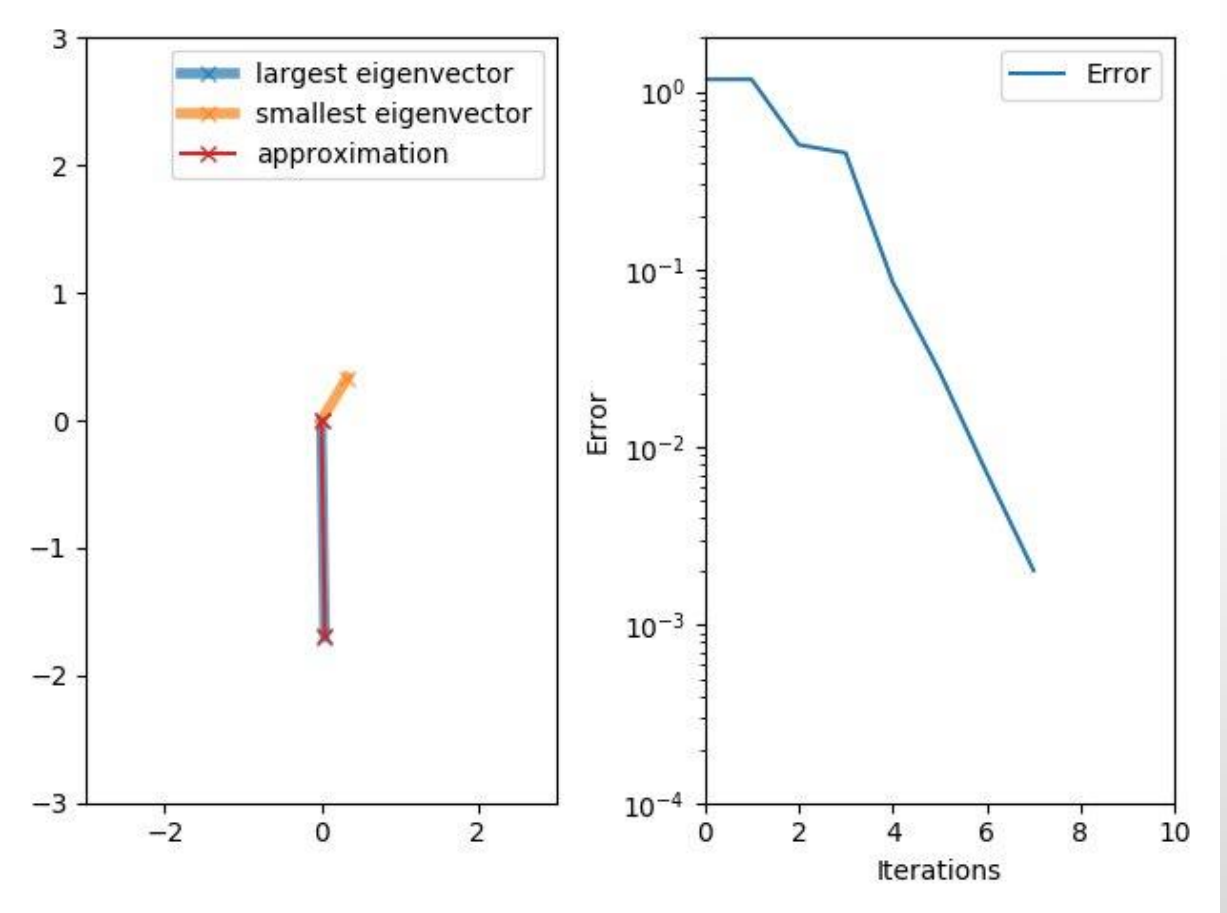
Power Iteration Algoritması



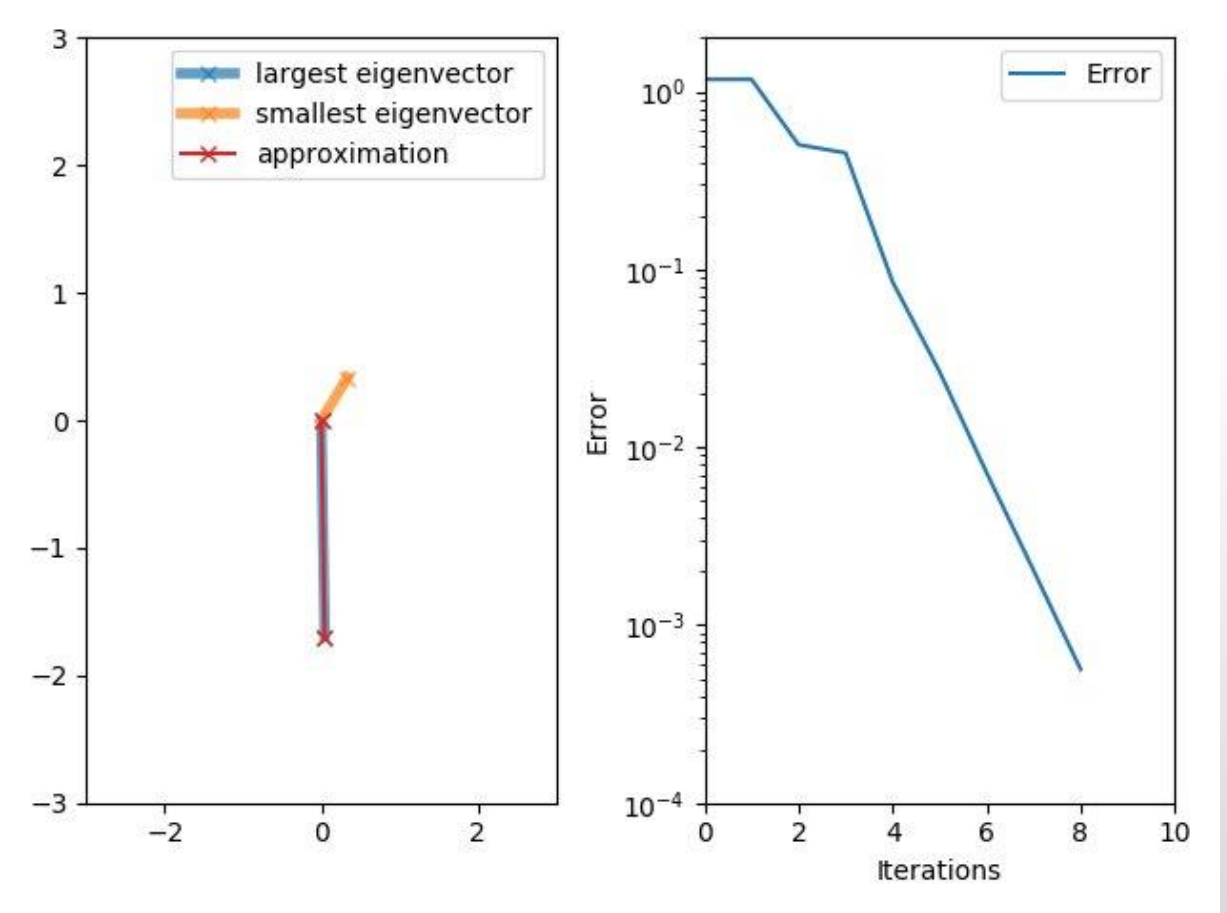
Power Iteration Algoritması



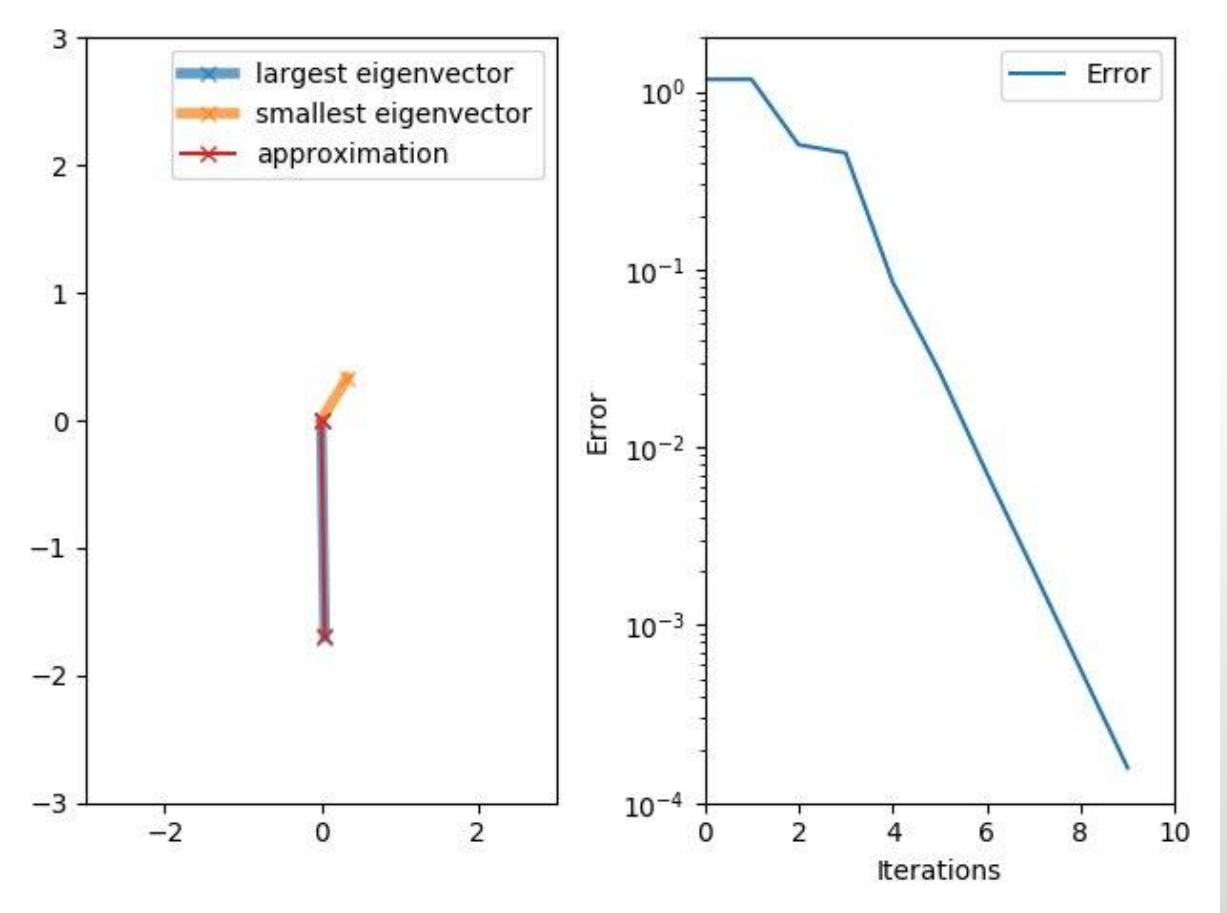
Power Iteration Algoritması



Power Iteration Algoritması



Power Iteration Algoritması



SON