

Metoda Searching

Heuristic Search

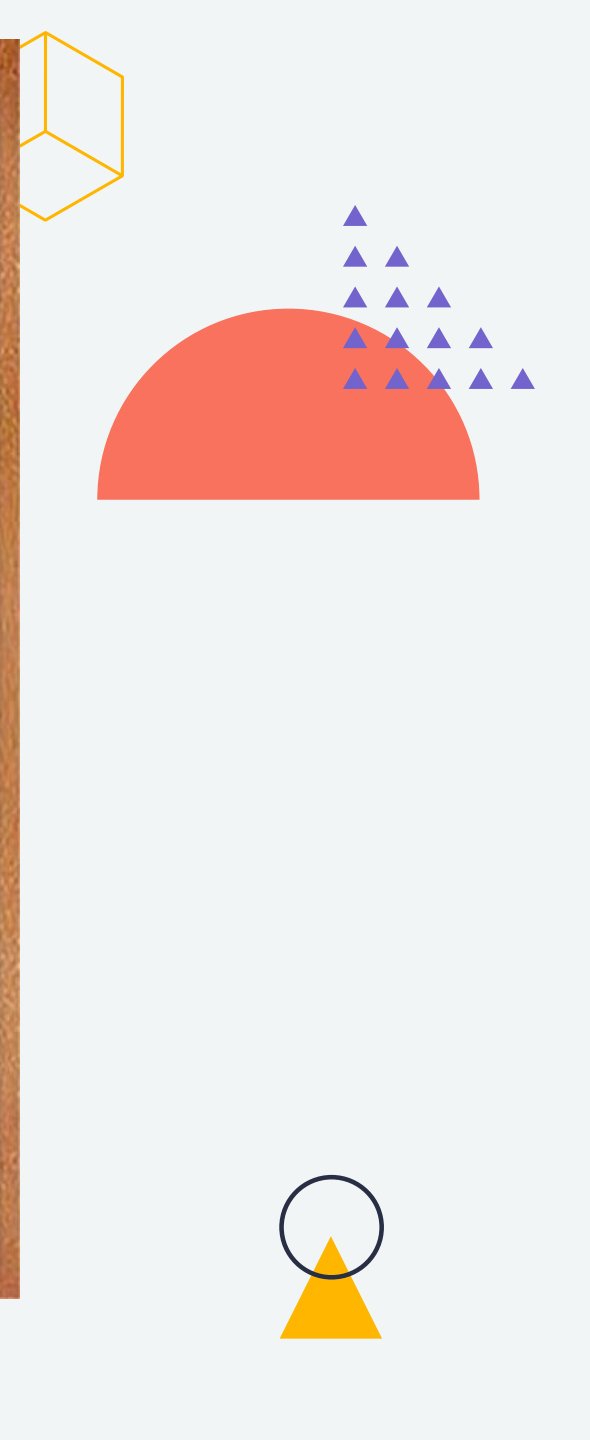
Dosen

RAHMIATI





Pencarian Heuristik

- Ada 4 metode pencarian heuristik
 - Pembangkit & Pengujian (*Generate and Test*)
 - Pendakian Bukit (*Hill Climbing*)
 - Pencarian Terbaik Pertama (*Best First Search*)
 - *Simulated Annealing*
- 

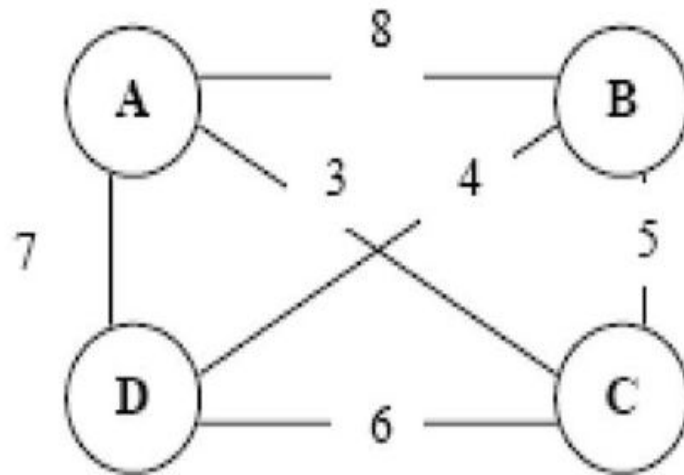
Pembangkit & Pengujian (*Generate and Test*)

- Pada prinsipnya metode ini merupakan penggabungan antara depth-first search dengan pelacakan mundur (backtracking), yaitu bergerak ke belakang menuju pada suatu keadaan awal.
- Algoritma:
 - Bangkitkan suatu kemungkinan solusi (membangkitkan suatu titik tertentu atau lintasan tertentu dari keadaan awal).
 - Uji untuk melihat apakah node tersebut benar-benar merupakan solusinya dengan cara membandingkan node tersebut atau node akhir dari suatu lintasan yang dipilih dengan kumpulan tujuan yang diharapkan.
 - Jika solusi ditemukan, keluar. Jika tidak, ulangi kembali langkah yang pertama.

Contoh

Traveling Salesman Problem (TSP)

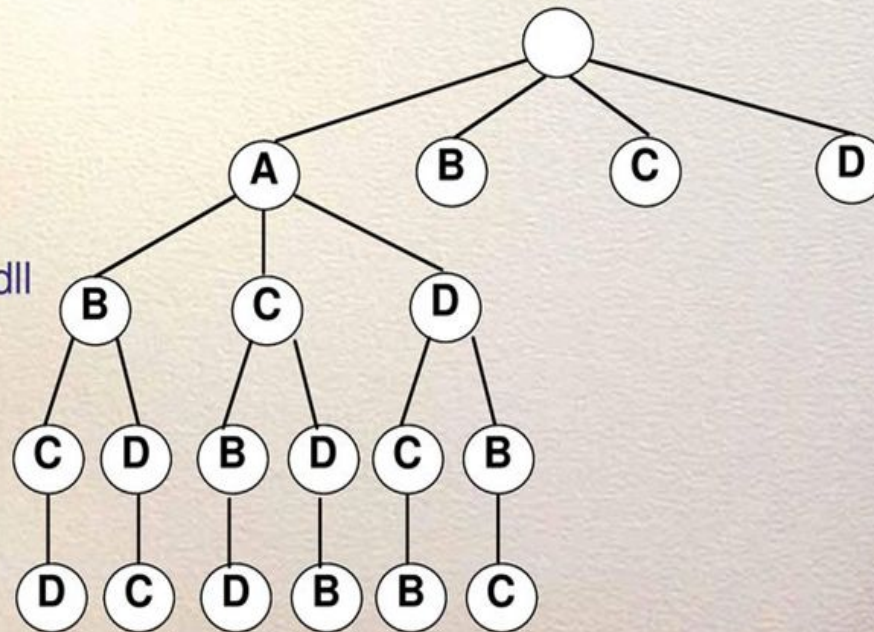
- Seorang salesman ingin mengunjungi n kota. Jarak antara tiap-tiap kota sudah diketahui. Ingin diketahui rute terpendek dimana setiap kota hanya boleh dikunjungi tepat 1 kali.



Contoh Traveling Salesman Problem (TSP)

- Generate & test akan membangkitkan semua solusi yang mungkin:

- A – B – C – D
- A – B – D – C
- A – C – B – D
- A – C – D – B, dll



Lintasan

Pencarian ke-	Lintasan	Panjang Lintasan	Lintasan terpilih	Panjang Lintasan terpilih
1.	ABCD	19	ABCD	19
2.	ABDC	18	ABDC	18
3.	ACBD	12	ACBD	12
4.	ACDB	13	ACBD	12
5.	ADBC	16	ACBD	12
6.	ADCB	18	ACBD	12
7.	BACD	17	ACBD	12
8.	BADC	21	ACBD	12
9.	BCAD	15	ACBD	12
10.	BCDA	18	ACBD	12
11.	BDAC	14	ACBD	12
12.	BDCA	13	ACBD	12



Pencarian ke-	Lintasan	Panjang Lintasan	Lintasan terpilih	Panjang Lintasan terpilih
13.	CABD	15	ACBD	12
14.	CADB	14	ACBD	12
15.	CBAD	20	ACBD	12
16.	CBDA	16	ACBD	12
17.	CDAB	21	ACBD	12
18.	CDBA	18	ACBD	12
19.	DABC	20	ACBD	12
20.	DACD	15	ACBD	12
21.	DBAC	15	ACBD	12
22.	DBCA	12	ACBD atau DBCA	12
23.	DCAB	17	ACBD atau DBCA	12
24.	DCBA	19	ACBD atau DBCA	12

Pembangkit & Pengujian (Generate and Test)

- Kelemahan
 - Perlu membangkitkan semua kemungkinan sebelum dilakukan pengujian
 - Membutuhkan waktu yang cukup lama dalam pencariannya

Pendakian Bukit (Hill Climbing)

- Metode ini hampir sama dengan metode pembangkitan & pengujian, hanya saja proses pengujian dilakukan dengan menggunakan fungsi heuristik.
- Pembangkitan keadaan berikutnya sangat tergantung pada feedback dari prosedur pengetesan.
- Tes yang berupa fungsi heuristic ini akan menunjukkan seberapa baiknya nilai terkaan yang diambil terhadap keadaan-keadaan lainnya yang mungkin.

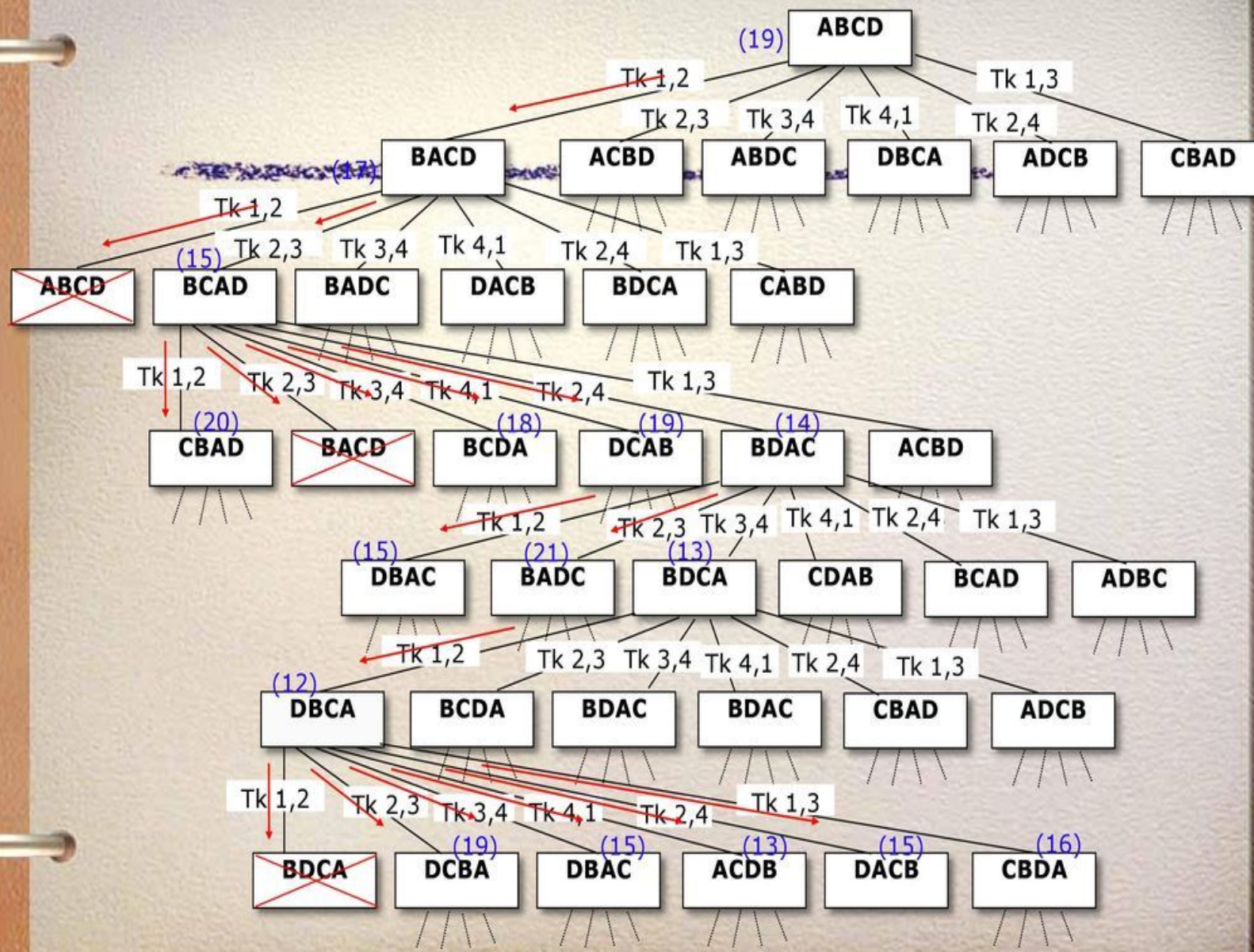
Simple Hill Climbing

- Algoritma
 - Mulai dari keadaan awal, lakukan pengujian: jika merupakan tujuan, maka berhenti; dan jika tidak, lanjutkan dengan keadaan sekarang sebagai keadaan awal.
 - Kerjakan langkah-langkah berikut sampai solusinya ditemukan, atau sampai tidak ada operator baru yang akan diaplikasikan pada keadaan sekarang:
 - Cari operator yang belum pernah digunakan; gunakan operator ini untuk mendapatkan keadaan yang baru.
 - Evaluasi keadaan baru tersebut.
 - Jika keadaan baru merupakan tujuan, keluar.
 - Jika bukan tujuan, namun nilainya lebih baik daripada keadaan sekarang, maka jadikan keadaan baru tersebut menjadi keadaan sekarang.
 - Jika keadaan baru tidak lebih baik daripada keadaan sekarang, maka lanjutkan iterasi.

Contoh TSP

- Operator : Tukar kota ke-i dengan kota ke-j (Tk i,j)
- Untuk 4 kota:
 - Tk 1,2 : tukar kota ke-1 dengan kota ke-2.
 - Tk 1,3 : tukar kota ke-1 dengan kota ke-3.
 - Tk 1,4 : tukar kota ke-1 dengan kota ke-4.
 - Tk 2,3 : tukar kota ke-2 dengan kota ke-3.
 - Tk 2,4 : tukar kota ke-2 dengan kota ke-4.
 - Tk 3,4 : tukar kota ke-3 dengan kota ke-4.
- Untuk N kota, akan ada operator sebanyak:

$$\frac{N!}{2!(N-2)!}$$

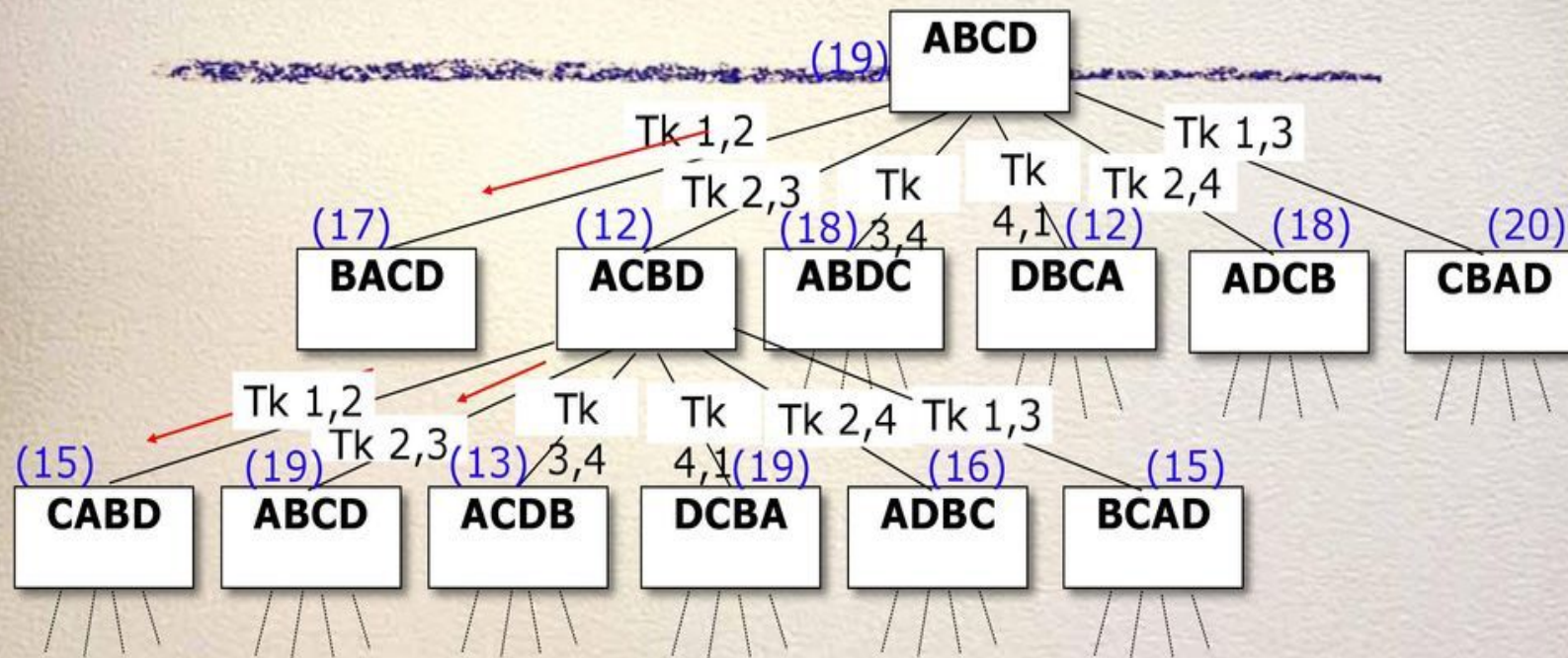


Steepest Ascent Hill Climbing

- Steepest-ascent hill climbing sebenarnya hampir sama dengan simple hill climbing, hanya saja gerakan pencarian tidak dimulai dari posisi paling kiri.
- Gerakan selanjutnya dicari berdasarkan nilai heuristik terbaik.
- Dalam hal ini urutan penggunaan operator tidak menentukan penemuan solusi.

Algoritma

- Mulai dari keadaan awal, lakukan pengujian: jika merupakan tujuan, maka berhenti; dan jika tidak, lanjutkan dengan keadaan sekarang sebagai keadaan awal.
- Kerjakan hingga tujuan tercapai atau hingga iterasi tidak memberikan perubahan pada keadaan sekarang.
- Tentukan SUCC sebagai nilai heuristic terbaik dari successor-successor.
- Kerjakan untuk tiap operator yang digunakan oleh keadaan sekarang:
- Gunakan operator tersebut dan bentuk keadaan baru.
- Evaluasi keadaan baru tersebut. Jika merupakan tujuan, keluar. Jika bukan, bandingkan nilai heuristicnya dengan SUCC. Jika lebih baik, jadikan nilai heuristic keadaan baru tersebut sebagai SUCC. Namun jika tidak lebih baik, nilai SUCC tidak berubah.
- Jika SUCC lebih baik daripada nilai heuristic keadaan sekarang, ubah node SUCC menjadi keadaan sekarang.





SEKIAN DAN TERIMA KASIH

