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.

BAB II

LANDASAN TEORI

2.1 Warehouse Management System

Warehouse Management System (WMS) secara sederhana dapat digunakan untuk melacak data barang. Saat ini, WMS tidak hanya digunakan untuk manajemen dalam ruang lingkup gudang saja, tetapi juga dapat digunakan untuk manajemen inventori, manajemen biaya, dan semua manajemen lainnya yang berhubungan dengan manajemen gudang. Manajemen gudang yang efektif dan efisien akan berpengaruh terhadap rantai distribusi.

Manajemen gudang mengawasi semua aktifitas yang dilakukan terhadap suatu barang yang melewati gudang. Manajemen gudang berhubungan dengan penerimaan, penyimpanan, dan perpindahan barang. Dalam manajemen gudang sendiri, terdapat manajemen terhadap tempat penyimpanan barang dan proses memuat serta membongkar barang. Infrastruktur fisik dari gudang, sistem pelacakan, dan sistem komunikasi diperlukan dalam proses manajemen gudang tersebut. Tujuan dari manajemen gudang yaitu untuk mengelola sumberdaya secara ekonomis agar pengeluaran biaya dari waktu penyelesaian pesanan dapat dioptimalkan.

Alasan-alasan mengapa WMS digunakan antara lain:

1) Menyederhanakan pertukaran informasi dan perpindahan barang pada gudang atau mengurangi proses yang tidak efisien.
2) Mengurangi biaya yang dikeluarkan untuk transportasi dan pengiriman.

Rancangan Bangun ..., Ardy Wibowo Haryanto,FTI UMN, 2014

3) Meningkatkan akurasi data barang beserta stoknya yang kemudian akan memperbaiki kinerja dalam memenuhi pesanan.
4) Mengurangi beban kerja.
5) Membantu dalam membuat perencanaan yang lebih baik.

"Detail untuk persiapan dan pemrosesan menggunakan WMS bisa saja sangat berbeda dari vendor software satu dengan yang lain, bagaimanapun juga logika dasarnya akan menggunakan kombinasi dari barang, lokasi, jumlah, ukuran, dan informasi pesanan untuk memeriksa dimana menaruh, dimana mengambil, dan urutan kegiatan yang harus dilakukan untuk pekerjaan ini" (Piasecki: 2003).

2.2 Manajemen Penyimpanan

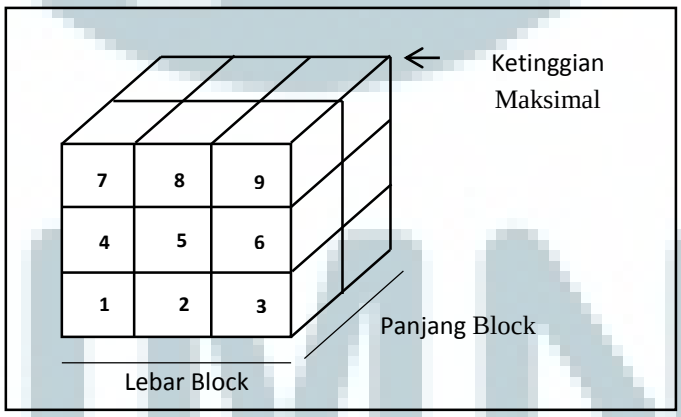
Manajemen adalah penggunaan sumber daya organisasi untuk mencapai sasaran dan kinerja yang tinggi baik untuk organisasi yang mencari keuntungan maupun yang tidak. Manajemen sebagai seni atau proses dalam menyelesaikan sesuatu yang terkait dengan pencapaian tujuan (Sule: 2008). Penyimpanan barang adalah menempatkan barang dalam kondisi tunggu untuk diorder atau dipersiapkan untuk diproses selanjutnya. Penyimpanan dilakukan sesuai dengan karakteristik barang. Jadi, manajemen penyimpanan barang yaitu penggunaan sumber daya organisasi untuk mencapai sasaran dan kinerja yang tinggi baik untuk organisasi yang mencari keuntungan maupun yang tidak dalam hal penempatkan barang dalam kondisi tunggu untuk diorder atau dipersiapkan untuk diproses selanjutnya.

Putaway adalah aktifitas penempatan barang yang sudah sesuai dengan dokumen ke tempat penyimpanan barang pada lokasi yang telah disediakan. Terdapat

dua cara dalam melakukan putaway, yaitu *direct putaway* (penempatan barang secara langsung) dan *directed putaway* (penempatan barang yang diarahkan oleh sistem).

2.3 Block stacking

Block stacking merupakan salah satu metode penataan barang pada gudang. Metode block stacking tidak memerlukan peralatan penyimpanan tertentu. Barang yang ditata dengan menggunakan metode ini akan disimpan secara berjajar dan ditumpuk kearah atas menjadi sebuah blok. Barang yang disimpan dengan menggunakan metode ini dapat diambil dengan menggunakan metode *Last In First Out* (LIFO). Terdapat beberapa kriteria yang harus diperhatikan dalam menumpuk barang seperti berat barang, panjang dan lebar blok, dan ketinggian maksimal dari tumpukan yang diperbolehkan. Secara umum, prioritas posisi penempatan barang dapat digambarkan sebagai berikut.



Gambar 2.1 *Block Stacking* (Sumber: Hambali: 2011)

Barang ditata mulai dari ujung kiri bawah (nomor 1) dengan prioritas ke kanan sepanjang yang telah ditentukan. Setelah itu, penataan barang akan diulang lagi dari kiri dengan posisi ditumpukkan di atasnya.

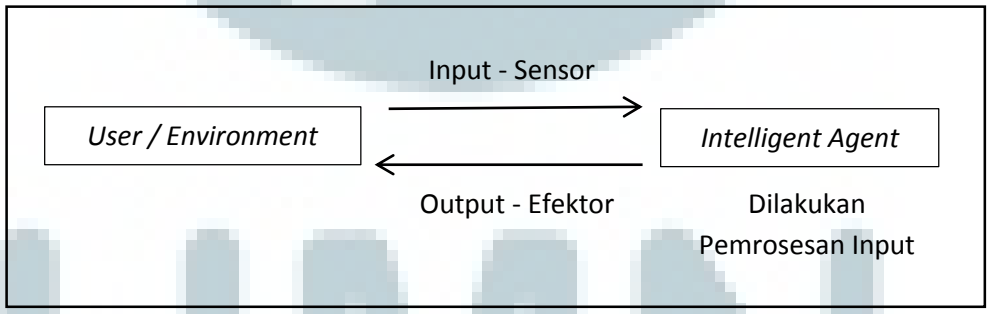
2.4 Artificial Intelligence

Artificial intelligence merupakan salah satu cabang dari ilmu komputer. Artificial intelligence bertujuan untuk memberikan kecerdasan berpikir dan berperilaku kepada suatu mesin.

Terdapat empat kategori artificial intelligence yaitu:

1) Sistem yang bertindak seperti orang. Contoh: Turing test.
2) Sistem yang berpikir seperti orang. Contoh: Cognitive science.
3) Sistem yang berpikir secara rasional. Contoh: Logical approach.
4) Sistem yang bertindak secara rasional. Contoh: Intelligent agent approach.

Dalam artificial intelligence terdapat istilah *intelligent agent*. *Intelligent agent* yaitu sistem yang dapat menerima input dari *user* atau *environment* melalui sensor dan kemudian input tersebut akan diproses serta akan menghasilkan output yang sesuai melalui efektor.



Gambar 2.2 *Intelligent Agent* (Sumber: Russel: 2009)

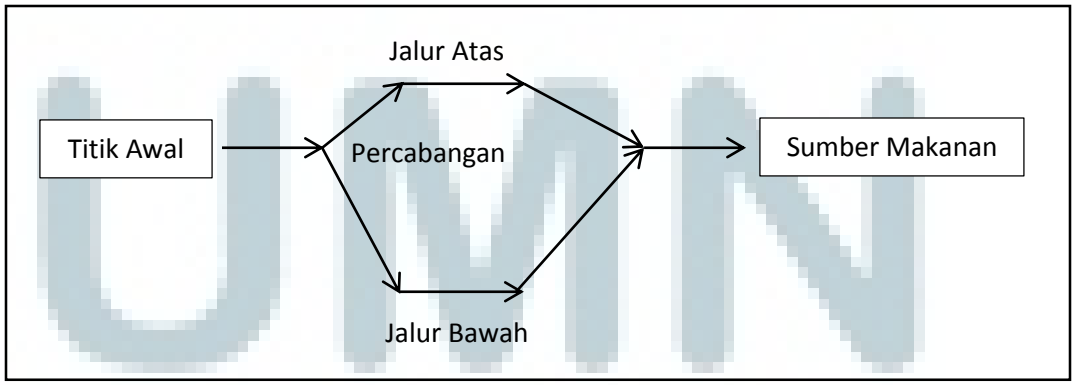
2.5 Algoritma Ant Colony

Algoritma ant colony merupakan salah satu algoritma yang menggunakan pendekatan heuristik. Algoritma ini dapat menyelesaikan suatu permasalahan dengan menemukan solusi yang baik. Algoritma ant colony lebih baik dari algoritma

penguatan tiruan (*simulaten annealing*) dan algoritma genetic. Dalam situasi yang mungkin berubah secara dinamis, algoritma ant colony dapat berjalan secara berkelanjutan dan menyesuaikan dengan perubahan yang terjadi dalam waktu saat ini *(real time).*

Algoritma ant colony dibuat berdasarkan kebiasaan semut dalam mencari makan dari sarangnya ke sumber makanan. Koloni semut secara alami dapat mencari rute terpendek dari sarangnya hingga mencapai sumber makananya. Pertama kali, koloni semut melakukan pencarian rute secara acak. Semut mempunyai *pheromone trail* (semacam bau-bauan). Semut menggunakan *pheromone trail* untuk meninggalkan jejak pada rute yang telah dilaluinya. Oleh karena itu, semut lainnya akan mengikuti jalur yang terdapat aroma *pheromone trail* tersebut untuk mencari tujuan. Aroma *pheromone trail* yang berada di rute terpendek lebih kuat dari aroma *pheromone trail* yang berada di rute terjauh karena aroma *pheromone trail* yang berada di rute terjauh akan cepat menguap. Hal ini dikarenakan semut jarang melewati rute tersebut.

Cara kerja Algoritma ant colony dapat dijelaskan sebagai berikut.



Gambar 2.3 Cara Kerja Algoritma Ant Colony (Sumber: Devi: 2012)

1) Pertama kali, semut-semut berkeliling secara acak.

2) Saat semut-semut menemukan jalur yang bercabang, semut-semut akan menentukan arah jalan secara acak. Sebagian semut akan memilih jalur bagian atas dan sebagian semut lainnya akan memilih jalur bagian bawah.
3) Saat semut berhasil menemukan sumber makanan, semut akan kembali ke koloninya sekaligus meninggalkan jejak dengan menggunakan *pheromone trail.*
4) Karena jalur bagian atas lebih pendek, maka semut yang melalui jalur bagian atas akan tiba ke sumber makanan terlebih dahulu dengan asumsi kecepatan semua semut sama.
5) *Pheromone trail* yang ditinggalkan semut di jalur bagian atas memiliki aroma yang lebih kuat dari *pheromone trail* yang ditinggalkan semut di jalur bagian bawah.
6) Semut-semut lainnya akan lebih tertarik mengikuti jalur bagian atas yang memiliki aroma *pheromone trail* yang lebih kuat.

Penguapan *pheromone trail* diperlukan untuk mencegah konvergensi pada penyelesaian. Jika tidak terdapat penguapan p*heromone trail*, jalur yang dipilih semut pertama akan tetap menarik semut-semut lainnya untuk dilewati. Penguapan *pheromone trail* menyebabkan eksplorasi ruang penyelesaian menjadi lebih terbatasi.

Terdapat empat macam metode algoritma ant colony, yaitu:

1) Foraging

Mengimplementasikan tingkah laku semut dalam mencari makan untuk mencari jalur tercepat dalam menemukan solusi.

2) Division of Labor

Mengimplementasikan tingkah laku semut dalam membagi pekerjaan menjadi bagian-bagian tertentu dengan hak-hak tertentu.

3) *Cemetry Organization and Brood Sorting*

Mengimplementasikan tingkah laku semut dalam membangun makan atau ketika akan bertelur dimana semut-semut menyusunnya dengan rapi.

4) *Cooperative Transport*

Metode yang menggabungkan berbagai macam tingkah laku semut untuk menghasilkan suatu sistem.

Diperlukan beberapa variabel yang perlu diinisialisasi dalam ant algoritma ant colony, variabel-variabel tersebut yaitu sebagai berikut.

1) Intensitas *pheromone trail* untuk antar *state* (T_{ij}).
2) Visibilitas dari suatu solusi yang akan dipilih oleh semut (η_{ij}).
3) Parameter pengendali intensitas *pheromone trail* (α), nilai $\alpha \geq 0$.
4) Parameter pengendali visibilitas (β), nilai $\beta \geq 0$.
5) Banyak semut (m).
6) Banyak *state* (n).
7) Tetapan local *pheromone trail* (σ), nilai $0 > \sigma > 1$.
8) Tetapan penguapan *pheromone trail* (ρ), nilai $0 > \rho > 1$.

Seekor semut akan bergerak dari *node* i ke *node* j dengan probabilitas:

$$P_{ij} = \frac{(T_{ij}^{\alpha})(\eta_{ij}^{\beta})}{\sum(T_{ij}^{\alpha})(\eta_{ij}^{\beta})} \quad \text{..... Rumus 2.1}$$

Sebuah *pheromone trail* lokal diupdate untuk setiap solusi yang dibangun untuk menghalangi solusi berikutnya untuk menggunakan komponen yang sama dalam urutan yang sama, sebagai berikut.

$$T_{ij} = (1 - \sigma)T_{ij} + \sigma T_{ij}^{0} \quad \text{..... Rumus 2.2}$$

Keterangan,

T_{ij}^{0} = Nilai inisialisasi jumlah *pheromone trail* yang diendapkan.

Pada akhir setiap iterasi, *pheromone trail* pada semua *edge* yang dilalui / *edge* yang mengantarkan ke solusi yang ditemukan, diperbarui dan dihancurkan.

$$T_{ij} = (1 - \rho)T_{ij} + \rho\Delta T_{ij}$$ ..... Rumus 2.3

Keterangan,

ΔT_{ij} = Jumlah *pheromone trail* yang diendapkan.

Berikut merupakan pseudocode dari algoritma ant colony.

```
Initialize pheromone trail
Do While (kriteria untuk berhenti tidak terpenuhi)
    Do While (Setiap semut belum menyelesaikan perjalanan)
        LocalPheromoneTrailUpdate()
    End Do
    AnalyzeTour()
    GlobalPheromoneTrailUpdate()
End Do
```

2.6 Algoritma Backtracking

Algoritma backtracking merupakan algoritma yang sering digunakan untuk memecahkan permasalahan yang bersifat kombinasi. Prinsip dasar dari algoritma backtracking yaitu mencoba satu demi satu kemungkinan yang bisa dilakukan untuk memperoleh solusi yang terbaik. Algoritma backtracking bekerja secara rekursif dan melakukan pencarian solusi suatu persoalan secara sistematis dari semua kemungkinan

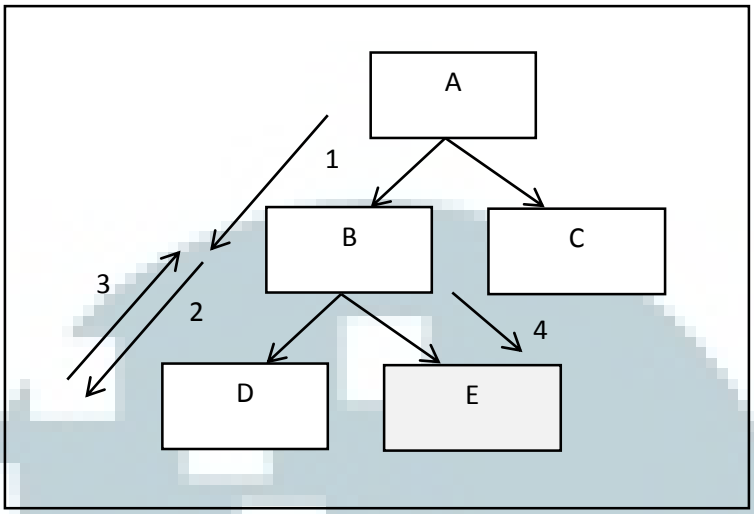
solusi yang ada. Algoritma backtracking berbasiskan pada Depth-First-Search (DFS) algorithm.

Dalam algoritma backtracking, solusi-solusi yang ada disusun ke dalam bentuk pohon solusi (*tree*). Dari *tree* tersebut, dilakukan pencarian dengan menggunakan algoritma DFS sehingga ditemukan solusi terbaik.

Langkah-langkah pencarian solusi dengan menggunakan algoritma backtracking:

1) Solusi dicari dengan melalui lintasan dari *root* ke *leaf*. *Node* yang sudah dilahirkan disebut *life node*. *Life node* yang diperluas disebut *E-node* (*Expand-node*).
2) Jika lintasan yang diperoleh dari peluasan *E-node* tidak mengarah ke solusi, maka *node* tersebut akan menjadi *dead node*. *Dead node* tidak akan diperluas lagi.
3) Jika posisi terakhir berada di *dead node*, maka pencarian dilakukan dengan membangkitkan *child node* lainnya. Jika tidak terdapat *child node*, maka dilakukan backtracking ke *parent node*.
4) Pencarian berhenti ketika solusi telah ditemukan atau tidak ada *life node* yang dapat diperlukan.

Berikut ini merupakan gambar simulasi pencarian solusi dengan menggunakan algoritma backtracking.



Gambar 2.4 Simulasi Pencarian Solusi dengan Menggunakan Algoritma Backtracking

(Sumber: Teneng: 2010)

2.7 Algoritma Depth First Search

Depth-First-Search (DFS) algorithm merupakan algoritma yang melakukan pencarian secara mendalam. Algoritma DFS melakukan penelusuran dengan mengunjungi secara rekursif salah satu dari sejumlah *node* yang dibangkitkan sebelum pindah ke *child node* yang lain dari *root node*. Waktu yang dibutuhkan oleh algoritma DFS sangat dipengaruhi oleh posisi suatu node pada suatu kedalaman dari *tree* pencarian. Kedalaman *tree* yang tidak terbatas dapat menyebabkan pencarian dengan menggunakan algoritma DFS tidak akan berhenti.