METODE PENCARIAN HEURISTIK

Pencarian Heuristik

- Merupakan teknik yang digunakan untuk meningkatkan efisiensi dari proses pencarian

- Dalam pencarian *state space,* heuristik adalah aturan untuk memilih cabang-cabang yang paling mungkin menyebabkan penyelesaian permasalahan dapat diterima

Metode Pencarian Heuristik

1. *Generate and Test* (Pembangkit dan Pengujian)
2. *Hill Climbing* (Pendakian Bukit)
3. *Best First Search* (Pencarian Terbaik Pertama)
4. *Simulated Annealing*

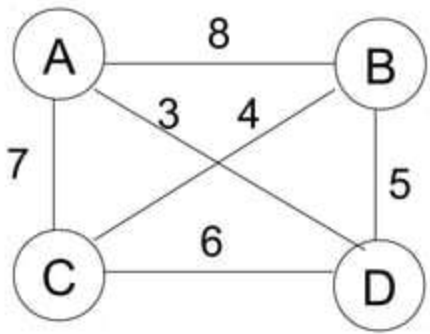
Generate and Test (Pembangkit dan Pengujian)

- Pengabungan antara depth first search dengan pelacakan mundur (backtracking)
- Nilai Pengujian berupa jawaban ‘ya’ atau ‘tidak’
- Jika pembangkit possible solution dikerjakan secara sistimatis, maka prosedur akan mencari solusinya, jika ada.

- Algoritma:
  - Bangkitkan suatu kemungkinan solusi
  - Uji apakah node tersebut merupakan solusi, dengan cara membandingkan node atau node akhir suatu lintasan yang dipilih dengan kumpulan tujuan yang diharapkan
  - Jika solusi ditemukan, keluar. Jika tidak, ulangi langkah pertama.nya dengan

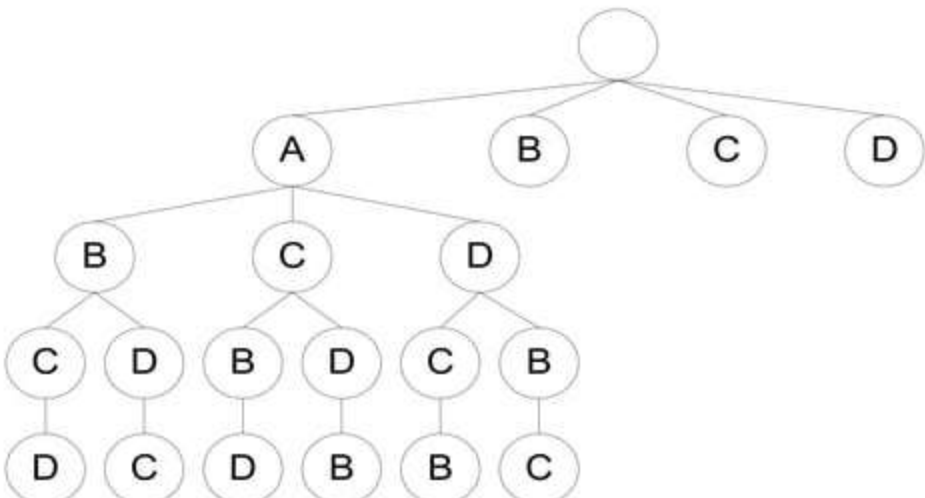
- Contoh kasus: Traveling Salesman Problem (TSP)

  Seorang salesmen ingin mengunjungi n kota. Jarak antara tiap-tiap kota sudah diketahui. Diinginkan rute terpendek dimana setiap kota sudah diketahui.



Contoh Kasus

- Penyelesaian dengan metode *Generate and Test*



Contoh Kasus

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

Dst…

Hill Climbing
(Pendakian Bukit)

- Hampir sama Generate and Test, perbedaan terjadi pada feedback dari prosedur test untuk pembangkitan keadaan berikutnya.

- Tes yang berupa fungsi heuristik akan menunjukkan seberapa baik nilai terkaan yang diambil terhadap keadaan lain yang mungkin

Simple Hill Climbing

- Algoritma:

1. Evaluasi keadaan awal, jika tujuan berhenti jika tidak lanjut dengan keadaan sekarang sebagai keadaan awal

2. Kerjakan langkah berikut sampai solusi ditemukan atau tidak ada lagi operator baru sebagai keadaan sekarang

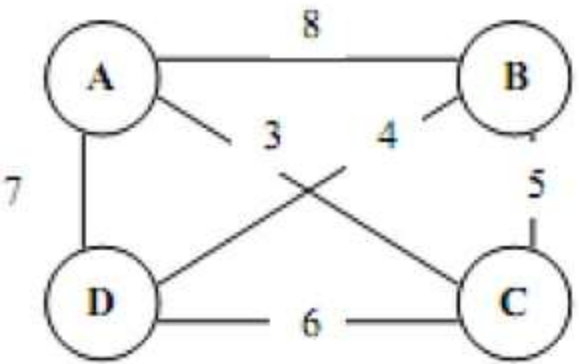
i. Cari operator yang belum pernah digunakan. Gunakan operator untuk keadaan yang baru.

ii. Evaluasi keadaan sekarang:
   a) Jika keadaan tujuan , keluar.
   b) Jika bukan tujuan, namun nilainya lebih baik dari sekarang, maka jadikan keadaan tersebut sebagai keadaaan sekarang
   c) Jika keadaan baru tidak lebih baik daripada keadaan sekarang, maka llanjutkan iterasi.

iii. Jika keadaan baru tidak lebih baik dari pada keadaan sekarang, maka lanjutkan interasi.

Traveling Salesman Problem Dengan *Simple Hill Climbing*

- Seorang salesman ingin mengunjungi n kota.
- Jarak antara tiap-tiap kota sudah diketahui.
- Kita ingin mengetahui rute terpendek dimana setiap kota hanya boleh dikunjungi tepat 1 kali.
- Misal ada 4 kota dengan jarak antara tiap-tiap kota seperti berikut ini :



Steepest – Ascent HC

- Gerakan pencarian selanjutnya berdasar nilai heuristik terbaik
- Algoritma:

1) Evaluasi keadaan awal, jika tujuan berhenti jika tidak lanjut dengan keadaan sekarang sebagai keadaan awal

2) Kerjakan hingga tujuan tercapai atau hingga iterasi tidak memberi perubahan sekarang.

i. Tentukan SUCC sebagai nilai heuristik terbaik dari successor-successor

ii. Kerjakan tiap operator yang digunakan oleh keadaan sekarang.

   a. Gunakan operator tersebut dan bentuk keadan baru
   b. Evaluasi keadaan baru. Jika tujuan keluar, jika bukan bandingkan nilai heuristiknya dengan SUCC. Jika lebih baik jadikan nilai heuristik keadaan baru ter sebut sebagai SUCC. Jika tidak, nilai SUCC tidak berubah.

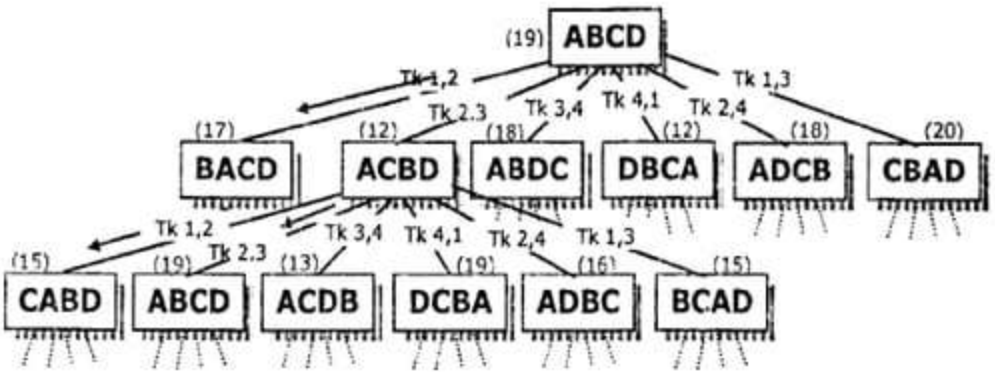
iii. Jika SUCC lebih baik dari nilai heuristik keadaan sekarang, ubah SUCC menjadi keadaan sekarang.

Pada steepest-ascent hill climbing ini, ada 3 masalah yang mungkin, yaitu:

- Local optimum: keadaan semua tetangga lebih buruk atau sama dengan keadaan dirinya.
- Plateou: keadaan semua tetangga sama dengan keadaan dirinya.
- Ridgez local optimum yang lebih disebabkan karena ketidak mampuan untuk menggunakan 2 operator sekaligus.

Traveling Salesman Problem Dengan *Steepest Ascent Hill Climbing*

Contoh 2.5: Traveling Salesman Problem dengan *steepest ascent hill climbing*.



Gambar 2.24 *Metode Steepest Ascent Hill Climbing.*

Best-First Search

- Metode yang membangkitkan suksesor dengan mempertimbangkan harga (didapat dari fungsi heuristik tertentu) dari setiap node
- Kombinasi dari BFS dan DFS
- Pencarian dilakukan dengan melihat satu lintasan, dan memungkinkan untuk berpindah ke lintasan lain.

Simulated Annealing (SA)

- SA memanfaatkan analogi antara cara pendinginan dan pembekuan metal menjadi sebuah struktur crystal dengan energi yang minimal (proses penguatan) dan proses pencarian untuk *state* tujuan minimal

- SA lebih banyak menjadi jebakan pada *local minimal*.

- *SA* berusaha keluar dari jebakan *minimum local*.

Algoritma: *Simulated Annealing*

1. Evaluasi keadaan awal. Jika tujuan maka KELUAR. Jika tidak lanjutkan dengan keadaan awal sebagai keadaan sekarang

2. Inisialisasi BEST_SO_FAR untuk keadaan sekarang

3. Inisialisasi T sesuai dengan annealing shedule

4. Kerjakan hingga solusi ditemukan atau sudah tidak ada operator baru lagi akan diaplikasikan kekondisi sekarang

a. Gunakan operator yang belum pernah digunakan untuk menghasilkan keadaan baru

b. Evaluasi kondisi baru dengan menghitung:

$$\Delta E = \text{nilai sekarang} - \text{nilaia keadaan baru}$$

i. Jika kondisi baru tujuan maka KELUAR

ii. Jika bukan tujuan, namun nilainya lebih baik dari sekarang, maka jadikan keadaan tersebut sebagai keadaaan sekarang

iii. Jika nilai kondisi baru tidak lebih baik daripada keadaan sekarang, maka tetapkan kondisi baru sebagai keadaan sekarang dengan probabilitas:

$$p' = e^{-\Delta E/T}$$

c. Perbaiki T sesuai dengan annealing scheduling

5. BEST_SO_FAR adalah jawaban yang dimaksud

OR Graph

- Dibutuhkan 2 antrian yang berisi node-node:
  - OPEN (berisi node-node yang sudah dibangkitkan, sudah memiliki fungsi heuuristik namun belum diuji)
  - CLOSED (berisi node-node yang sudah diuji)

- Fungsi lain yang dibutuhkan:
  - f'(n) : pendekatan dari fungsi f(n) (fungsi evaluasi terhadap node n)
  - g(n) : biaya yang dikeluarkan dari keadaan awal sampai ke node n
  - h'(n) : estimasi tambahan bbiaya yang harus dikeluarkan dari node n sampai mendapatkan tujuan.

Algoritma:

- Tempatkan node awal pada antrian OPEN
- Lakukan langkah berikut hingga tujuan ditemukan atau sampai antrian OPEN kosong
  - Ambil node terbaik dari OPEN
  - Bangkitkan semua successornya
  - Untuk tiap-tiap successornya kerjakan:
    - Jika node tersebut belum pernah dibangkitkan, evaluasi node tersebut dan masukkan ke OPEN
    - Jika node tersebut sudah pernah dibangkitkan sebelumnya, ubah parent jika lintasan baru lebih menjanjikan. Hapus node tersebut dari antrian OPEN.

Greedy Search

- Best First Search dengan hanya mempertimbangkan harga perkiraan (estimated cost)
- Harga sesungguhnya tidak digunakan
- Studi kasus:
  - Pencarian jalur dalam suatu daerah yang direpresentasikan dalam suatu graph. Node menyatakan kota dan busur menyatakan jarak antar kota (harga sesungguhnya) dan h'(n) adalah harga perkiraan dari node n menuju node tujuan (G).

- Dengan data sbb:
  I - A (75); A – B (85); B – G (300);
  I - C (140); C– D (160); D – G (200);
  I - E (120); E – F (180); F – G (250);

- Dengan h'(n) = fungsi heuristik (jarak garis lurus dari node n menuju G)

I	A	B	C	D	E	F
400	360	280	300	180	400	200

- Tentukan jalur terpilih?

Algoritma A*

- Perbaikan dari best-first search dengan memodifikasi fungsi heuristiknya.
- Meminimumkan total biaya lintasan.
- Fungsi f' sebagai estimasi fungsi evaluasi terhadap node n: $f'(n) = g(n) + h(n)$
- Jika:
  - h' = h : Proses pelacakan sampai pada tujuan
  - g = h' = 0, f' random: Sistem tidak dapat dikendalikan
  - g = k (konstanta) dan h' = 0 : Sistem menggunakan breadth first search
- Membutuhkan 2 antrian : OPEN dan CLOSED

Algoritma

1. Set : OPEN = {S}, dan CLOSED = { }, S: node awal
2. Kerjakan jika OPEN belum kosong:
3. Cari node n dari OPEN dimana nilai f(n) minimal. Kemudian tempatkan node n pada CLOSED
   a. Jika n adalah tujuan, keluar
   b. Ekspan node keanak-anaknya
   c. Kerjakan untuk setiap anak n, yaitu n':

- Jika n' belum ada di OPEN atau CLOSED, maka:
  - Masukkan n' ke OPEN. Kemudian set back pointer dari n' ke n.
  - Hitung:
    - h(n')
    - g(n') = g(n) + c(n,n') (biaya dari n ke n')
    - f(n') = g(n') + h (n')
- Jika n' telah ada di OPEN atau CLOSED dan jika g(n') lebih kecil (untuk versi n' yang baru), maka:
  - Buang versi lama n'
  - Ambil n' di OPEN, dan set backpointer dari n' ke n.