A* Algorithm

$$f(n) = g(n) + h(n)$$

f(n) : estimasi total cost dari start ke goal
g(n) : cost asli dari start node ke node n
h(n) : estimated cost dari node n ke target node

Hill Climbing Algorithm

1. Evaluasi intial state
2. Loop sampai solusi ketemu atau udah gak ada operator lagi
    a. Pilih operator baru
    b. Evaluasi state baru:
        i. Jika state baru = state goal maka selesai
        ii. Jika state baru lebih baik dibandingkan state sekarang maka buat state baru = state sekarang

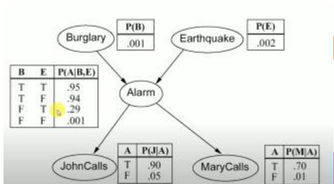
Problem yang dapat muncul di Hill Climbing Algorithm

1. Local Maximum
2. Plateau / Flat Maximum
3. Ridge

Bayesian Network

1. Pikirin semua kemungkinan yang terjadi
2. Hitung Probabilitasnya, kalau misalnya ada syarat yang harus diperhatikan maka dikalikan, kalau engga berarti dijumlahkan contoh

Ex:



Probabilitas dari Alarm bunyi tidak dikarenakan burglary atau earthquake terjadi, dan john dan marry menelepon.
Kemungkinan yang terjadi adalah:

- Burglary tidak terjadi, Earthquake tidak terjadi
- Alarm bunyi dikarenakan Burglary dan Earthquake tidak terjadi.
- Marry Menelepon karena Alarm bunyi
- John Menelepon karena Alarm bunyi

Maka dapat kita hitung dengan

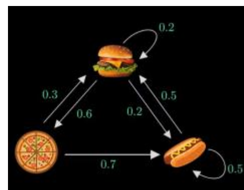
$$P(\neg B) \cdot P(\neg E) \cdot P(A|\neg B, \neg E) \cdot P(J|A) \cdot P(M|A)$$

Probabilitas dari john menelepon.
Kemungkinan yang terjadi adalah:

- John Menelepon karena Alarm Bunyi
- John Menelepon karena Alarm tidak bunyi
- Alarm bunyi karena Burglary, dan Earthquake terjadi
- Alarm bunyi karena Burglary, dan Earthquake tidak terjadi
- Alarm bunyi karena burglary terjadi, dan earthquake tidak terjadi
- Alarm bunyi karena burglary tidak terjadi, dan earthquake terjadi
- Alarm tidak bunyi karena Burglary, dan Earthquake terjadi
- Alarm tidak bunyi karena Burglary, dan Earthquake tidak terjadi
- Alarm tidak bunyi karena burglary terjadi, dan earthquake tidak terjadi
- Alarm tidak bunyi karena burglary tidak terjadi, dan earthquake terjadi

$$= P(j|a)\{P(a|b,e)*P(b,e)+P(a|\neg b,e)*P(\neg b,e)+P(a|b,\neg e)*P(b,\neg e)+P(a|\neg b,\neg e)*P(\neg b,\neg e)\}$$
$$+ P(j|\neg a)\{P(\neg a|b,e)*P(b,e)+P(\neg a|\neg b,e)* P(\neg b,e)+P(\neg a|b,\neg e)* P(b,\neg e)+P(\neg a|\neg b, \neg e)* P(\neg b, \neg e)\}$$

Markov Model



Ubah Kebentuk Matriks

*) perlu diperhatikan matriksnya gak boleh kebalik, kalo kebalik salah

Kalo misalkan dikasih probabilitas awal dan disuruh untuk mencari probabilitas dari sebuah rangkaian kejadian maka kita bisa melakukan perkalian matriks.
Ex:
Assume P(burger) =0,P(pizza) = 1, P(hotdog) = 0.
Kalau ingin mencari nilai probabilitas terjadinya burger, pizza, dan hotdog berurutan di 1 hari yang sama maka dilakukan

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.3 & 0 & 0.7 \end{bmatrix}$$

Anggapannya di hari kedua peluang burger, pizza dan hotdog adalah 0.3, 0 dan 0,7
Kalau ingin mencari hari n maka dilakukan perkalian nilai probabilitas hari n-1 dikalikan dengan matriksnya

$$\begin{bmatrix} 0.3 & 0 & 0.7 \end{bmatrix} \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.41 & 0.18 & 0.41 \end{bmatrix}$$

*) contoh hari kedua.

Tapi kalo misalkan kita disuruh untuk mencari probabilitas awal, kita harus menggunakan metode Eigenvector.

$$\pi A = \pi$$
$$Av = \lambda v$$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 3 & -2 & 0 \\ 2 & 3 & 4 \end{bmatrix}$$

Cara untuk mencari eigenvalue.

$$A - \lambda I = \begin{bmatrix} 1-\lambda & 0 & 0 \\ 3 & -2-\lambda & 0 \\ 2 & 3 & 4-\lambda \end{bmatrix}$$

$$\begin{vmatrix} 1-\lambda & 0 & 0 \\ 3 & -2-\lambda & 0 \\ 2 & 3 & 4-\lambda \end{vmatrix} = (1-\lambda)[(-2-\lambda)(4-\lambda) - (0)(3)] - 0 + 0$$

$$(1-\lambda)(-2-\lambda)(4-\lambda) = 0$$

$$\lambda = 1 \qquad \lambda = -2 \qquad \lambda = 4$$

*) ini cari determinan

Setelah udah dapat eigenvaluenya lanjut kita masukin eigenvalue kedalam matriks kita satu per satu

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 3 & -2 & 0 \\ 2 & 3 & 4 \end{bmatrix} \quad \begin{matrix} \lambda = 1 \\ \lambda = -2 \\ \lambda = 4 \end{matrix} \quad (A - \lambda I)\vec{x} = \vec{0}$$

$$A - (1)I = \begin{bmatrix} 1 & 0 & 0 \\ 3 & -2 & 0 \\ 2 & 3 & 4 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 3 & -3 & 0 \\ 2 & 3 & 3 \end{bmatrix}$$

$$3x_1 - 3x_2 = 0 \qquad 2 + 3 + 3x_3 = 0$$
$$x_1 = x_2 = 1 \qquad x_3 = -5/3 \qquad \vec{x} = \begin{bmatrix} 1 \\ 1 \\ -5/3 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 3 & -2 & 0 \\ 2 & 3 & 4 \end{bmatrix} \quad \begin{matrix} \lambda = 1 \\ \lambda = -2 \\ \lambda = 4 \end{matrix} \quad (A - \lambda I)\vec{x} = \vec{0}$$

$$A - (-2)I = \begin{bmatrix} 1 & 0 & 0 \\ 3 & -2 & 0 \\ 2 & 3 & 4 \end{bmatrix} - \begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 \\ 3 & 0 & 0 \\ 2 & 3 & 6 \end{bmatrix}$$

$$x_1 = 0 \qquad x_2 = -2$$
$$x_1 = 0 \qquad x_3 = 1 \qquad \vec{x} = \begin{bmatrix} 0 \\ -2 \\ 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 3 & -2 & 0 \\ 2 & 3 & 4 \end{bmatrix} \quad \begin{matrix} \lambda = 1 \\ \lambda = -2 \\ \lambda = 4 \end{matrix} \quad (A - \lambda I)\vec{x} = \vec{0}$$

$$A - (4)I = \begin{bmatrix} 1 & 0 & 0 \\ 3 & -2 & 0 \\ 2 & 3 & 4 \end{bmatrix} - \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix} = \begin{bmatrix} -3 & 0 & 0 \\ 3 & -6 & 0 \\ 2 & 3 & 0 \end{bmatrix}$$

$x_1 = 0$ this makes x_3 a

$x_2 = 0$ free variable

$$\vec{x} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Ambilah nilai vector yang gak ada nilai negatif dan jadikan probabilitas kita.

Intelligent Agent

- P(Performance)
  Sifat-sifat yang dihasilkan oleh AI
- E(Enviroment)
  Lokasi Dimana AI berinteraksi
- A(Actuators)
  Sesuatu yang dapat menggerakan AI
- S(Sensors)
  Cara agar AI mendapatkan input.

Agent Types

- Simple
  Action dilakukan berdasarkan kondisi sekarang
- Modal
  Action dilakukan berdasarkan adanya pemodelan untuk lingkungan
- Goal
  Action dilakukan berdasarkan kombinasi antara pemodelan yang dibuat dan tujuan
- Utility
  Action dilakukan berdasarkan utilitas

First-Order Logic (FOL):

- **Term:** Variabel, konstanta, dan fungsi.
  Contoh: f(x), di mana f adalah fungsi dan x adalah variabel.
- **Predikat:** Fungsi yang menghasilkan nilai benar atau salah.
  Contoh: P(x) berarti predikat P diterapkan pada x.
- **Kuantor:**
  - **Kuantor Universal (∀):** Merepresentasikan "untuk semua."
    Contoh: ∀x P(x) berarti "untuk semua x, P(x) benar."
  - **Kuantor Eksistensial (∃):** Merepresentasikan "ada."
    Contoh: ∃x P(x) berarti "ada x sehingga P(x) benar."
- **Konektif Logis:**
  - **Konjungsi (∧):** DAN
  - **Disjungsi (∨):** ATAU
  - **Negasi (¬):** TIDAK
  - **Implikasi (⇒):** JIKA-MAKA
  - **Bikondisional (⇔):** JIKA DAN HANYA JIKA

Conjunctive Normal Form (CNF):

- **Definisi:** Suatu formula berada dalam CNF jika berupa konjungsi dari disjungsi literal.
  - Contoh: (A ∨ B) ∧ (¬C ∨ D)
- **Langkah-Langkah Mengubah FOL ke CNF:**
  1. **Hilangkan Implikasi dan Bikondisional:** Gunakan ekuivalensi untuk menghapus ⇒ dan ⇔.
  2. **Pindahkan Negasi ke Dalam (Gunakan hukum De Morgan):** Pastikan negasi hanya diterapkan pada predikat individual.
  3. **Standarisasi Variabel:** Ganti nama variabel agar unik di setiap kuantor.
  4. **Skolemisasi:** Hapus kuantor eksistensial dengan memperkenalkan konstanta atau fungsi Skolem.
  5. **Hapus Kuantor Universal:** Setelah Skolemisasi, hapus semua kuantor universal.
  6. **Distribusikan Disjungsi terhadap Konjungsi:** Pastikan formula berada dalam CNF.

Resolusi:

- **Definisi:** Teknik pembuktian yang digunakan untuk menyimpulkan kesimpulan dari premis yang diberikan dalam FOL, berdasarkan CNF.
- **Langkah-Langkah Pembuktian Resolusi:**
  1. **Ubah Semua Pernyataan ke CNF:** Ubah semua premis dan negasi dari kesimpulan menjadi CNF.
  2. **Terapkan Aturan Resolusi:** Temukan klausa yang mengandung literal komplementer (misalnya, P dan ¬P) dan resolusi untuk membuat klausa baru.
  3. **Ulangi Hingga Mendapatkan Kontradiksi:** Lanjutkan resolusi sampai ditemukan kontradiksi (⊥, klausa kosong) atau tidak ada lagi resolusi yang mungkin.
- **Contoh:** Melakukan resolusi pada (P ∨ Q) dan (¬P ∨ R) menghasilkan (Q ∨ R).