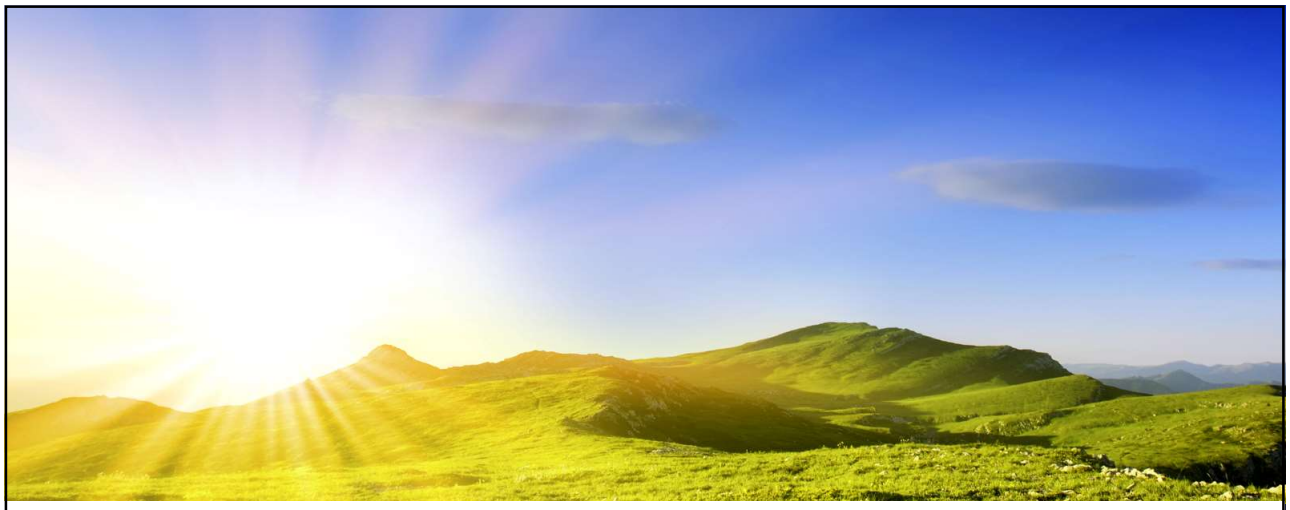
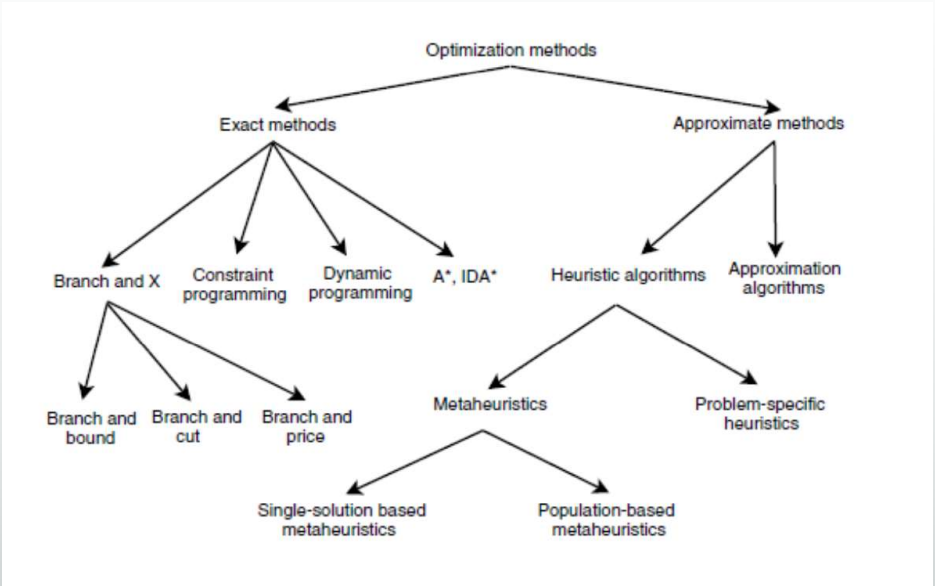


Sezgisel Metotlar
-2-

Kombinatoryal Optimizasyon Problemleri için Çözüm Yöntemleri



KOP için Çözüm Yöntemleri

- Kombinatoryal optimizasyon problemlerinin çözümünde kullanılan algoritmalar iki sınıfa ayrılır:
  - Kesin algoritmalar (Exact algorithms)
  - Yaklaşık algoritmalar (Approximate algorithms)
- Kesin algoritmalar, eniyi çözümü garanti eden algoritmalardır.
- Yaklaşık algoritmalar, eniyi çözümü garanti etmezler ancak makul çözüm zamanı içerisinde probleme iyi bir çözümü bulmaya çalışırlar

Kesin Algoritmalar:

- Doğrusal Programlama
- Dinamik Programlama
- Dal-sınır (branch-bound)
- Geri-izleme (backtracking), vd.

KOP için Çözüm Yöntemleri

- NP-zor problemlerde eniyi çözümü bulmak için gerekli zaman, problem boyutuna bağlı olarak üstel artış gösterir.
- Örneğin, bilinen kesin algoritmalar ile karesel atama problemine eni iyi çözüm en fazla 30 nesne için elde edilebilmektedir.

Yaklaşık (Approximate) Algoritmalar:

Bu algoritmalar,

- sezgisel algoritmalar
- yaklaşım algoritmaları

- Sezgisel algoritmalar, eniyi çözümü garanti etmeksizin daha az çözüm zamanı ile eniyiye yakın iyi bir çözümü elde etmeyi hedeflerler.
- Sezgisel algoritmalar, polinom zamanda probleme iyi bir çözüm elde etmeye çalışırlar.

KOP için Çözüm Yöntemleri

- Yaklaşım (approximation) algoritmaları, ispatlanabilir çözüm kalitesi ve çözüm zamanı sınırlarını sağlayan algoritmalardır.
  - ε-yaklaşım algoritması, eniyi çözümün ε katından daha kötü olmayan çözüm üretir.

Sezgisel Yöntemler:

- Çözüm Kurucu Algoritmalar (Constructive Algorithms)
- Yerel Arama Algoritmaları (Local Search Algorithms)

Genel Amaçlı Arama Algoritmaları

- Tavlama Benzetimi (Simulated Annealing)
- Tabu Arama (Tabu Search)
- Değişken Komşu Arama (Variable Neighborhood Search)
- Genetik Algoritmalar (Genetic Algorithms)
- Karınca Kolonisi Optimizasyonu (Ant Colony Optimization)
- Kuş Sürüsü Optimizasyonu (Particle Swarm Optimization)

Çözüm Kurucu Algoritmalar

- Probleme iyi bir çözümü iteratif olarak kurar.
- Bu algoritmalar, başlangıçta bir çözüm parçasını seçer ve probleme tam bir çözümü elde edinceye kadar adım adım çözüm parçalarını kısmı çözüme ekler.
- Herhangi bir adımda hangi çözüm parçasının kısmı çözüme ekleneceğine
  - rassal
  - sezgisel

  bir kural ile karar verir.
- Genellikle, her adımda bir sezgisel fonksiyon ile tahmin edilen ve en büyük faydanın söz konusu olduğu çözüm parçası kısmı çözüme eklenir.

Çözüm Kurucu Algoritmalar

- Bu tur algoritmalar (sezgisel kural kullanan),
  - Açgözlü çözüm kurucu sezgisel (greedy constructive heuristic)
  - Açgözlü sezgisel (greedy heuristic)
- Farklı problemler için farklı çözüm kurucu algoritmalar geliştirilmiştir.

Gezgin Satıcı Problemi için
- En yakın komsu sezgiseli (Nearset Neighborhood Heuristic)
- En ucuz ekleme sezgiseli (Cheapest Insertion Heuristic)

Çizelgeleme Problemi için
- En kısa işlem zamanı (shortest processing time SPT)
- En erken teslim tarihi (earliest due date EDD)

GSP için En Yakin Komsu Sezgiseli (EYK)

- Gezgin Satıcı Problemi (GSP) için geliştirilmiş olan en basit çözüm kurucu sezgiseldir.

Bu sezgiselin isleyişi aşağıdaki gibi özetlenebilir:
- Başlangıçta rassal olarak bir şehir seçilir.
- Tam tur elde edilinceye kadar her adımda seçilen şehire en yakın şehir seçilerek kısmı çözüme eklenir.

EYK Sezgiseli için Algoritma

Adim 0: $J \leftarrow \{\}$, $J' \leftarrow \{1,2,3,\ldots,n\}$

Adim 1: Rassal olarak bir sehir sec (i) ve kismi cozume ekle.
$J \leftarrow J + \{i\}$,
$J' \leftarrow J' - \{i\}$,
$s \leftarrow i$

Adim 2: Tam tur elde edinceye kadar asagidaki adimlari tekrarla:

i. i'ye en yakin ziyaret edilmemis sehri sec (j) ve kismi cozume ekle.

$j \leftarrow \text{argmin}\{ d_{ij} \mid j \in J' \}$
$J \leftarrow J + \{j\}$,
$J' \leftarrow J' - \{j\}$,

ii. $i \leftarrow j$

Adim 3: $J \leftarrow J + \{s\}$

Adim 4: Toplam tur uzunlugunu hesapla.

Adim 5: Elde edilen turu ve toplam tur uzunlugunu rapor et.

GSP için En Yakin Komsu Sezgiseli (EYK)

Örnek:

GSP için En Yakin Ekleme Sezgiseli (EYE)

- Gezgin satıcı problemi için önerilmiş bir diğer çözüm kurucu sezgiseldir.

Bu sezgiselin isleyişi aşağıdaki gibi özetlenebilir:

- Başlangıçta bir şehir seçilir.
- Bu şehrin en yakın komsusu bulunarak bir alt tur oluşturulur.
- Tam tur elde edilinceye kadar aşağıdaki işlem tekrarlanır.
  - Alt turda olmayan tüm şehirler alt turdaki mümkün pozisyonlara yerleştirilerek tur uzunluğunda en az artışı sağlayan şehir ilgili pozisyona yerleştirilir.

EYE Sezgiseli icin Algoritma

Adim 0: $J \leftarrow \{\}$, $J' \leftarrow \{1,2,3,...,n\}$

Adim 1: Baslangic sehrini (i) rassal olarak sec.

Adim 2: i'ye en yakin ziyaret edilmemis sehri sec (j) ve alt tur olustur.

$i - j - i$

Adim 3: Secilmeyen tum sehirleri alt turda olasi pozisyonlara yerlestirerek tur uzunluklarini hesapla.

Adim 4: En az tur uzunluguna sahip alt turu sec.

Adim 5: Alt tura yeni giren sehri (k) ziyaret edilmemis sehirler kumesinden cikar

$J' \leftarrow J' - \{k\}$

Adim 6: Tam tur elde edilmemis ise ($J' <> \emptyset$) Adim 3'e degilse Adim 7'e git.

Adim 7: Elde edilen turu ve toplam tur uzunlugunu rapor et.

GSP için En Yakin Ekleme Sezgiseli (EYE)

Örnek:

Yerel Arama Algoritmalari (YAA)

- Yerel arama algoritmaları (YAA), en eski ve en kolay eniyileme yöntemlerinden birisidir.
- YAA, NP-zor sınıfında yer alan problemlere makul zamanda kaliteli çözümü bulmak için kullanılan genel bir yaklaşımdır.
- Bir başlangıç çözümü ile başlar ve yerel değişmeler ile bu çözümü iyileştirmeye çalışır.
- Bu algoritmaları uygulayabilmek için aday çözümlerin kümesi (S) üzerinde "komşuluk yapısının" tanımlanması gerekir.
- "Komşuluk yapısı", algoritmanın mevcut bir çözümden hareket edebileceği mümkün çözümlerin kümesini tanımlar.
- YAA ile bulunan çözüm, yerel değişimler açısından eniyi çözümdür. Ancak, bu çözüm global eniyi çözüm olmayabilir.

Yerel Arama Algoritmaları (YAA)

- YAA'nın genel isleyişi aşağıdaki tanımlanabilir.

YAA, bir başlangıç çözümü, $s_0 (s_0 \in S)$ ile başlar ve s_0'in komşuları arasında daha iyi bir çözüm için arama yapar. s_0 komşusu N(s_0) olan iyi bir çözüm (s) bulunmuş ise, arama yeni çözümün komşuları N(s) arasında devam eder. YAA, yerel eniyi çözüme ulaştığında sonlanır.

Bir YAA icin;

- başlangıç çözümü nasıl elde edilecek?
- iyi bir komşuluk yapısı nasıl seçilecek?
- komsular arasında arama nasıl yapılacak?

gibi soruların cevaplanması gerekir.

Yerel Arama Algoritmaları (YAA)

Başlangıç çözümü;

- rassal
- çözüm kurucu sezgisel

ile seçilebilir.

- Bu alternatifler için YAA'nın çözüm zamanı ve elde edilen çözümlerin kalitesi farklı olacaktır .

Bir diğer yaklaşım;

YAA farklı başlangıç çözümleri geliştirilir. Elde edilen yerel eniyi çözümler arasından eniyisi seçilir.

Yerel Arama Algoritmaları (YAA)

Komşuluk yapısının secimi:

Mevcut çözüm (s) etrafındaki komsuları aramak için uygun bir hareket mekanizmasının seçilmesi gerekir.

İki tur hareket söz konusudur:

- Basit hareket
- Karmaşık hareket

Basit hareket mekanizmasının
- uygulamak kolaydır
- bilgisayarda uygulanma zamanı azdır.

Ancak, bu tur mekanizmalar ile az sayıda komşu elde edilir.

Karmaşık hareket mekanizmasının
- uygulamak zordur
- bilgisayarda uygulama zamanı fazladır.

Ancak, basit hareket mekanizmasına göre daha fazla sayıda komsu elde edilir.

Yerel Arama Algoritmaları (YAA)

- Basit ve karmaşık hareketler arasında çözüm zamanı ve çözüm kalitesi açısından bir ödünleşim söz konusudur.
  - Karmaşık hareketler ile büyük sayıda komsu arandığından dolay çözüm zamanı artar. Ancak, iyi yerel çözümlere erişme şansı artar.
  - Basit hareketler ile komşuların aranması hızlıdır Ancak . Ancak, zamansız yakınsamaya neden olabilir.
- YAA'da ne tur hareket mekanizmasının kullanılacağına yapılan ön denemeler sonucunda karar verilir.

Komşular arasından bir çözümün seçimi için kullanılan iki kural vardır:

1) Eniyi iyileştirme kuralı (best improvement rule): Bu kural, mevcut çözümün amaç fonksiyonu değerinde en büyük iyileştirmeyi yapan komşu çözümü seçer.
2) İlk iyileştirme kuralı (first improvement rule): Bu kural, mevcut çözümün amaç fonksiyonu değerinde ilk iyileştirmeyi yapan komşuyu seçer.