12 - Branch & Bound

[KOMS120403]

Desain dan Analisis Algoritma (2022/2023)

Dewi Sintiari

Prodi S1 Ilmu Komputer
Universitas Pendidikan Ganesha

Week 13 (June 2023)

Daftar isi

BnB vs Backtracking

Mirip dengan Backtracking, **BnB adalah pencarian solusi dengan membangun** *pohon ruang-status (state-space tree)*, **dan simpul "pemangkasan" yang tidak mengarah ke solusi**.

Backtracking biasanya diterapkan untuk masalah non-optimasi, tetapi dapat diterapkan juga untuk masalah optimasi. Tapi **BnB diterapkan untuk masalah optimasi**.

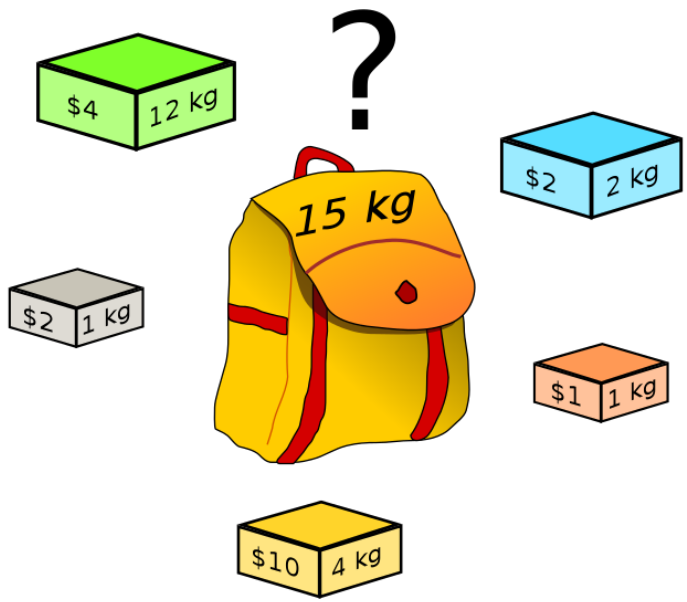
Dibandingkan dengan backtracking, branch-and-bound membutuhkan dua komponen tambahan:

- untuk setiap simpul dari pohon ruang-status, perlu menentukan batas untuk nilai terbaik dari fungsi tujuan $\rightarrow$ menggunakan **batas atas** (untuk masalah maksimisasi) dan **batas bawah** (untuk masalah minimisasi).
- perlu menentukan fungsi untuk nilai solusi terbaik yang terlihat sejauh ini (misalnya, nilai v pada rumus batas atas masalah knapsack).

Bagian 1. Knapsack problem (maximization)

1. Knapsack problem (1)

Diberikan n item dan ransel berkapasitas W. Setiap objek i memiliki bobot w_i dan profit v_i. Tentukan cara menyeleksi objek ke dalam ransel agar keuntungan maksimal. Berat total benda tidak boleh melebihi kapasitas ransel.



1. Knapsack problem (2)

Urutkan objek berdasarkan pada densitasnya, yaitu nilai $\frac{v_i}{w_i}$ dalam urutan menurun:

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \cdots \geq \frac{v_n}{w_n}$$

- Pohon ruang status yang dibangun adalah *pohon biner*. *Cabang kiri* menunjukkan objek i dipilih ($x_i = 1$), dan *cabang kanan* menunjukkan objek i tidak dipilih ($x_i = 0$).
- Setiap simpul pada kedalaman i dari pohon biner, untuk $i = 0, 1, 2, \ldots, n$ mewakili subset dari n objek yang dimasukkan ke dalam knapsack, dipilih dari i objek i.
- Setiap simpul dilengkapi dengan total bobot dan total profit dari objek yang dipilih.
- Batas atas profit dihitung berdasarkan rumus berikut:

$$ub = v + (W - w) \cdot \frac{v_{i+1}}{w_{i+1}}$$

di mana v dan w masing-masing adalah profit total dan berat total item yang sudah dipilih, W adalah kapasitas knapsack, dan $i + 1$ adalah indeks berikutnya yang akan diperhitungkan ke dalam solusi.

1. Knapsack problem (3)

Contoh: Diberikan $n = 4$, $K = 10$, dan karakteristik dari objek ditunjukkan pada tabel berikut.

item	weight	value	$\frac{\text{value}}{\text{weight}}$
1	4	\$40	10
2	7	\$42	6
3	5	\$25	5
4	3	\$12	4

The knapsack's capacity W is 10

1. Knapsack problem (4)

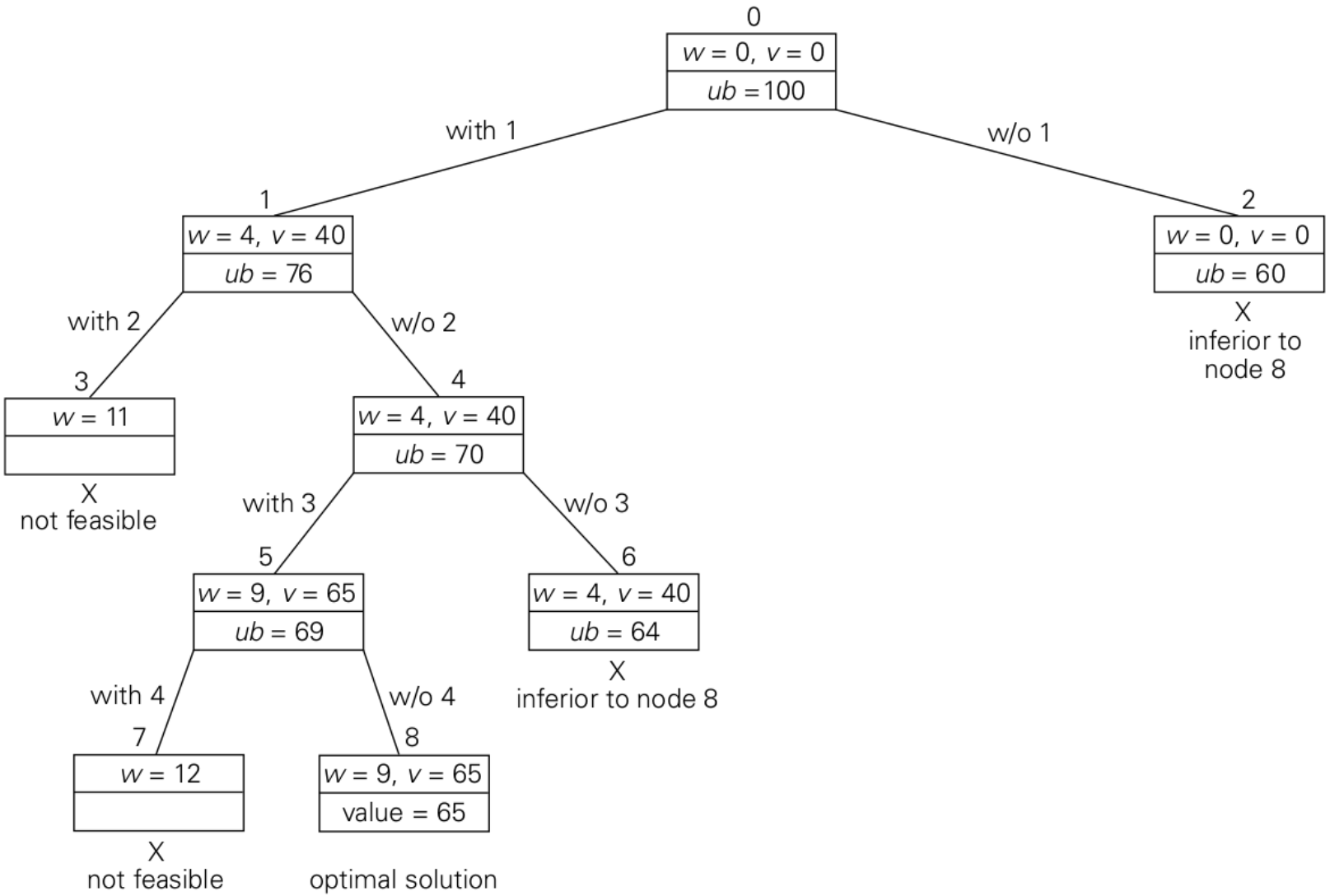


Figure: Pohon ruang-status untuk instance dari knapsack problem (*prunning* dilakukan karena bobot total sudah melebihi kapasitas yang diizinkan)

- Pada simpul akar, bobot total objek yang sudah dipilih w dan nilai totalnya v sama dengan 0. Nilai batas atas yang dihitung dengan rumus di atas adalah \$100.

- Simpul 1, anak kiri dari simpul akar, mewakili himpunan bagian yang menyertakan objek 1. Total bobot dan nilai objek yang sudah disertakan masing-masing adalah 4 dan \$40; nilai batas atas adalah $40 + (10 - 4) \cdot 6 = \$76$.

- Simpul 2 mewakili himpunan bagian yang tidak menyertakan objek 1. Dengan demikian, $w = 0$, $v = \$0$, dan $ub = 0 + (10 - 0) \cdot 6 = \$60$.

- Karena simpul 1 memiliki batas atas yang lebih besar daripada batas atas simpul 2, maka simpul 1 lebih menjanjikan untuk masalah maksimalisasi ini, dan kita menelusuri percabangan dari simpul 1 terlebih dahulu.

- Anak-anaknya, yakni simpul 3 dan 4, masing-masing mewakili himpunan bagian dengan objek 1, masing-masing "dengan" dan "tanpa" objek 2. Karena bobot total w dari setiap subset yang diwakili oleh simpul 3 melebihi kapasitas knapsack, simpul 3 dapat segera dihentikan.

- Simpul 4 memiliki nilai w dan v yang sama dengan simpul *parent*-nya sehingga batas atasnya sama dengan $ub = 40 + (10 - 4) \cdot 5 = \$70$.

- Untuk simpul lain, langkah-langkahnya serupa.

- Pada simpul 8, kita menemukan solusi yang layak untuk instance masalah ini (pada titik ini, kita belum mengetahui apakah ini merupakan solusi optimal).

- Untuk mengetahuinya, kita perlu melakukan runut balik (*backtrack*) ke simpul-simpul sebelumnya yang sudah dikunjungi namun merupakan *promising node*. Dalam hal ini, kita dapat memperhatikan bahwa simpul 6 memiliki batas atas yang lebih kecil dari simpul 8 (yaitu $64 < 65$), sehingga simpul 6 tidak perlu ditelusuri lebih lanjut, dan menjadi dead node.

- Kita melakukan runut balik lagi untuk mengunjungi simpul 2. Sama halnya dengan simpul 6, simpul 2 memiliki ub yang lebih kecil dari simpul 8 (yaitu $60 < 65$), sehingga simpul 2 tidak perlu ditelusuri lebih lanjut, dan menjadi dead node.

- Karena tidak ada lagi promising node yang dapat ditelusuri, maka penelusuran dihentikan, dan diperoleh solusi optimal yang diberikan oleh simpul 8, yaitu $X = \{1, 0, 1, 0\}$, atau objek yang diambil adalah objek nomor 1 dan 3.

1. Knapsack problem (5)

Analisis

Mengapa rumus batas atas didefinisikan sebagai berikut:

$$ub = v + (W - w) \cdot \frac{v_{i+1}}{w_{i+1}}$$

Apakah ini benar-benar *upper-bound* dari solusi optimal pada simpul status terkait?

Perhatikan bahwa semua objek diurutkan sesuai dengan densitas-nya, dengan urutan menurun, yaitu

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \cdots \geq \frac{v_n}{w_n}$$

Pada rumus batas atas di atas, nilai v adalah total profit sementara dari semua objek yang dipilih, dan nilai v_{i+1}/w_{i+1} adalah **densitas terbesar** di antara semua objek yang belum diperhitungkan ke dalam solusi. Jadi rumus tersebut memberikan batas atas untuk solusi jika penelusuran dilakukan pada simpul tersebut.

1. Knapsack problem (6)

Analisis

Lalu, apakah penelusuran bisa dilakukan secara DFS?

Untuk metode Branch-and-Bound, pembangunan pohon ruang status **dapat dilakukan secara BFS dan DFS**. Hal ini berbeda dengan metode Backtracking yang umumnya melakukan penelusuran secara DFS.

Bagian 2. Assignment problem (minimisasi)

2. Assignment problem (1)

Pasangkan n orang ke n pekerjaan sehingga total biaya dari penugasan tersebut sekecil mungkin.

Contoh dari masalah penugasan ditentukan oleh matriks biaya berukuran $n \times n$, yang dituliskan dengan matriks C.

$$C = \begin{array}{c} \begin{matrix} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{matrix} \\ \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \end{array} \begin{matrix} \\ \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{matrix}$$

Re-formulasi: pilih satu elemen di setiap baris matriks sehingga tidak ada dua elemen yang dipilih berada di kolom yang sama dan jumlahnya sekecil mungkin.

2. Assignment problem (2)

Analisis

Bagaimana kita dapat menemukan **batas bawah** pada biaya seleksi optimal tanpa benar-benar menyelesaikan masalah?

Observasi: untuk setiap kemungkinan solusi (termasuk solusi optimal), biaya (*cost*) tidak mungkin lebih kecil dari jumlah elemen terkecil di setiap baris matriks biaya (pada slide sebelumnya).

Dengan demikian, lower bound dapat ditentukan dengan cara menjumlahkan cost dari elemen pada baris yang sudah dipilih, ditambahkan dengan elemen terkecil dari setiap baris yang belum dipilih (*sebab ini adalah masalah minimasi, jadi kita mencari lower bound*).

Untuk lebih jelasnya, perhatikan slide berikutnya.

2. Assignment problem (3)

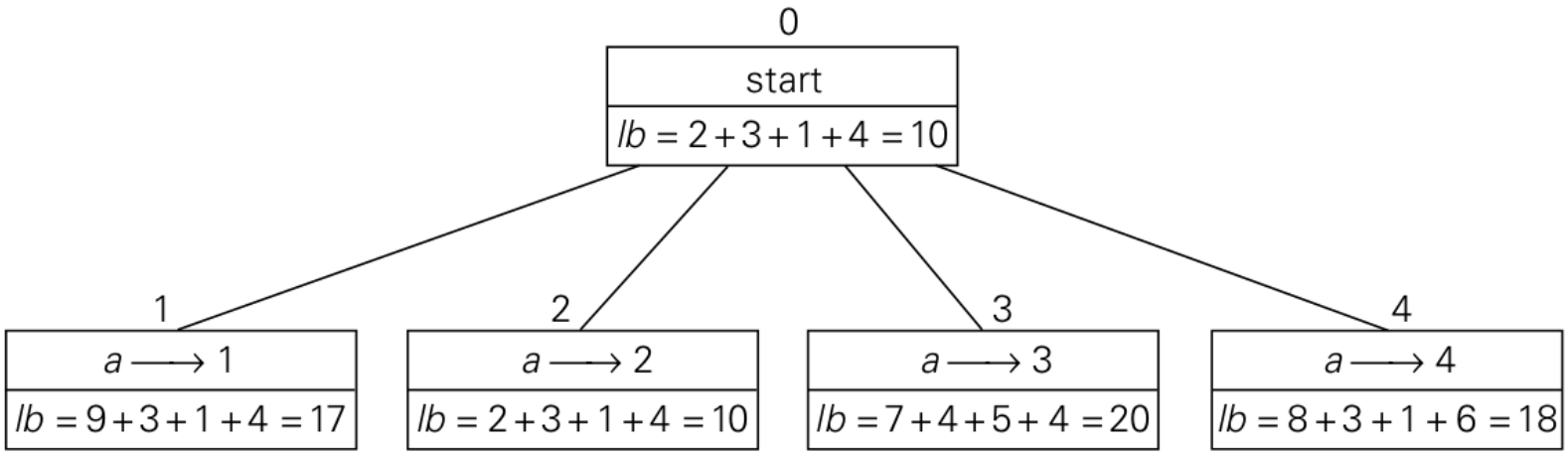


Figure: Level 0 dan 1 dari *state-space tree* untuk instance dari masalah penugasan

- Akar dibentuk oleh *batas bawah minimum*, diambil dari jumlah nilai minimum pada setiap baris.
- Anak-anak (*children*) dari simpul akar diberikan dengan menugaskan pekerja *a* untuk setiap kemungkinan pekerjaan.
- Simpul di level 1 yang memiliki biaya minimum diperluas, simpul lainnya dipangkas.
- Simpul-simpul baru pada level 2 dibangun dengan menugaskan pekerja *b*.
- Demikian seterusnya hingga semua pekerja diperhitungkan.

2. Assignment problem (3)

Analisis pohon ruang status

- Batas bawah dari simpul **akar** adalah: $2 + 3 + 1 + 4 = 10$
- Pada kedalaman 1 dari pohon ruang-status, untuk setiap pilihan yang sah yang memilih 9 untuk pekerja a (baris pertama), batas bawahnya adalah $9 + 3 + 1 + 4 = 17$. Untuk setiap pilihan sah yang memilih 2 untuk pekerja a (baris pertama), batas bawahnya adalah $2 + 3 + 1 + 4 = 10$, dan seterusnya.. .(lihat gambar di slide berikutnya)
- Pada kedalaman 1 pohon ruang status, kita pilih simpul yang memiliki batas bawah minimum sebagai simpul yang diperluas (sehingga solusi saat ini adalah $(a \rightarrow 2, b \rightarrow 1)$.
- Lanjutkan proses ini sampai semua staf ditugaskan dengan semua pekerjaan (yaitu sampai kita memilih satu elemen dari setiap baris dan kolom).
- Pohon ruang status lengkap ditunjukkan oleh gambar di slide berikutnya.

2. Assignment problem (5)

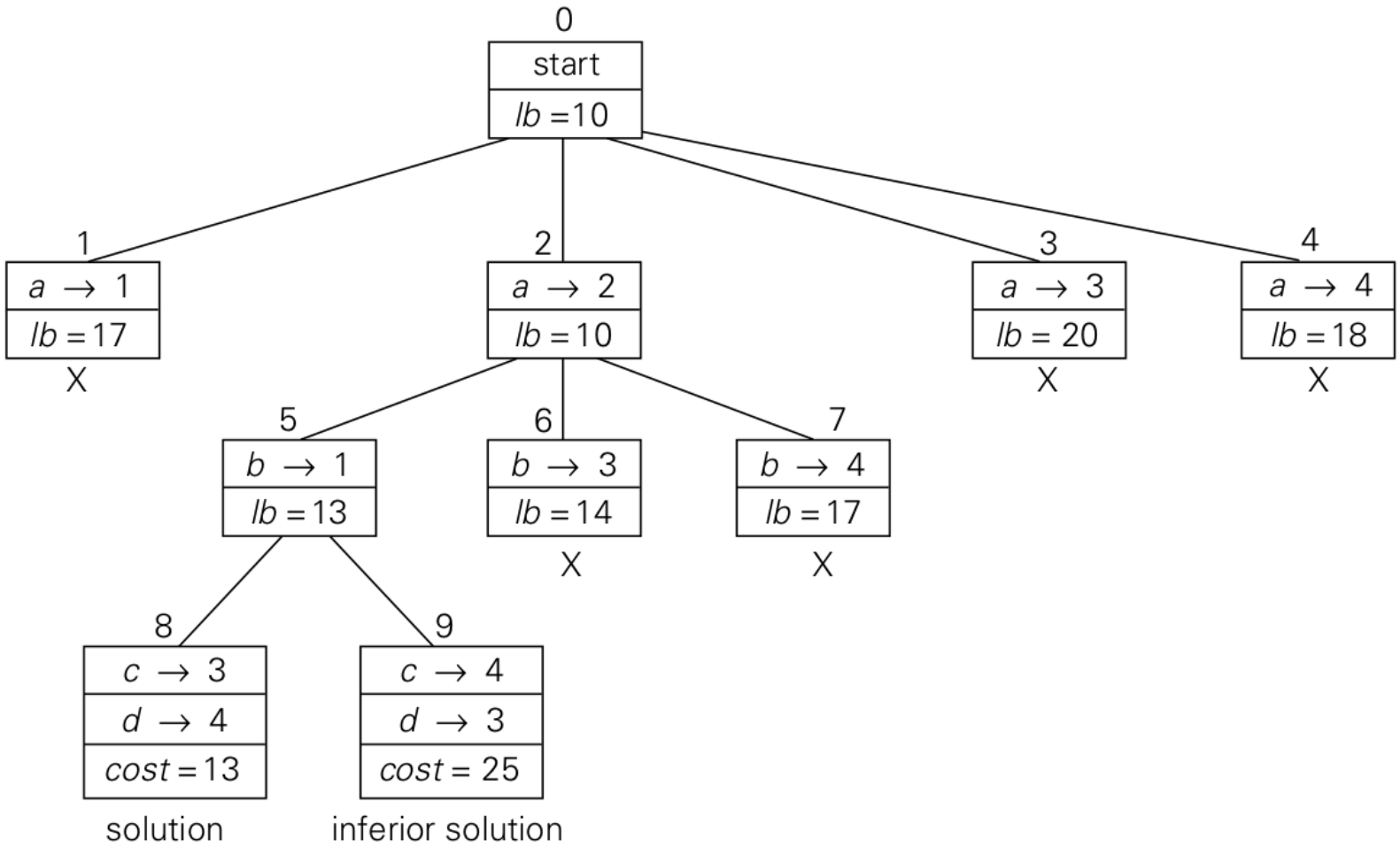


Figure: Pohon ruang status dari contoh masalah penugasan, dengan biaya optimal 13. Solusinya diberikan oleh: $X = (a \rightarrow 2, b \rightarrow 1, c \rightarrow 3, d \rightarrow 4)$

Kekurangan Branch and Bound

Meskipun BnB adalah teknik yang cukup bagus digunakan untuk menyelesaikan permasalahan optimasi, namun teknik ini memiliki keterbatasan dalam hal kompleksitas komputasional khususnya jika permasalahan yang diselesaikan berukuran besar dan kompleks.

Banyaknya sub-masalah dapat berkembang secara eksponensial (karena adanya percabangan pada setiap langkah).

Memang kita menggunakan *bound* untuk mengeliminasi sub-masalah, namun *bound* tersebut mungkin tidak cukup baik untuk dapat mengeliminasi banyak sub-masalah.

Dengan demikian, BnB mungkin membutuhkan banyak iterasi dan kebutuhan memori yang cukup tinggi untuk menemukan solusi optimal, atau untuk menunjukkan bahwa permasalahan tersebut tidak memiliki solusi optimal.

Bagaimana agar Branch and Bound dapat bekerja secara optimal?

Salah satu tantangan dari BnB adalah *bagaimana merancang strategi branching (membangun percabangan) yang memiliki sifat-sifat berikut.*

- Dapat mengkonstruksi *state-space tree* yang seimbang dan sub-permasalahan yang efisien.
- Dapat menyediakan batasan yang akurat dan terpercaya untuk sub-permasalahan.

Remark. Fungsi pembatas yang buruk dapat membentuk pohon yang *skewed* dan tidak efisien, dimana beberapa sub-masalah memiliki ukuran yang terlalu besar ataupun terlalu kecil. Di samping itu, fungsi pembatas yang tidak tepat juga dapat menyebabkan eksplorasi berjalan lambar dan tidak efektif, dimana sub-masalah tidak tereliminasi dengan optimal.

Latihan (petunjuk)

- Kerjakan secara berkelompok (2 orang).
- Kerjakan soal nomor 2 dan 3. Buatlah sebuah video untuk menjelaskan jawaban Anda. Unggah video Anda di Youtube.
- Setiap anggota kelompok mengerjakan masing-masing 1 soal (anggota pertama mengerjakan soal nomor 2 dan anggota kedua mengerjakan soal nomor 3). Lihat halaman berikutnya untuk membaca soal.
- Kumpulkan jawaban Anda dalam **satu file pdf** (dikumpulkan oleh **satu perwakilan kelompok**). Sertakan link video pada lembar jawaban.

Soal latihan

2. Buatlah sebuah instance untuk Integer Knapsack Problem dengan 4-6 item, dan aplikasikan algoritma Branch-and-Bound untuk menyelesaikan permasalahan tersebut dengan menggunakan fungsi pembatas seperti yang dibahas di kelas untuk menentukan *upper bound* pada setiap node-nya. Gambarkan pohon ruang statusnya, dengan memberikan nomor untuk menunjukkan urutan node yang dibangkitkan.

3. Buatlah sebuah instance untuk Assignment Problem dengan 4 job dan 4 staff, dan aplikasikan algoritma Branch-and-Bound untuk menyelesaikan permasalahan tersebut dengan menggunakan fungsi pembatas seperti yang tertera pada slide pembelajaran (textbook Levitin), untuk menentukan *lower bound* pada setiap node-nya. Gambarkan pohon ruang statusnya, dengan memberikan nomor untuk menunjukkan urutan node yang dibangkitkan.

end of slide...