Evrimsel Hesaplama ve Uygulamaları

1.HAFTA

Çözülecek Problemler

Çözülecek Problemler

- Mühendisler, bilgisayar bilimcileri tarafından ele alınan problemler
- Problemler ve Problem Çözücüler
- Problem sınıfları ve farklı problem sınıflandırma yolları
- Yapay zekanın ilişkilendirilebildiği tipik problemler
  - Bulmacalara benzer (ünlü zebra yapbozu)
  - Sayısal problemler (bir şehirden diğer şehre giden en kısa yol)
  - Örüntü çıkarma (yeni bir müşterinin cinsiyeti, yaşı, adresi vb. dikkate alındığında kitapçıdan ne alacağına karar vermek)

Çözülecek Problemler

- Problemler farklı şekillerde sınıflandırılabilir:
  - Black box model
  - Search problems
  - Optimization vs constraint satisfaction
  - NP problems

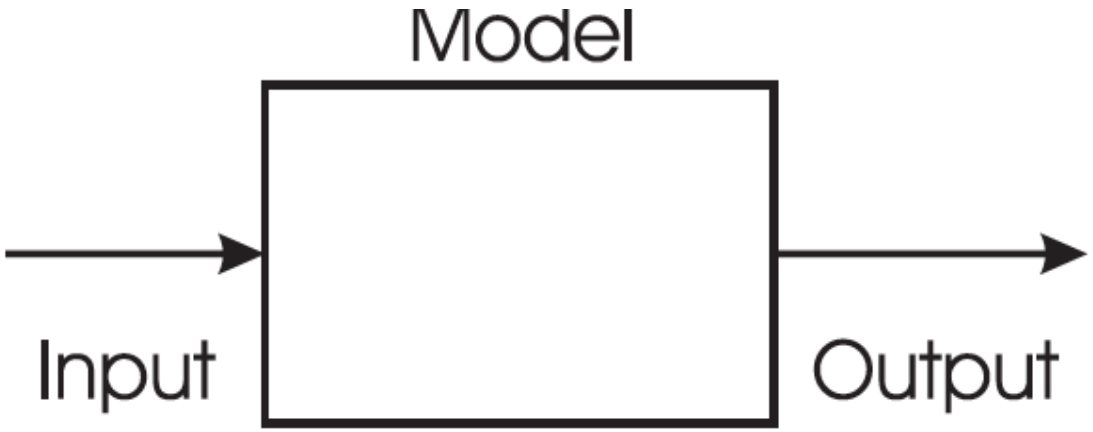
Kara Kutu Modeli

- Bilgisayar sistemlerindeki modele dayanmaktadır.
  - Sistem ilk olarak bir insan, sensör ya da başka bir bilgisayardan girdi bekler.
  - Girdi geldiğinde detaylarına çok hakim olunmayan (bu nedenle kara kutu) bazı hesaplama modelleri ile bilgiyi işler.
    - Ardışık konumlar listesinden toplam yolu hesap eden bir formül
    - Bazı meteorolojik girdi verileri ile yağmurun gerçekleşme olasılığını tahmin eden istatistiksel bir araç
  - Bilgi işlendikten sonra bazı çıktılar sunar.
    - Ekranda bir mesaj, dosyaya değerleri yazma, bir motora veri gönderme vb.
- Uygulamaya bağlı olarak;
  - Bir ya da daha fazla girdi
  - Basit ya da karmaşık model
- Modeli bilmek herhangi bir veriye göre çıktıyı hesaplama anlamına gelir.

Kara Kutu Modeli

- Bazı somut örnekler;
  - Uçak kanatlarının tasarımı
    - Girdi: Önerilen bir kanat şeklinin tarifi
    - Model: Direnç ve kaldırma katsayılarını tahmin etmek için karmaşık akış dinamiklerine ait denklemler
    - Çıktı: Bu tahminlerin değerleri
  - Akıllı evler için ses kontrol sistemi
  - Taşınabilir bir müzik çalar

Kara Kutu Modeli

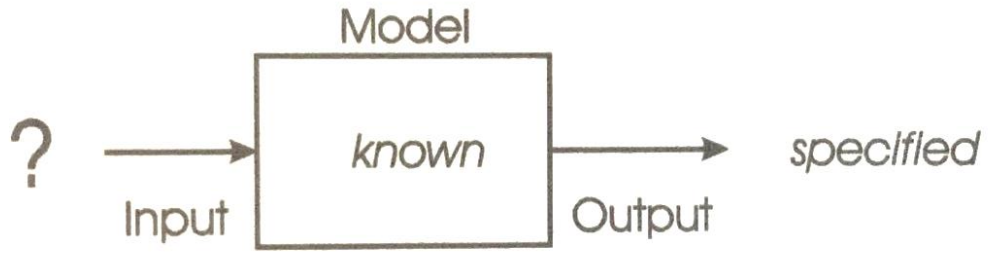


- "Kara Kutu" modeli 3 bileşen içermektedir.

- Bu bileşenlerden biri bilinmediğinde: yeni problem tipi
  - Optimization
  - Modelling
  - Simulation

Kara Kutu Modeli - Optimizasyon

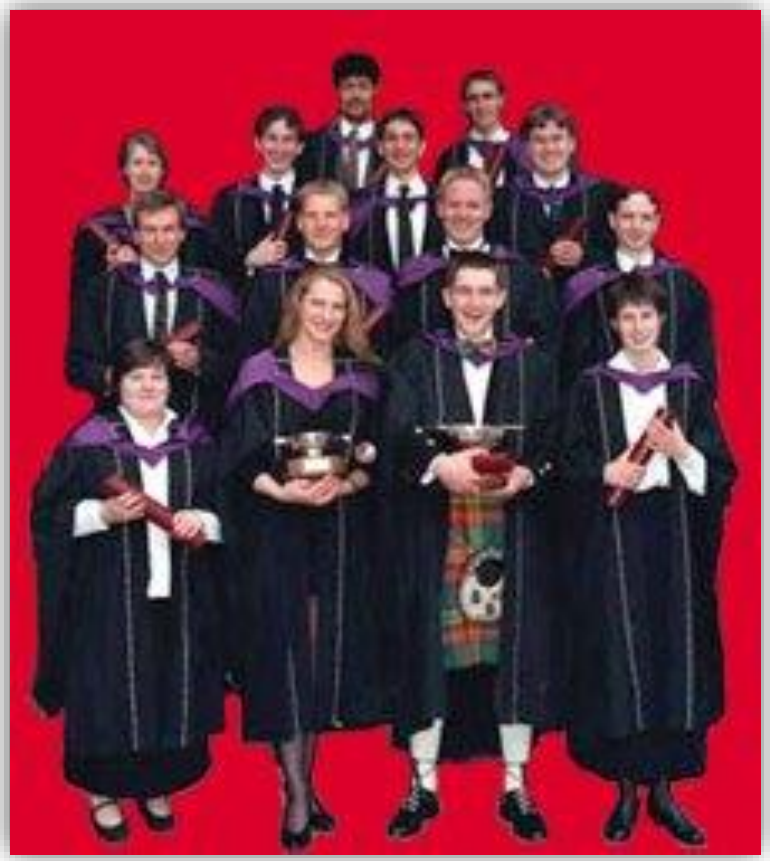
- Model and desired output is known, task is to find input



- Examples:
  - Time tables for university, call center, or hospital
  - Design specifications
  - Traveling salesman problem (TSP)
  - Eight-queens problem, etc.

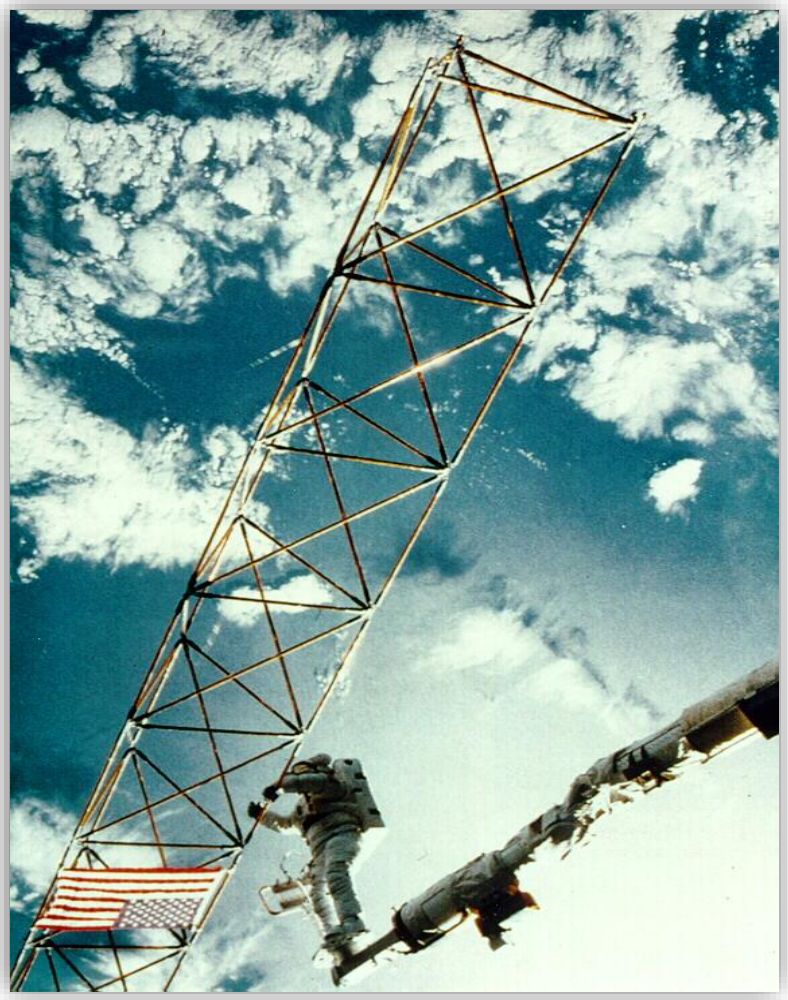
Optimisation Example 1: University Timetabling

- Enormously big search space
- Timetables must be *good*
- “Good” is defined by a number of competing criteria
- Timetables must be feasible
- Vast majority of search space is infeasible



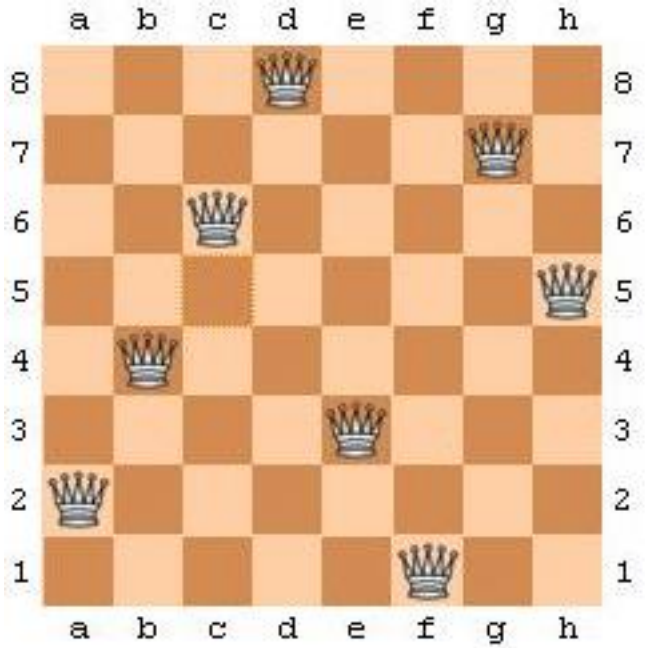
Optimization Example 2: Satellite structure

- Optimised satellite designs for NASA to maximize vibration isolation
- Evolving: design structures
- Fitness: vibration resistance
- Evolutionary “creativity”



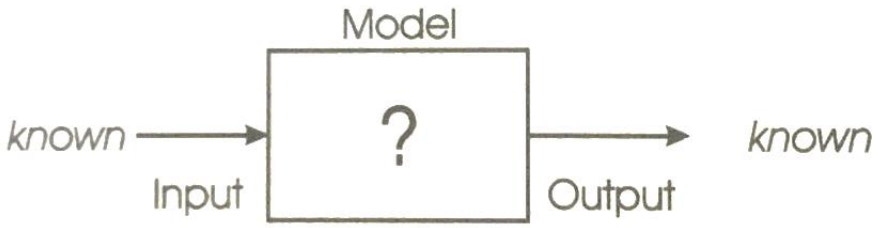
Optimization Example 3: 8 queens problem

- Given an 8-by-8 chessboard and 8 queens
- Place the 8 queens on the chessboard without any conflict
- Two queens conflict if they share same row, column or diagonal
- Can be extended to an n queens problem (n>8)



Kara Kutu Modeli – Modelleme (Modelling)

- İlgili girdi ve çıktı kümelerine mevcut ve her bilinen girdi için doğru çıktıyı üretecek bir model aranır.



- Not: Modelleme problemleri optimizasyon problemlerine dönüştürülebilir.

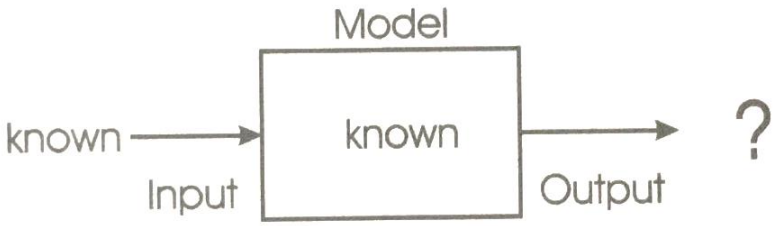
- Örnekler;
  - Evrimsel makine öğrenmesi
  - Borsa tahmini
  - Akıllı evler için ses kontrol sistemi

Modelling Example: Load Applicant Creditibility

- British bank evolved creditibility model to predict loan paying behavior of new applicants
- Evolving: prediction models
- Fitness: model accuracy on historical data

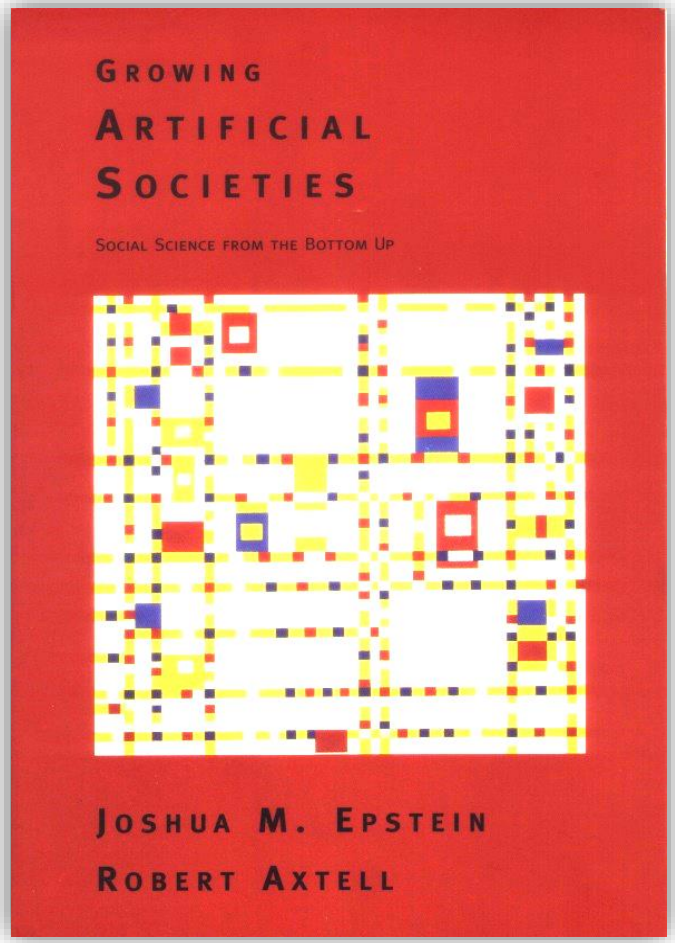
Kara Kutu Modeli: Simülasyon

- Belirli bir model var ve farklı girdi koşulları altında ortaya çıkan çıktıları bilmek istenmektedir.



- Sıklıkla tasarım ve sosyo-ekonomik içeriklerde karşılaşılmaktadır.
- Genellikle gelişen dinamik ortamlardaki “what-if (farzedelim, .. olursa ne olur)” sorularını yanıtlamak için kullanılır.
- Örnekler;
  - Evrimsel ekonomi, Yapay Yaşam
  - Hava tahmin sistemi
  - Yeni vergi sistemlerinin etki analizi

Simulation Example: Evolving Artificial Societies



- Simulating trade, economic competition, circuit design etc. to calibrate models
- Use models to optimise strategies and policies
- Evolutionary economy
- Survival of the fittest is universal (big/small fish)

Arama Problemleri

- Kara kutu görünümüm arkasındaki köklü bir varsayım: Hesaplama modeli taraflıdır (yönlü)
  - Girdilerden çıktılara doğru hesaplanır ve geri çevrilemez.

- Simülasyon, optimizasyon ve modellemeden farklıdır.
  - Tek ihtiyaç modellerin uygulanması- simulator

- Optimizasyon ve modelleme problemleri çok büyük olasılıklar arasında arama yapar.
  - Bu nedenle bu yolla çözülen problemler arama problemi olarak görülebilir.

- Arama Uzayı: istenen çözümü içeren tüm ilgilenilen nesnelerin birleşimi.
  - Arama alanı bir modele olası tüm girdilerden – Optimizasyon
  - Üzerinde çalıştığımız olguyu tanımlayan tüm olası hesaplama modellerinden - Modelleme

Arama Uzayı – Search Space

- Arama Uzayının Büyüklüğü
  - *n* adet şehir üzerinden farklı turlar için arama alanı ne kadar büyük?
  - Peki gerçek değerli parametreler için?

- Arama uzayının belirlenmesi, bir arama probleminin tanımlanmasındaki ilk adımdır.
- İkinci adım, bir çözümün tanımıdır.

- For optimisation problems such a definition can be
  - Explicit, e.g., a board configuration where the number of checked queens is zero
  - Implicit, e.g., a tour that is the shortest of all tours.

- For modelling problems, a solution is defined by the property that it produces the correct output for every input.
  - Transformed into optimization problems

- Benefit of classifying these problems: distinction between
  - search problems, which define search spaces, and
  - problem-solvers, which tell how to move through search spaces.

Optimizasyon vs Kısıt Memnuniyeti (Optimization vs Constraint Satisfaction)

- Optimize edilecek amaç fonksiyonu vs sağlanacak kısıtlar

- Amaç Fonksiyonu → Aday Çözümün Kalitesi
- Kısıt → Gereksinim sağlandı mı sağlanmadı mı?

- Amaç Fonksiyonu Örnekleri
  - Bir satranç tahtasındaki kontrolsüz kraliçelerin sayısı (maksimize edilecek);
  - Belirli bir kümede her bir şehri ziyaret eden bir turun uzunluğu (en aza indirilmesi);
  - Belirli bir model m (maksimize edilmek üzere) tarafından doğru şekilde etiketlenmiş bir koleksiyondaki görüntülerin sayısı.

- Çözümler bazı kısıtlara tabi olabilir:
  - Bir satranç tahtasında sekiz kraliçe yapılandırmasını bulun, öyle ki iki kraliçe birbirini kontrol etmiyor.
  - Seyahat eden bir satıcı için, X şehrinin Y kentinden sonra ziyaret edileceği şekilde minimum uzunlukta bir tur bulun.

Optimizasyon vs. Kısıt Memnuniyeti

- İkisi kombine edildiğinde ;

	Objective function	
Constraints	**Yes**	**No**
Yes	Constrained optimisation problem	Constraint satisfaction problem
No	Free optimisation problem	No problem

- Bahsedilen örnekler hangi kategoriye uyar?
- Kısıtlama problemleri optimizasyon problemlerine dönüştürülebilir.
- 8-Vezir problemini bir FOP / CSP / COP'a nasıl formüle edebiliriz?

Optimizasyon vs Kısıt Memnuniyeti

- Gezgin satıcı problemi → FOP
- 8- Vezir problemi → CSP
- X şehrinin Y kentinden sonra ziyaret edilme kısıtı ile minimum tur → COP
- CSP → optimizasyon problemine dönüştürülebilir.
  - Modelleme problemlerinin optimizasyon problemine dönüştürülme hilesi ile benzerdir.
  - Eşleşmeyen vezir çifti sayısı amaç fonksiyonu olarak ayarlanabilir.

8-Vezir Problemi

- Doğal dilde problem formülasyonu:
  - İki vezirin birbirini kontrol etmeyeceği şekilde bir satranç tahtasına sekiz vezir yerleştirin.

- Resmi Tanımlamalar:
  - FOP
    - 8 vezirli tüm yerleşim kombinasyonları → S arama uzayı
    - Verilen yerleşimdeki boş vezir sayısını kontrol eden amaç fonksiyonu ***f*** **,** (*s ϵ S, f(s)=8*)
  - CSP
    - FOP ile aynı arama uzayı
    - İki vezirin birbirini kontrol etmeme kısıtı *ϕ(s) = true*
  - COP
    - Kısıtlarla ile birlikte farklı arama uzayı
    - Aynı satır ve sütundan birden fazla vezir olamaz. Arama uzayı bu şekilde daraltılabilir. *ϕ'(s)*
    - Sadece diyagonal çakışmaları kontrol eden yeni amaç fonksiyonu ***g***
    - ***g=0*** **,** *ϕ' (s) = true*

Değerlendirme

- Bu örnekler, bir sorunun doğasının göründüğünden daha az belirgin olduğunu göstermektedir.
- Aslında, hepsi problemi nasıl biçimlendirmeyi seçtiğimize bağlıdır.
- Hangi formalizasyonun tercih edileceği tartışılacak bir konudur.
- Bazı formalizasyonların daha doğal olduğu veya soruna diğerlerinden daha uygun olduğu söylenebilir.
  - 8-vezir problemi doğası gereği CSP olarak tercih edebilir ve diğer tüm formalizasyonları ikincil dönüşümler olarak düşünebilirsiniz.
  - Trafik işareti tanıma problemi de her şeyden önce bir modelleme problemi olarak düşünülebilir ve pratik amaçlar için bir optimizasyon problemine dönüştürülebilir.
- Algoritmik düşünceler burada da büyük bir etkiye sahip olabilir.
  - Birinde özgür optimizasyon problemlerini iyi çözebilen, ancak kısıtlamalarla baş edemeyen bir algoritma varsa, problemleri serbest optimizasyon olarak resmileştirmek çok mantıklıdır.

NP Problems

- We only looked at classifying the problem, not discussed problem solvers
- This classification scheme needs the properties of the problem solver
- Benefit of this scheme: possible to tell how difficult the problem is
- Roughly speaking, the basic idea is to call a problem easy if there exists a fast solver for it, and hard otherwise.
- This notion of problem hardness leads to the study of computational complexity.
- Depending on the type of objects in the corresponding search space (S)
  - Continuous → Numerical optimization problems
  - Discrete(binary or integers) → Combinatorial optimization problems
    - Finite or -worst case- countably infinite

NP problems: Key notions

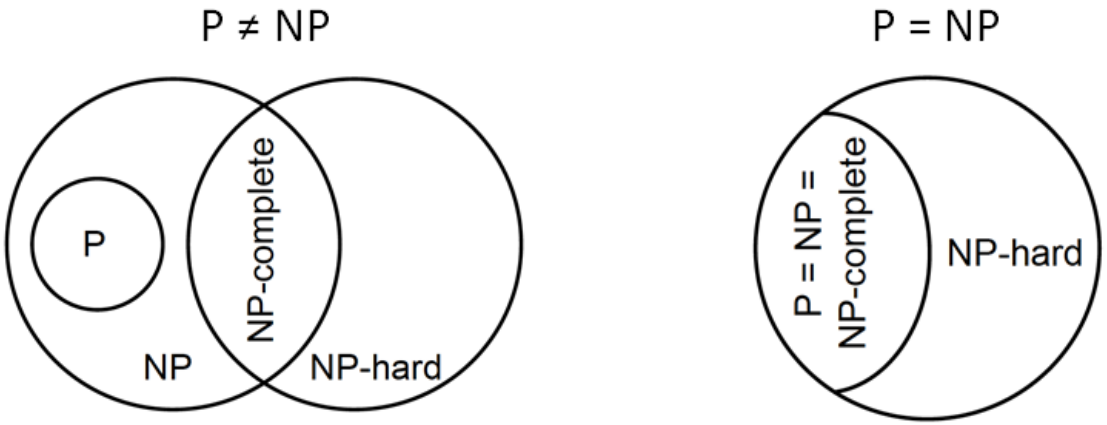
- Problem size: dimensionality of the problem at hand and number of different values for the problem variables
  - number of cities to visit, number of queens to place
- Running-time: number of operations the algorithm takes to terminate
  - Worst-case as a function of problem size
  - Polynomial, super-polynomial, exponential
- Problem reduction: transforming current problem into another via mapping

NP problems: Class

- The difficultness of a problem can now be classified:
  - Class P: algorithm can solve the problem in polynomial time (worst-case running-time for problem size n is less than F(n) for some polynomial formula F)
    - Minimum Spanning Tree
  - Class NP: problem can be solved and any solution can be verified within polynomial time by some other algorithm (P subset of NP)
    - Subset-Sum Problem
  - Class NP-complete: problem belongs to class NP and any other problem in NP can be reduced to this problem by an algorithm running in polynomial time
    - Traveling Salesman Problem
  - Class NP-hard: problem is at least as hard as any other problem in NP-complete but solution cannot necessarily be verified within polynomial time
    - Halting Problem

NP problems: Difference between classes

- P is different from NP-hard
- Not known whether P is different from NP



- If this (P=NP) were to be the case then the implications would be enormous for computer science and mathematics as it would be known that fast algorithms must exist for problems which were previously thought to be difficult.
- For now: use of approximation algorithms and metaheuristics

Kaynaklar

- A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing, Natural Computing Series, Springer