Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

Team project ©2017
Dony Pratidana S. Hum | Bima Agus Setyawan S. IIP

BAB II

LANDASAN TEORI

2.1 Optimasi

Menurut Sun & Yuan (2006), pada umumnya, optimasi diterapkan pada matematika, perhitungan matematika, dan penelitian operasi yang memiliki aplikasi yang luas dalam ilmu pengetahuan, teknik, manajemen bisnis, militer, dan teknologi ruang. Kriteria yang terikat dalam pencarian solusi yang optimal dari suatu masalah akan didefinisikan secara matematis, solusi terbaik dari suatu masalah dapat ditemukan berdasarkan beberapa skema melalui metode ilmiah dan alat. Dalam rangka menemukan solusi yang optimal dari suatu masalah dapat dilakukan dengan membangun model masalah, menentukan algoritma untuk mencapai solusi, membentuk teori konvergensi algoritma, dan pengalaman numerik dengan masalah tipikal atau masalah kehidupan nyata. Saat ini, beragam metode optimasi *modern* dapat menyelesaikan masalah optimasi yang sulit dan berskala besar, serta menjadi alat yang sangat diperlukan untuk menyelesaikan masalah di berbagai bidang.

UNIVERSITAS
MULTIMEDIA
NUSANTARA

2.2 Manajemen Persediaan

Mengacu pada sumber bacaan, menurut (Bose, 2006) persediaan adalah sumber daya apapun yang memiliki nilai ekonomi. Persediaan terdiri dari bahan baku, barang jadi, pemakaian, dan penyimpanan. Mengatur persediaan adalah tentang merencanakan dan merancang prosedur untuk menjaga tingkat optimal sumber daya. Manajemen persediaan yang baik sama juga dengan manajemen keuangan yang baik. Berdasarkan pengamatan, "ketika anda butuh uang, maka lihat persediaan anda terlebih dahulu sebelum anda melihat bankir anda". Penggunaan modal yang efisien dalam usaha dapat membantu meningkatkan layanan pelanggan dan keuntungan. Tujuan tersebut dapat dicapai dengan sejumlah modal tertentu baik dengan memaksimalkan pengeluaran atau dengan memaksimalkan batas keuntungan, atau dengan memaksimalkan keduanya. Itu berarti bahwa manajemen harus dapat mencari akal untuk membuat keuntungan secepat mungkin, yang seringkali sulit untuk diterapkan. Dan, sulit juga untuk meningkatkan batas keuntungan secara luas karena persaingan dalam bisnis. Dengan demikian, perputaran modal dan produktivitas terkadang tidak menjadi efektif.

Beberapa teknik modern telah dikembangkan dan diterapkan oleh manajer untuk menemukan solusi dari suatu masalah (Bose, 2006). Diantaranya, manajemen persediaan adalah yang paling efektif karena memungkinkan manajer untuk meningkatkan produksi dalam mendapat keuntungan dengan mengurangi biaya bahan, mencegah modal kerja yang besar dalam jangka waktu yang panjang dan meningkatkan pertukaran modal. Konsep dari manajemen persediaan telah menjadi salah satu dari banyak aspek analitis dari manajemen. Manajemen persediaan melibatkan optimasi

sumber daya yang tersedia untuk menyimpan stok berbagai macam bahan. Kurangnya persediaan dapat menyebabkan persediaan habis, menyebabkan penghentian produksi, tetapi persediaan yang sangat tinggi di sisi lain dapat mengakibatkan peningkatan biaya produksi akibat tingginya biaya dalam memuat persediaan. Sehingga dalam mengoptimasikan persediaan harus memastikan stok tidak terlalu banyak atau terlalu sedikit.

Pendekatan yang paling mendasar untuk menata ulang persediaan adalah dengan teknik reorder point. Reorder point digunakan sebagai titik yang ditetapkan sebagai sinyal untuk melakukan pengisian ulang. Berikut adalah rumus yang digunakan untuk menghitung reorder point (Mercado, 2007).

Reorder point = lead time in days demands per day ...(2.1)

2.3 Dynamic Programming

Dynamic Programming adalah metode yang digunakan untuk memecahkan masalah dengan melibatkan beberapa urutan keputusan, solusi yang optimal dari suatu masalah didapatkan berdasarkan solusi optimal dari setiap bagian masalah. Untuk Dynamic Programming menjadi komputasi yang efisien harus ada *subproblem* umum dari salah satu *subproblem* lainnya, solusi untuk *subproblem* hanya perlu ditentukan sekali dan dapat digunakan untuk *subproblem* berikutnya (Lew dan Mauch, 2007).

Algoritma Dynamic Programming terdiri dari empat langkah sebagai berikut.

1. Karakterisasi struktur dari solusi optimal.
2. Mendefinisikan nilai dari solusi optimal secara rekursif.
3. Hitung nilai dari solusi optimal secara *bottom-up*.

Rancang Bangun Sistem..., Michaela Irene, FTI, 2017

4. Membangun solusi optimal berdasarkan informasi komputasi.

Langkah 1-3 merupakan landasan dasar Dynamic Programming untuk menemukan solusi terhadap masalah. Langkah 4 digunakan untuk menemukan solusi optimal yang didapat dari informasi yang terdapat pada langkah 3 (Cormen, 2007).

Pendekatan Iterative Dynamic Programming menggunakan cara lain yaitu dengan mengisi tabel secara langsung tanpa menggunakan fungsi rekursif. Pada akhirnya dapat ditemukan solusi dari permasalahan awal yang terdapat di dalam tabel (Tseng, 2008).

Berikut adalah *pseudocode* Dynamic Programming untuk *knapsack problem* (Hristakeva, 2005).

```
For i = 0 to N do
        For j = 0 to Capacity
                If  j < Weights[i]  then
                        Table[i, j] ← Table[i-1, j]
                Else
                        Table[i, j] ← maximum { Table[i-1, j] AND
                                                Values[i] + Table[i-1, j-Weights[i]] }
Return Table[N, Capacity]
```

Gambar 2.1 *Pseudocode* Dynamic Programming

UNIVERSITAS
MULTIMEDIA
NUSANTARA

Berikut adalah *pseudocode* Dynamic Programming untuk menentukan item yang dipilih dari solusi optimal (Hristakeva, 2005).

```
n←N   c←C
Start at position Table[n, c]
While the remaining capacity is greater than 0 do
        If Table[n, c] = Table[n-1, c] then
                Item n has not been included in the optimal solution
        Else
                Item n has been included in the optimal solution
                Process Item n
                Move on row up to n-1
                Move to column c-weight[n]
```

Gambar 2.2 Dynamic Programming untuk Menentukan Item Dipilih

2.4 Knapsack Problem

Knapsack problem (Kellerer, dkk., 2014) adalah permasalahan optimasi pemilihan benda yang akan ditampung ke dalam suatu wadah berkapasitas terbatas. Optimasi ini dilakukan agar proses pemilihan dapat menghasilkan keuntungan yang maksimal. Setiap benda yang akan dimasukkan memiliki nilai dan berat yang berbeda sebagai penentu prioritas pada pemilihan tersebut. *Knapsack problem* terdiri dari beberapa jenis antara lain 0/1 *knapsack problem*, *fractional knapsack problem*, *bounded knapsack problem*, dan *unbounded knapsack problem*. Pada 0/1 *knapsack problem*, kemungkinan pemilihan yang diambil antara 1 dan 0 sehingga dapat diformulasikan sebagai formulasi 0/1 knapsack problem berikut ini.

$$\text{maximize} \sum_{j=1}^{n} p_j \, x_j$$

$$\text{subject to} \sum_{j=1}^{n} w_j x_j \leq c,$$

$$x_j \in \{0,1\},\ j = 1, \dots, n \qquad \text{...(2.2)}$$

Benda yang dipilih pada *fractional knapsack problem* hanya dalam bentuk pecahan, benda tersebut dapat dibagi-bagi seperti tepung, gula, dan sebagainya. Pada *bounded knapsack problem*, setiap benda tersedia sebanyak N unit dan jumlahnya terbatas, sedangkan pada *unbounded knapsack problem* tersedia lebih dari 1 unit dan jumlahnya tidak terbatas.

2.4.1 Multiple Constraint Knapsack Problem

Multiple Constraint Knapsack Problem (MCKP) atau dapat disebut juga dengan *Multidimensional Knapsack Problem*, merupakan salah permasalahan kombinatorial NP-hard. Pada MCKP, setiap pilihan item memiliki lebih dari satu dimensi pilihan. Permasalahan ini bertujuan untuk memperoleh solusi optimal dengan memilih kombinasi item berdasarkan beberapa batasan dan tidak melebihi kapasitas yang tersedia. MCKP dapat dirumuskan sebagai Formulasi MCKP berikut ini (Kellerer, dkk., 2014).

$$\text{maximize} \sum_{j=1}^{n} p_j x_j$$

$$\text{subject to} \sum_{j=1}^{n} w_{ij} x_j \leq c_i,\ \ i = 1, \dots, d$$

$$x_j \in \{0,1\},\ j = 1, \dots, n \qquad \text{...(2.3)}$$