

Yapay Zeka

Ders 6: Bilgi ile arama

Doç. Dr. Mehmet Dinçer Erbaş

Bolu Abant İzzet Baysal Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

Bilgi ile arama

- Önceki bölümde gördüğümüz üzere bilgisiz arama metotları
 - ~ sistematik şekilde yeni durumları oluşturur
 - ~ bu durumların hedef olup olmadığını test eder
 - ~ bu şekilde problemlere çözüm bulmaya çalışır.
- Ne yazık ki bu metotlar basit olmakla birlikte birçok problem için oldukça verimsizdir.
- Bu bölümde ise problem ile alakalı bilgiler kullanan bilgi arama metotlarını göreceğiz.
 - ~ Bu algoritmalar bilgi kullanarak daha verimli şekilde problemleri çözebilirler.

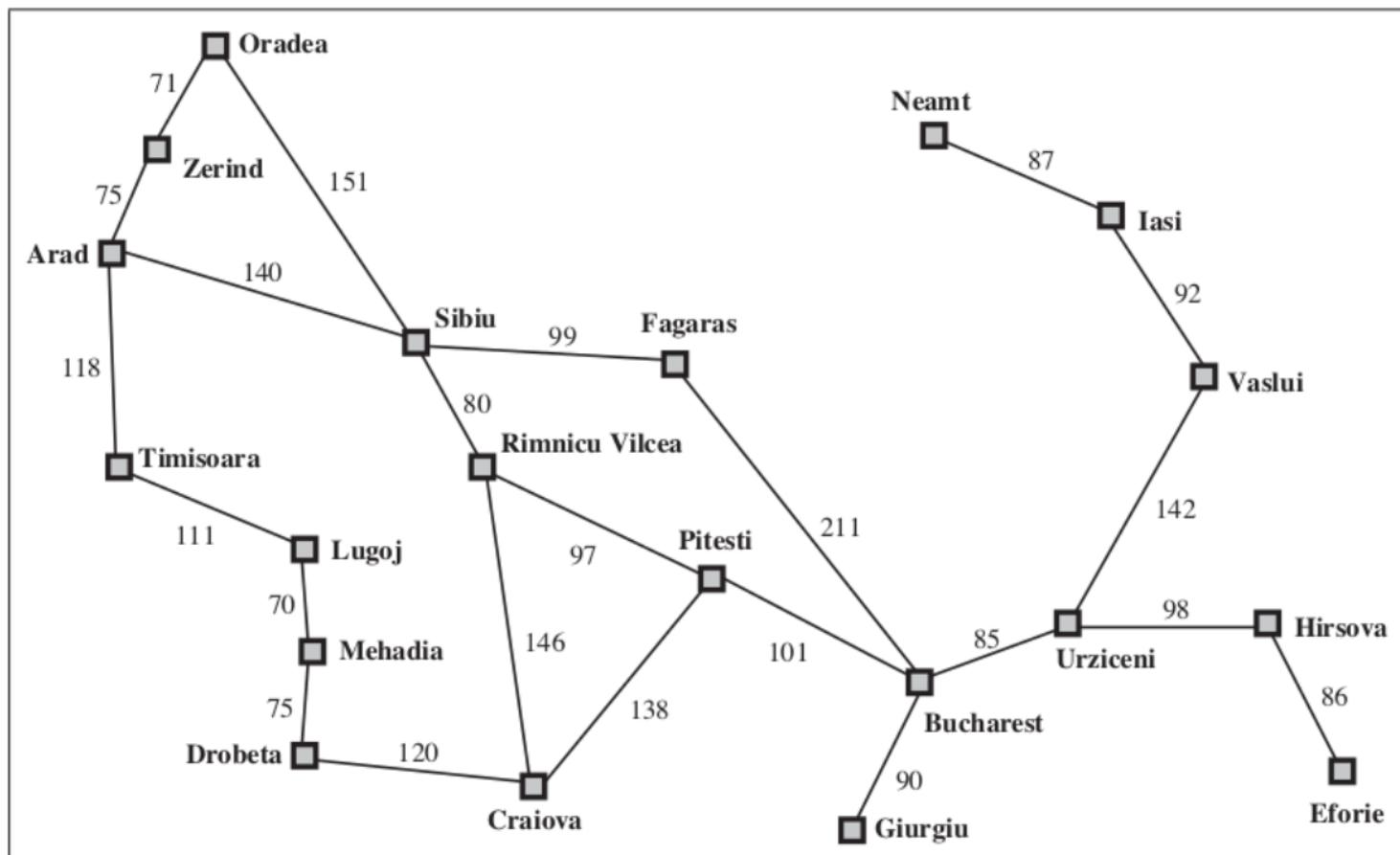
Bilgi ile arama

- Genel olarak en-iyi-öncelikli arama yaklaşımını kullanacağız.
 - ~ Bu yaklaşım daha önce gördüğümüz AĞAÇ-ARAMA veya GRAF-ARAMA algoritmalarının bir versiyonudur.
 - ~ Açılanak düğümler $f(n)$, değerlendirme fonksiyonu ile seçilir.
 - Değerlendirme fonksiyonu bir maliyet tahmini değeri belirler ve en düşük değerlendirmeye sahip düğüm açılmak üzere seçilir.
 - En-iyi-öncelikli arama algoritmasının graf arama olarak implementasyonu daha önce gördüğümüz sabit-maliyetli arama algoritmasına benzer.
 - ~ Sabit maliyet arama algoritmasının implementasyonundaki g değerinin yerini f alır.

Bilgi ile arama

- f fonksiyonun seçimi arama stratejisini belirler.
- Çoğu en-iyi-öncelikli arama algoritması f fonksiyonun bir parçası olarak bir bulușsal fonksiyon (İng: heuristic function), $h(n)$ içerir.
 - ~ $h(n) \Rightarrow n$ durumundan hedef durumuna en ucuz yolun maliyeti
- Örnek verirsek, Romanya tatili örneğimizde haritadaki her şehirden Bükreş şehrine düz-çizgi uzaklıği kullanabiliriz.
- n hedef durumu ise $h(n) = 0$.

Bilgi ile arama



Bilgi ile arama

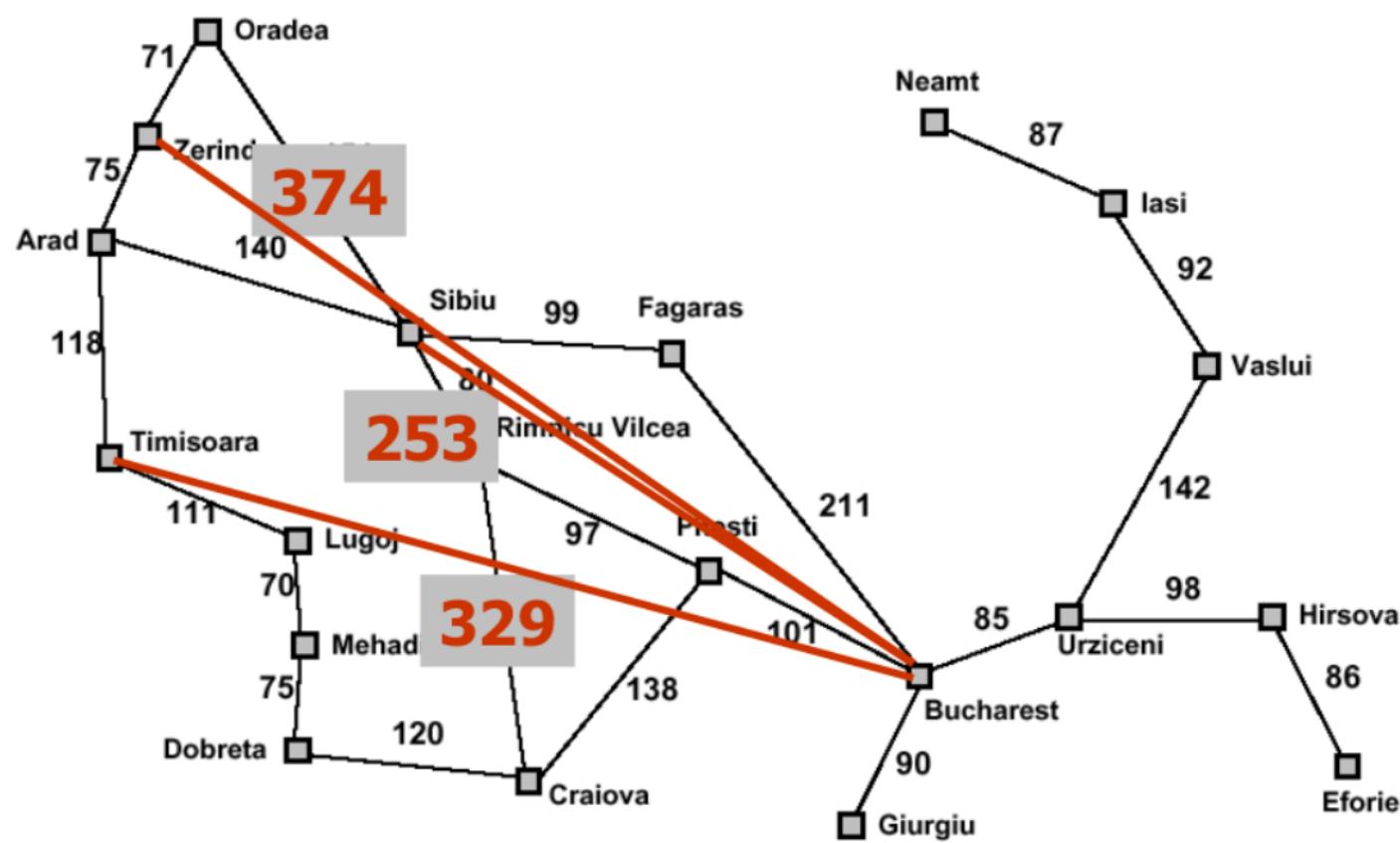
- Açıgözlü en-iyi-öncelikli arama (İng: Greedy best-first search)
 - ~ Açıgözlü en-iyi-öncelikli arama algoritması hedefe en yakın düğümü açar
 - ~ Hedefe en yakın düğümü açarak hızlıca bir çözüme ulaşmayı hedefler
 - ~ $f(n) = h(n)$
 - ~ Romanya tatili örneğini üzerinde algoritmayı çalıştıralım.
 - ~ Buluşsal fonksiyon olarak şehirlerin Bükreş şehrinden düz-çizgi uzaklığını kullanabiliriz.

Bilgi ile arama

- Açıgözlü en-iyi-öncelikli arama

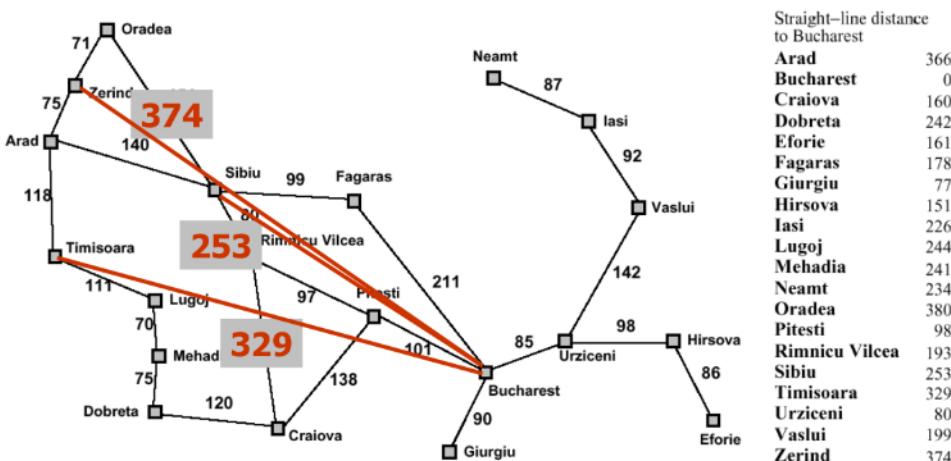
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Bilgi ile arama



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

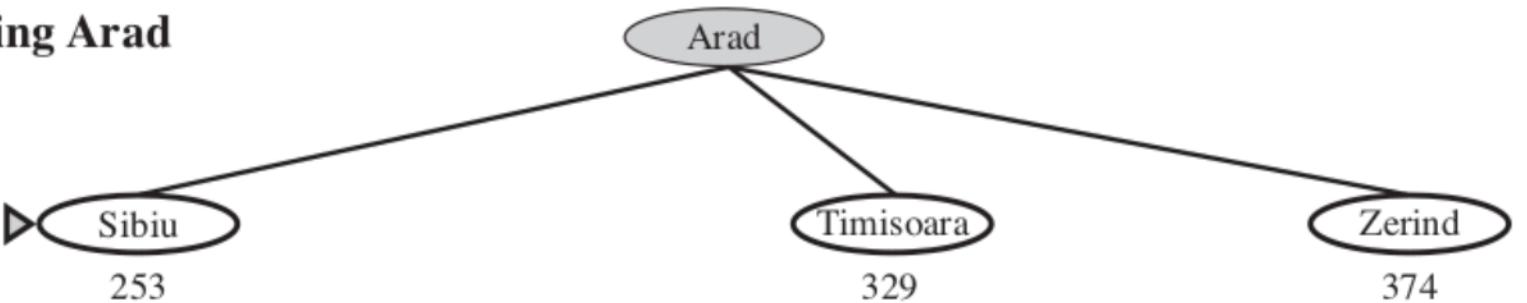
Bilgi ile arama



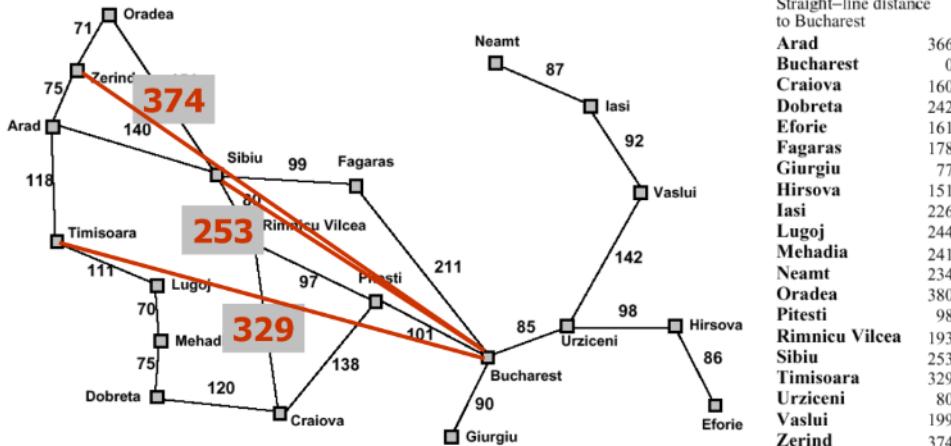
(a) The initial state



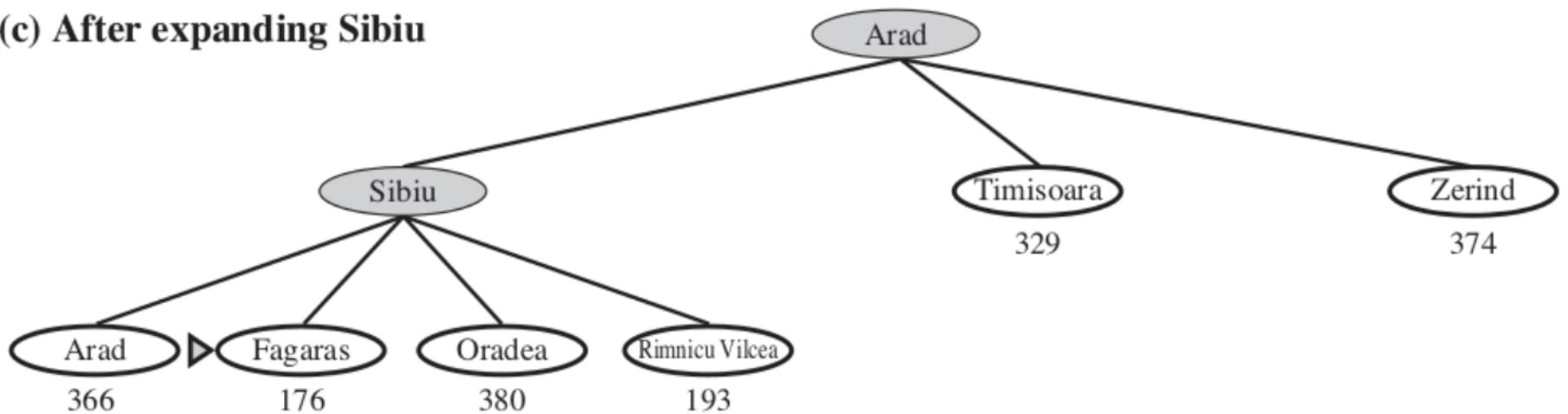
(b) After expanding Arad



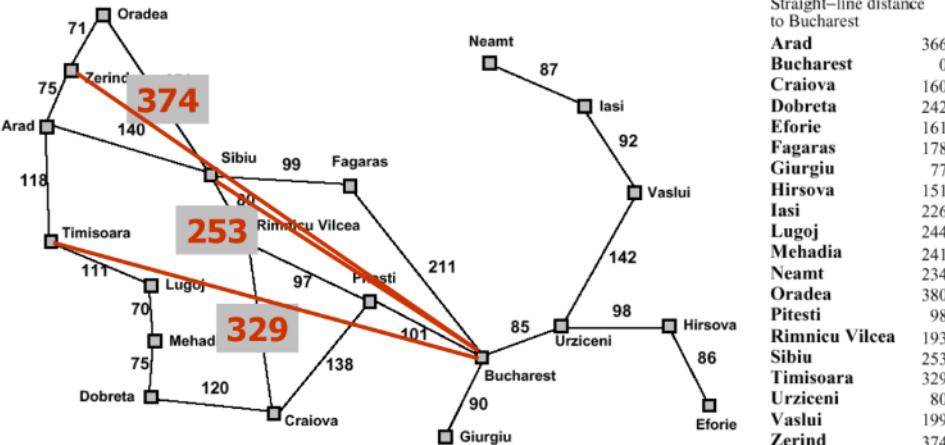
Bilgi ile arama



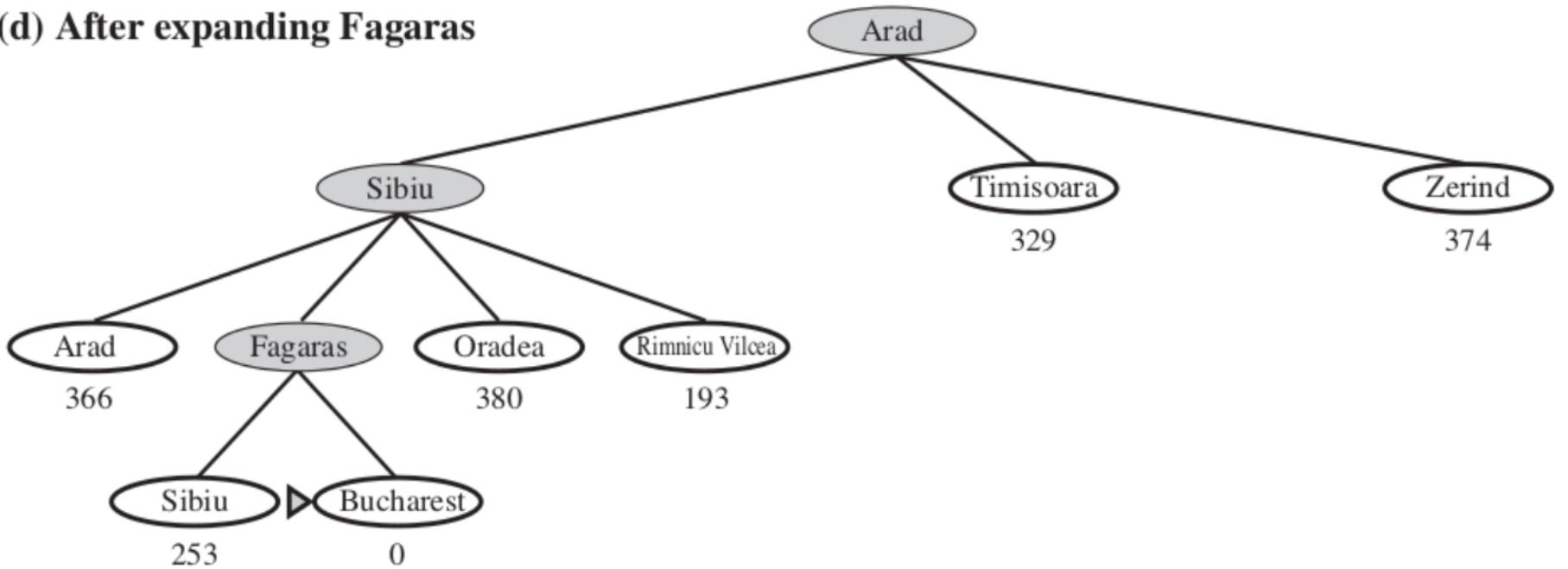
(c) After expanding Sibiu



Bilgi ile arama



(d) After expanding Fagaras



Bilgi ile arama

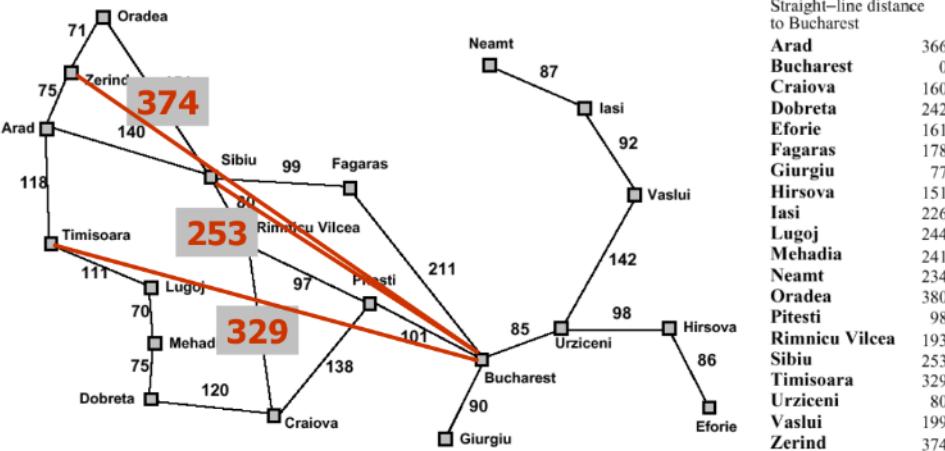
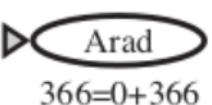
- Açıgözlü en-iyi-öncelikli arama
 - ~ Bu örneğimizde arama operasyonu minimal işlem gerektiriyor
 - açılan her düğüm çözüm yolunun parçası.
 - ~ Ancak bulunan çözüm optimal değil.
 - Aslında bu sebeple “açıgözlü” deniyor.
 - ~ Bulunduğu durumda mümkün olduğunda hedefe yaklaşmaya çalışıyor.
 - ~ Iasi şehrinden Fagaras şehrine gittiğimizi düşünelim
 - Algoritma sonuca ulaşamaz.
 - Graf arama versiyonu sınırlı uzaylarda bütündür, ancak sınırlı olmayan uzaylarda bütün değildir.
 - ~ En kötü durumda zaman ve yer karmaşıkçı arama ağaçları versiyonu için $O(b^m)$, m arama uzayının maksimum derinliği.
 - ~ İyi bir buluşsal fonksiyon ile karmaşıklık azaltılabilir.
 - ~ Azaltma miktarı problem tipine ve buluşsal fonksiyonun kalitesine bağlıdır.

Bilgi ile arama

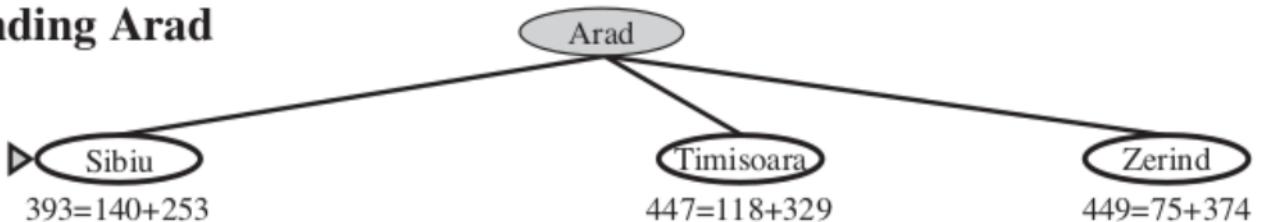
- A* arama
 - ~ A* arama algoritması toplam tahmini çözüm maliyetini minimize etmeyi amaçlar.
 - En bilinen en-iyi-öncelikli arama algoritmasıdır.
 - ~ Düğümleri değerlendirirken $g(n)$, düğüme ulaşma maliyeti ile $h(n)$, düğümden hedefe ulaşma maliyetini birleştirir.
 - $f(n) = g(n) + h(n)$
 - ~ $g(n)$ başlangıç düğümünden düğüm n'e gitminin maliyetini verirken, $h(n)$ düğüm n'den en ucuz yoldak hedefe ulaşmanın tahmini maliyetini verir. Öyleyse
 - $f(n) = n$ düğümünden geçen en ucuz çözümün tahmini maliyeti
 - ~ $h(n)$ belli koşulları sağladığında A* arama hem bütün hem optimaldir.
 - ~ A* arama algoritması, g yerine $g + h$ kullanılması dışında Sabit maliyet arama algoritması ile aynıdır.

Bilgi ile arama

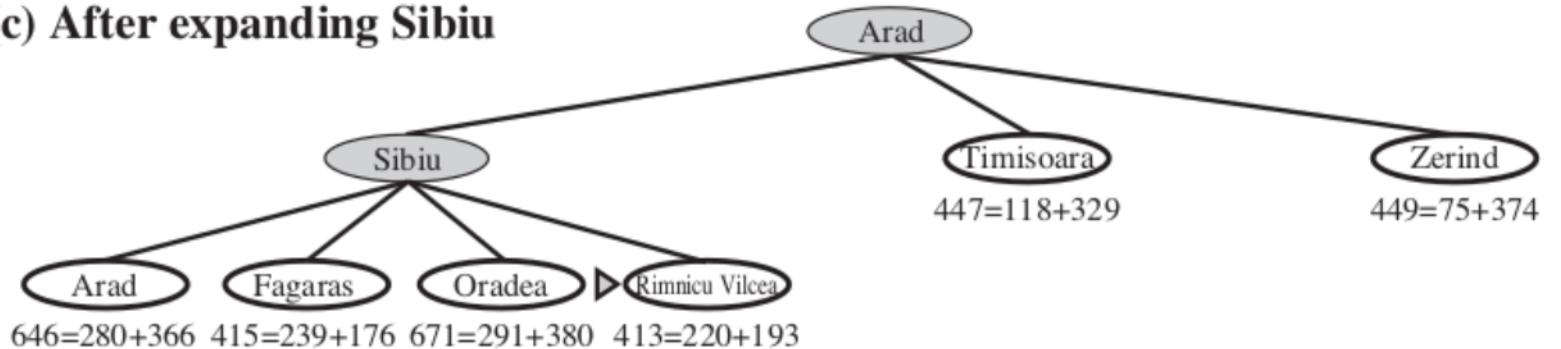
(a) The initial state



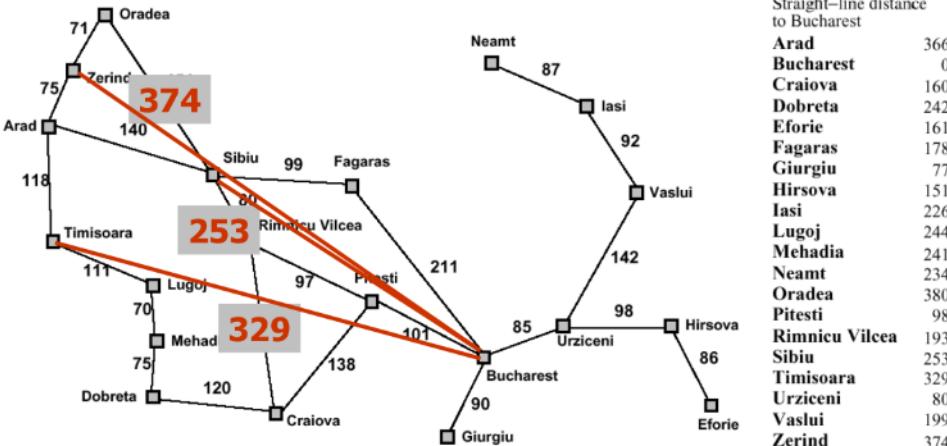
(b) After expanding Arad



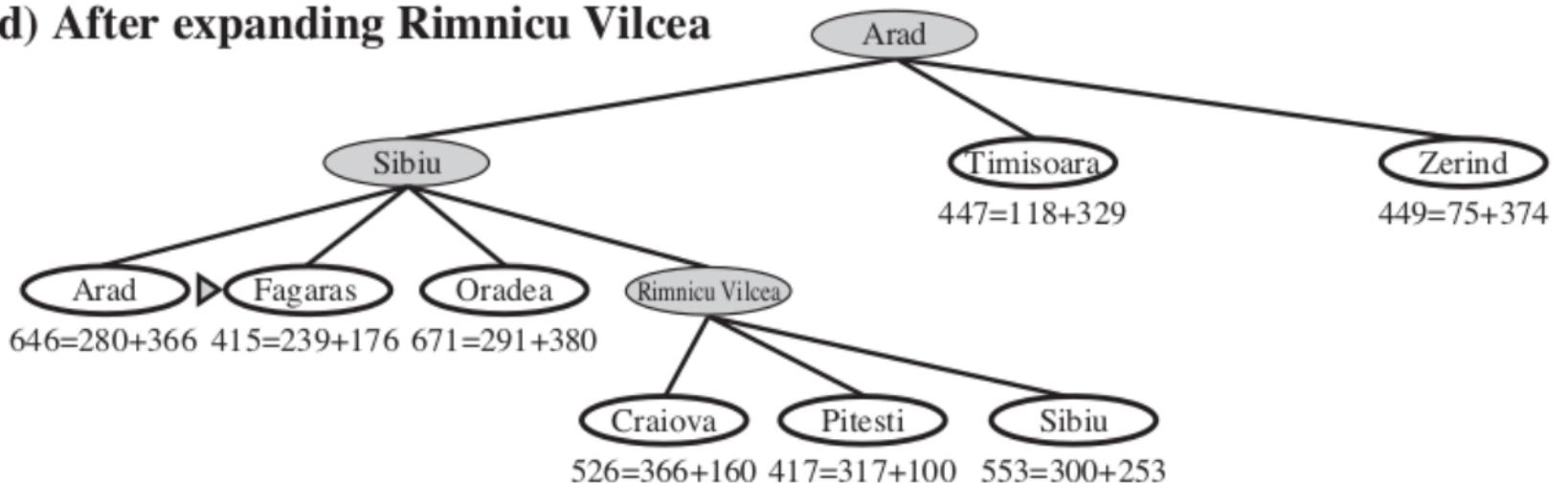
(c) After expanding Sibiu



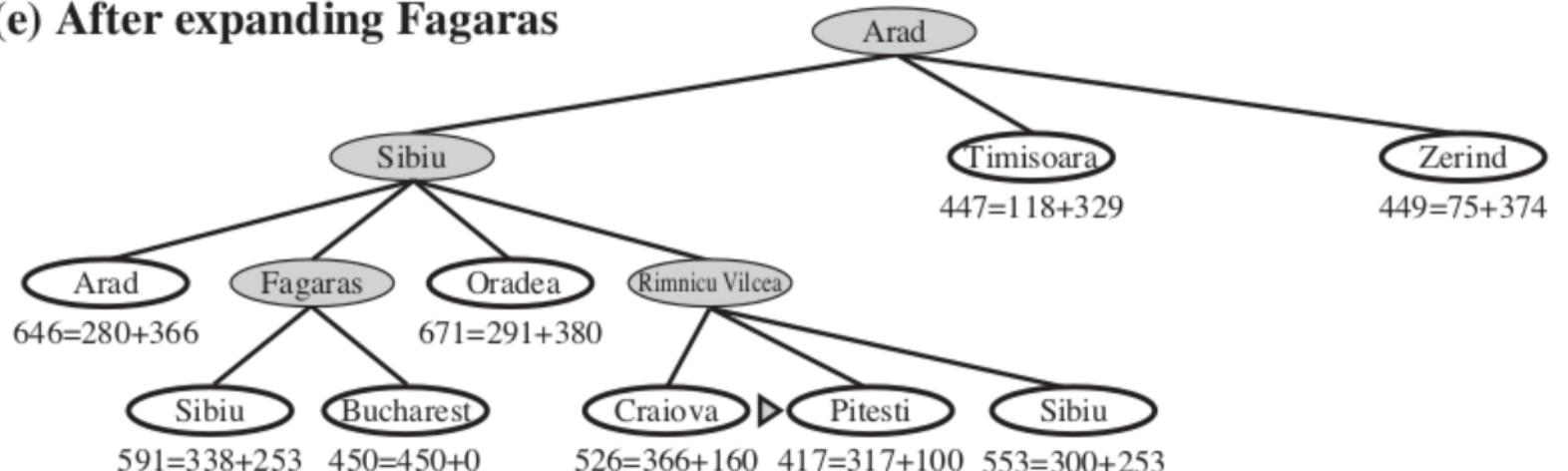
Bilgi ile arama



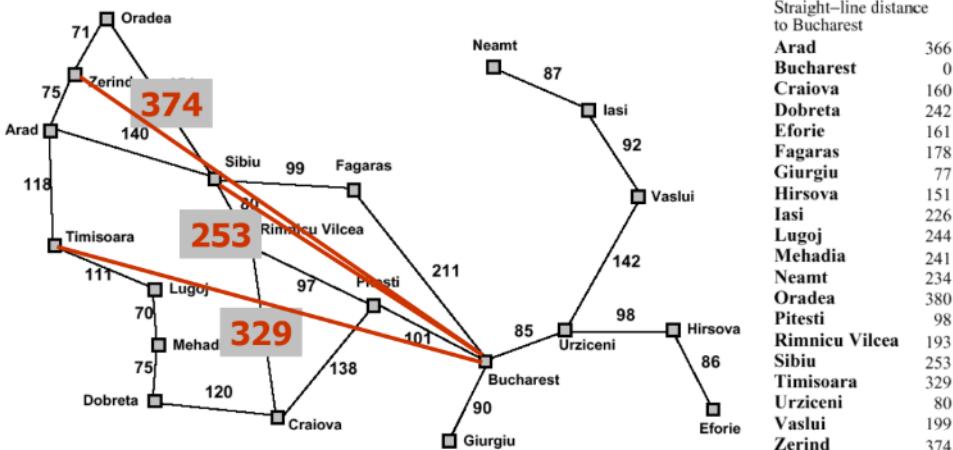
(d) After expanding Rimnicu Vilcea



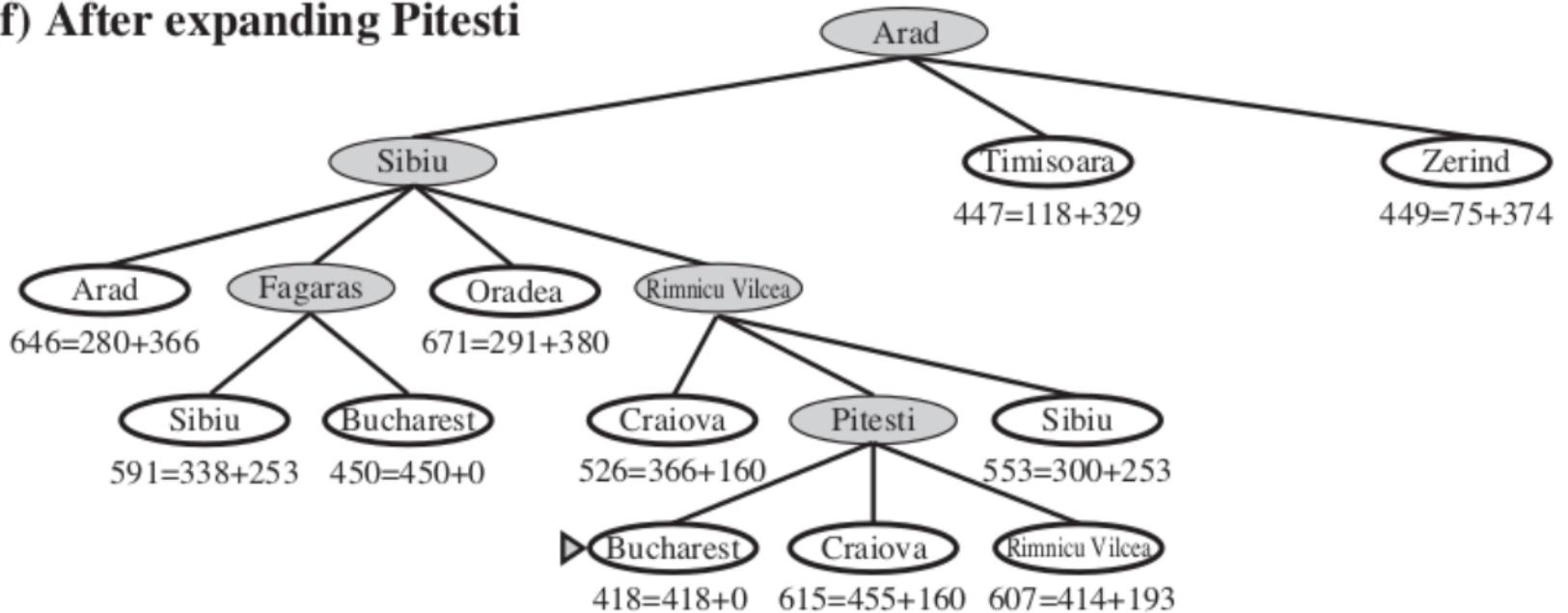
(e) After expanding Fagaras



Bilgi ile arama



(f) After expanding Pitesti



Bilgi ile arama

- A* arama
 - ~ Optimallik için gerekli koşullar: Onanırlık (İng: admissibility), tutarlılık (İng: consistency).
 - Algoritmanın optimal olabilmesi için $h(n)$ onanır olmalıdır.
 - ~ Onanır bir buluşsal fonksiyon asla hedefe ulaşma maliyetini olduğundan fazla tahmin etmez.
 - ~ $g(n)$ n düğümüne ulaşma maliyeti ise ve $h(n)$ n'den hedefe ulaşma maliyetini fazla tahmin etmezse, $f(n)$ de bir çözümün maliyetini asla fazla tahmin etmez.
 - ~ Örneğimiz için kullandığımız doğru çizgi uzaklığı onanır buluşsal fonksiyondur.
 - Algoritmanın optimal olabilmesi için $h(n)$ tutarlı olmalıdır.
 - ~ $h(n)$ fonksiyonunun tutarlı olması için aşağıdaki denklemin geçerli olması gerekmektedir.
 - $h(n) \leq c(n,a,n') + h(n')$
 - Bu eşitsizliğe üçgen eşitsizlik denir.
 - Her tutarlı buluşsal fonksiyon aynı zamanda onanırdır.

Bilgi ile arama

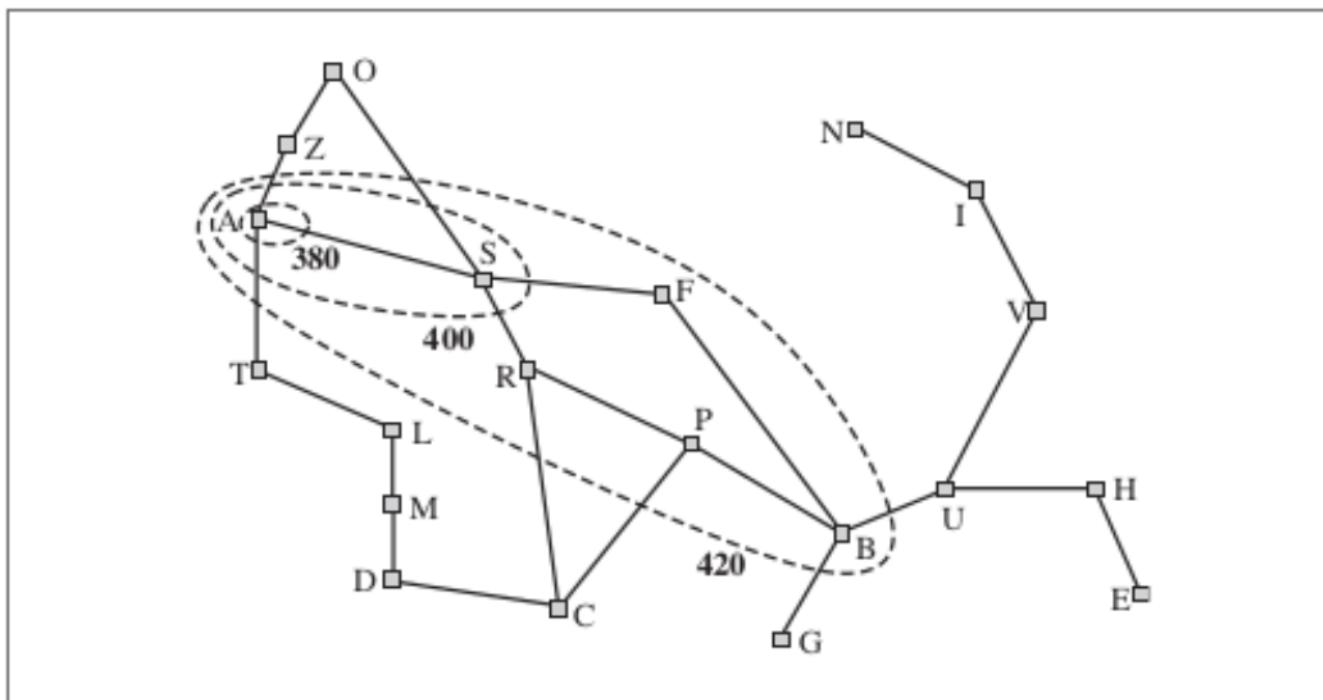
- A* algoritmasının optimalliği
 - ~ Daha önce belirttiğimiz üzere
 - A* algoritmasının ağaç arama versionu $h(n)$ onanır ise optimaldir.
 - A* algoritmasının graf arama versiyonu $h(n)$ tutarlı ise optimaldir.
 - ~ İkinci durumu ispatlayalım
 - Eğer $h(n)$ tutarlı ise, $f(n)$ değerleri bir yol boyunca azalmayandır.
 - ~ n' n düğümünün ardılı olsun, öyleyse bir a aksiyonu için $g(n') = g(n) + c(n,a,n')$ olmalı. Öyleyse
 - ~ $f(n') = g(n') + h(n') = g(n) + c(n,a,n') + h(n') \geq g(n) + h(n) = f(n)$

Bilgi ile arama

- A* algoritmasının optimalliği
 - ~ İkinci durumun ispatı (devam)
 - A* bir düğümü açmak için seçtiğinde, o düğüme gelen optimal yol bulunmuştur.
 - Durum böyle olmasaydı, n düğümüne gelen optimal yol üzerinde sınırla başka bir n' düğümü olurdu.
 - Ancak f fonksiyonu az önce gösterdiğimiz üzere azalmayandır, böyle bir n' düğümü daha az bir f-maliyetine sahip olurdu ve algoritma tarafından daha önce seçilirdi.
 - ~ Bu iki gözlemi birleştirirsek, söyleyebiliriz ki A* tarafından açılan graf araması ile açılan düğümler $f(n)$ tarafından azalmayan sırayla açılır.
 - ~ Öyleyse açılan ilk hedef düğüm optimal çözüm olmalıdır. Çünkü f hedef düğümlerin doğru maliyetini veriyorsa (hedef olduğu için $h = 0$), sonra açılan hedef düğümlerin her biri en az bu düğüm kadar maliyetli olmalıdır.

Bilgi ile arama

- A* algoritması
 - ~ Belirttiğimiz üzere f üzerinden hesaplanan maliyetlere herhangi bir yol üzerinde azalmayandır. Öyleyse durum uzayında konturlar (eş uzaklık çizgileri) çizebiliriz.



Bilgi ile arama

- A* algoritması
 - ~ Sabit maliyetli arama kullanıldığında bu konturlar başlangıç durumunun etrafında daireler şeklindedir.
 - ~ İsabetli buluşsal fonksiyonlar kullanıldığında ise kontur çizgileri hedef durumuna doğru yönelir ve çizgiler arası, optimal yol etrafında azalır.
 - ~ Eğer C* optimal çözüm yolunun maliyeti ise
 - A* $f(n) < C^*$ olan bütün düğümleri açar.
 - Bunu yaptıktan sonra A*, hedef düğümünü seçmeden önce, "hedef konturu"ndaki düğümlerin bir kısmını açabilir.
 - ~ Bütün olması için C^* aşağı maliyete sahip sınırlı sayıda düğüm olmalıdır.
 - Bu durum b sınırlı olduğunda ve her bir adımın maliyetinin en az ϵ olduğunda geçerlidir.

Bilgi ile arama

- A* algoritması
 - ~ A* $f(n) > C^*$ olan düğümleri açmaz.
 - Örneğimizde Timisoara şehrini içeren düğüm açılmamıştır.
 - Bu sebeple Timisoara altındaki alt ağaç budanmıştır.
 - Kullandığımız buluşsal fonksiyon onanır olduğu için bu alt ağaç görmezden gelebiliriz.
 - Budama konsepti, yani bazı olasılıkları denemeden eleme işlemi, birçok yapay zeka probleminin çözümünde kullanılan bir yöntemdir.

Bilgi ile arama

- A* algoritması
 - ~ Zaman karmaşıklığı
 - h hesaplamasındaki relatif hata ve çözümün uzunluğuna bağlı olarak üstel (exponential) olarak hesaplanır.
 - ~ Yer karmaşıklığı
 - Oluşturulan bütün düğümler hafızada tutulmalıdır.
 - ~ Çalışma zamanı üstel olarak hesaplanmakla birlikte, bu durum A* algoritmasının en büyük sorunu değildir.
 - ~ Bütün oluşturulan düğümler hafızada tutulduğu için A* algoritması büyük ölçekli problemlerde genellikle çok yüksek miktarda hafıza gerektirir.

Bilgi ile arama

- Hafıza sınırlı bulușsal arama
 - ~ A* algoritmasının hafıza ihtiyacını azaltmak için kullanabilecek yöntemlerden biri yinelemeli derinleşen aramaya benzer bir yöntem kullanmaktadır.
 - YDA yerine YDA* algoritması
 - ~ Standart yinelemeli derinleşen arama ile YDA* arasındaki ana fark, YDA algoritmasının kullanılan derinlik değeri yerine, YDA* algoritması kesme yaparken f-maliyeti kullanır.
 - ~ Her yinelemede, yeni kesme değeri önceki kesme değerini geçen en ufak f-maliyetine sahip düğümün değeridir.
 - ~ Ancak bu yöntem de yinelemeli derinleşen aramadaki sorunları içermektedir.
 - ~ Yinelemeli en iyi öncelikli arama algoritması yinelemeli bir algoritmadır ve doğrusal artan zamanda en iyi öncelikli arama yapmaya çalışır.
 - Yinelemeli derinlik öncelikli aramaya benzer, ancak belli bir yol üzerinde aramaya devam etmek yerine bulunduğu düğümün üst düğümlerinden en iyi f-maliyete sahip düğümü hatırlar ve arama belli bir limiti aştığında geri dönerek alternatif yolları dener.

Bilgi ile arama

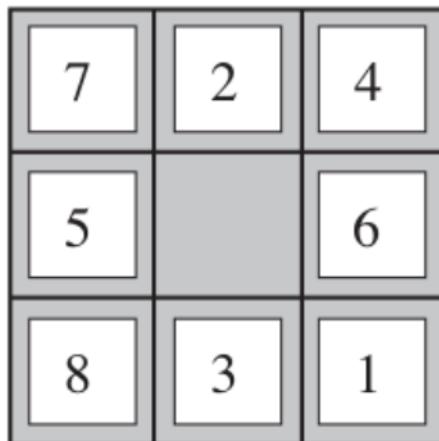
- Buluşsal fonksiyonlar
 - ~ 8'li bulmaca
 - Çözüm için gereken ortalama adım sayısı 22.
 - Ortalama dallanma faktörü 3.
 - Demek ki toplamda $3^{22} \approx 3.1 \times 10^{10}$ durumu incelemeliyiz.
 - Aslında sadece $9! / 2 = 181,440$ durum birbirinden farklı.
 - ~ Bu kadar durumu teker teker test etmek mümkün.
 - Ancak 15'li bulmacayı düşünürsek bu sayı yaklaşık 10^{13} .
 - ~ Yani iyi çalışan bir buluşsal fonksiyon bulmalıyız.

Bilgi ile arama

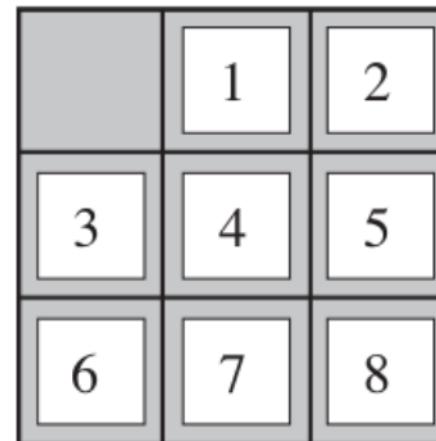
- Buluşsal fonksiyonlar (Ing: Heuristic functions)
 - ~ Sonuca daha hızlı ulaşabilmemizde buluşsal fonksiyonların etkisi fazladır.
 - ~ 8'li bulmacayı düşünelim
 - A* algoritması ile en kısa yoldan çözüme ulaşmak istiyorsak, gereken adım sayısını olduğundan fazla tahmin etmeyen bir bilişsel fonksiyona ihtiyacımız var.
 - ~ Bu fonksiyon onanır olmalı.
 - İki tane çoklukla kullanılan buluşsal fonksiyon mevcut
 - ~ h_1 = doğru yerine yerleşmemiş döşeme sayısı
 - Bu onanır bir buluşsal fonksiyondur, çünkü yerinde bulunmayan döşemeler en az bir kere hareket etmelidir.
 - ~ h_2 = yerinde olmayan döşemelerin yerlerine uzaklıkları toplamı.
 - Döşemeler diagonal olarak hareket edemeyeceği için dikey ve yatay uzaklıklar toplanır.
 - Bu uzaklığı blok uzaklığı veya manhattan uzaklığı adı verilir.
 - Bu buluşsal fonksiyon da onanırdır, çünkü her döşeme doğru yerine geçebilmek için en az hesaplanan uzaklık kadar hareket etmelidir.

Bilgi ile arama

- Buluşsal fonksiyonlar
 - ~ 8'li bulmaca
 - $h1(n)$ = yerinde olmayan döşeme sayısı
 - $h2(n)$ = toplam Manhattan uzaklığı



Start State



Goal State

- $h1(n) = ?$
- $h2(n) = ?$

Bilgi ile arama

- Buluşsal fonksiyonlar
 - ~ Bir buluşsal fonksiyonun kalitesini ölçmek için kullanılan metriklerden biri efektif dallanma faktöründür: b^*
 - ~ A* algoritması tarafından belli bir problem çözülürken N tane düğüm oluşturuluyor ve çözüm d derinliğinde ise b^* şu şekilde hesaplanır:
 - $N + 1 = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$
 - ~ Örneğin A* 52 düğüm oluştururak bir problemi çözüyor ve çözüm 5 derinliğinde ise efektif dallanma faktörü 1.92 olarak hesaplanır.
 - ~ İstediğimiz b^* değerinin 1'e yakın olmasıdır.
 - ~ h_1 ve h_2 buluşsal fonksiyonlarını karşılaştırmak için 1200 tane rastgele oluşturulmuş, 2 ile 24 arası uzunlukta çözümü olan problem farklı algoritmalarla çözülmüş.

Bilgi ile arama

d	Search Cost (nodes generated)			Effective Branching Factor		
	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Bilgi ile arama

- Buluşsal fonksiyonlar
 - ~ Peki h_2 her zaman h_1 'den iyi sonuç mu veriyor?
 - Evet!
 - ~ Bunun nedeni, herhangi bir n düğümü için $h_2(n) \geq h_1(n)$
 - Öyleyse h_2 , h_1 domine eder diyoruz.
 - ~ Domine etme durumu şu şekilde sonuç verir: h_2 kullanan A* algoritması h_1 kullanan algoritmasından, bir problemi çözerken daha fazla düğüm üretmez

Bilgi ile arama

- Gevsetilmiş problem
 - ~ Onanır bulusal fonksiyonlar asıl problemin gevsetilmiş versiyonuna bulunan tam çözüm ile oluşturulabilir.
 - ~ 8'li bulmaca için düşünürsek:
 - Bir döşeme istenilen her pozisyon'a hareket ettirilebilir dersek
~ $h_1(n)$ en kısa çözümü verir.
 - Bir döşeme komşu olan her pozisyon'a hareket ettirilebilir dersek
~ $h_2(n)$ en kısa çözümü verir.
 - ~ Bir problemin daha az kısıtlama ve koşulla tanımlanmış haline gevsetilmiş problem denir.

Bilgi ile arama

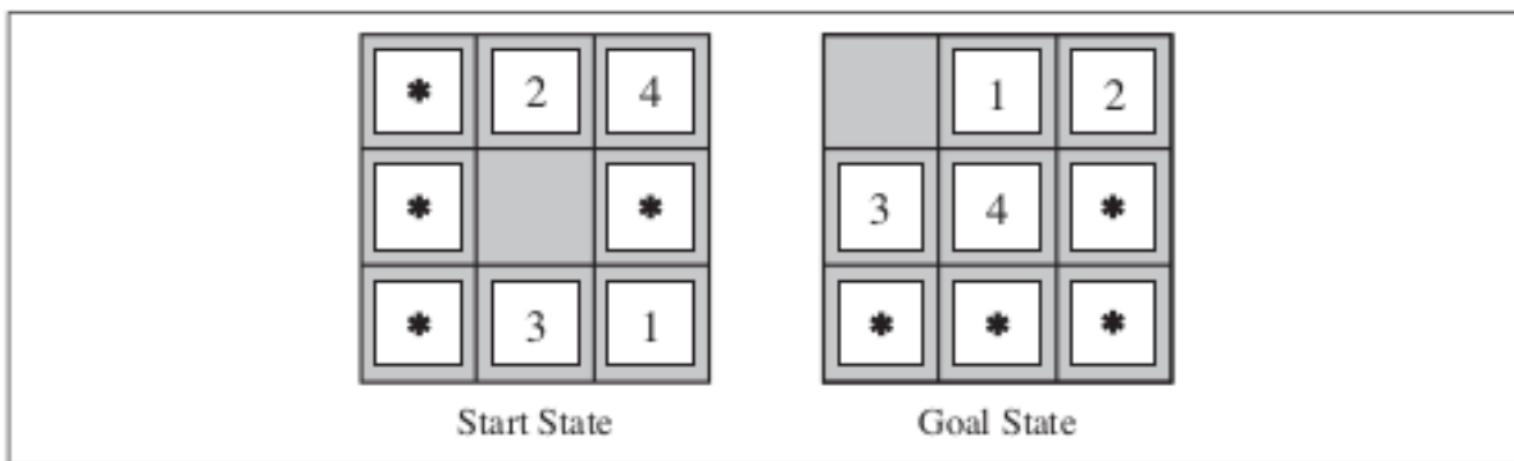
- Gevsetilmiş problem
 - ~ Gevsetilmiş problem oluşturulurken durum uzayına yeni kenarlar eklenir
 - ~ Öyleyse asıl problemin optimal çözümü aynı zamanda gevsetilmiş problemi çözer.
 - ~ Ancak gevsetilmiş problem için daha kısa çözümler olabilir çünkü eklenen kenarlar sayesinde kısa yollar oluşabilir.
 - ~ Öyleyse gevsetilmiş problem için bulunan optimal çözüm, asıl problem için onanır bulusal fonksiyondur.
 - ~ Oluşturulan bulusal fonksiyon gevsetilmiş problem için tam maliyeti hesapladığına göre, üçgen eşitsizliğine uymalıdır.
 - Öyleyse tutarlıdır.

Bilgi ile arama

- Gevsetilmiş problem
 - ~ Eğer bir problem resmi olarak tanımlanırsa, gevsetilmiş versiyonu oluşturulabilir.
 - ~ Örneğin, 8'li bulmacadaki aksiyonlar şu şekilde tanımlanmışsa
 - Bir döşemenin A pozisyonundan B pozisyonuna hareket ettirilebilmesi için:
 - ~ A yatay ve dikey olarak B'ye komşu olmalı ve B boş olmalıdır.
 - Bu koşulların birini veya ikisini birden kaldırarak üç farklı gevsetilmiş problem oluşturabiliriz.
 - ~ A B'ye komşu olmalıdır.
 - ~ B boş olmalıdır
 - ~ Koşul yoktur. Her pozisyondan diğerine hareket edilebilir.
 - Bu gevsetilmiş problemlerden elde edilen buluşsal fonksiyonları düşünürsek ikinci kural ortadan kaldırıldığında h_2 , kuralların ikisi de kaldırıldığında ise h_1 olarak belirlenir.

Bilgi ile arama

- Buluşsal fonksiyonlar
 - ~ Birçok farklı buluşsal fonksiyon ürettiğimizde en iyisini seçmeliyiz
 - ~ $h_1, h_2, h_3, \dots, h_m$ şeklinde fonksiyonlarımız var ise
 - $h(n) = \max\{h_1(n), h_2(n), h_3(n), \dots, h_m(n)\}$
 - ~ Onanır buluşsal fonksiyonlar altproblemlere bulunan çözümlerden elde edilebilir.



Bilgi ile arama

- Buluşsal fonksiyonlar
 - ~ Her farklı altprobleme ait çözümler üreterek örüntü veritabanı (İng: pattern database) üretebiliriz.
 - ~ Bundan sonra bu saklanmış bilgiyi kullanarak buluşsal fonksiyon oluşturabiliriz.
 - ~ Farklı alt problemlere ait bilgileri saklayarak, oluşturulan düğüm sayısını oldukça fazla azaltabiliriz.

