2. LANDASAN TEORI

2.1. Tinjauan Pustaka

2.1.1. *Warehouse Inventory Order Picking*

Warehouse inventory order picking merupakan proses pengambilan sejumlah barang dari lokasi penyimpanan gudang untuk memenuhi pesanan pelanggan (Petersen, C. G., 1997). Sebagai proses yang memakan paling banyak membutuhkan tenaga dan biaya, proses ini menjadi prioritas utama bagi perusahaan untuk meningkatkan produktivitas mereka. Terdapat 4 teknik pengambilan barang yang sering digunakan dalam pelaksanaan *order picking,* yaitu pengambilan barang berdasarkan *batch, zone, wave,* dan *discrete* (Lopienski, 2020). Penentuan teknik pengambilan barang sebaiknya didasarkan pada ukuran gudang, banyak dan jenis barang yang ada di dalamnya, serta banyak staf gudang.

2.1.2. Algoritma *Pathfinding*

Algoritma *Pathfinding* merupakan algoritma yang dikembangkan pada skripsi ini dengan tujuan untuk menghasilkan urutan barang yang akan diambil. Proses *order picking* dimulai dengan staf gudang berangkat dari titik depo, mengambil semua barang yang dibutuhkan, lalu kembali ke titik depo (bisa satu atau dua titik yang berbeda). Proses ini menyerupai konsep *Traveling Salesman Problem* (TSP), dan variasinya yaitu *Open Loop Travelling Salesman Problem* (OTSP). Pada TSP, seorang *salesman* mulai berpindah dari *node* awal, dan kembali ke tempat awal setelah bertemu dengan *n* pelanggan dengan syarat setiap pelanggan hanya ditemui satu kali (Yousefikhoshbakht, 2021). Sementara pada OTSP, salesman masih harus mengunjungi setiap pelanggan namun tidak kembali ke *node* awal (Chieng & Wahid, 2014).

Algoritma *Pathfinding* mengambil inspirasi dari algoritma A* untuk bisa mencari jalur optimal antara simpul-simpul dalam graf berbobot. Algoritma ini menggabungkan metode pencarian heuristik dengan biaya aktual untuk menentukan jalur terbaik dari simpul awal ke simpul tujuan. Dimulai dari titik awal, algoritma mengeksplorasi tetangga-tetangganya dan memilih yang memiliki biaya paling rendah, yang mencakup kombinasi jarak Euclidean sebagai heuristik dan biaya aktual sejauh ini. Secara iteratif, jalur dibangun dengan memperhatikan kapasitas maksimum transporter, dan jika kapasitas terlampaui, algoritma menambahkan titik

d untuk mengosongkan kembali beban yang diangkut. Proses ini berlangsung hingga seluruh *node* dikunjungi, dan hasilnya adalah urutan barang yang diambil untuk proses *order picking.*

Perbedaan utama dari algoritma ini dengan algoritma A* standar adalah jika dalam A* standar, tidak perlu melalui setiap titik untuk mencapai tujuan. Sebaliknya, pada algoritma *Pathfinding* diperlukan untuk melewati setiap titik agar bisa mencapai tujuan. Karakteristiknya ini cocok diterapkan dalam mencari rute *order picking*, dimana semua barang harus diambil terlebih dahulu agar proses *order picking* dapat diselesaikan.

Berikut adalah *pseudocode* dari algoritma *Pathfinding* :

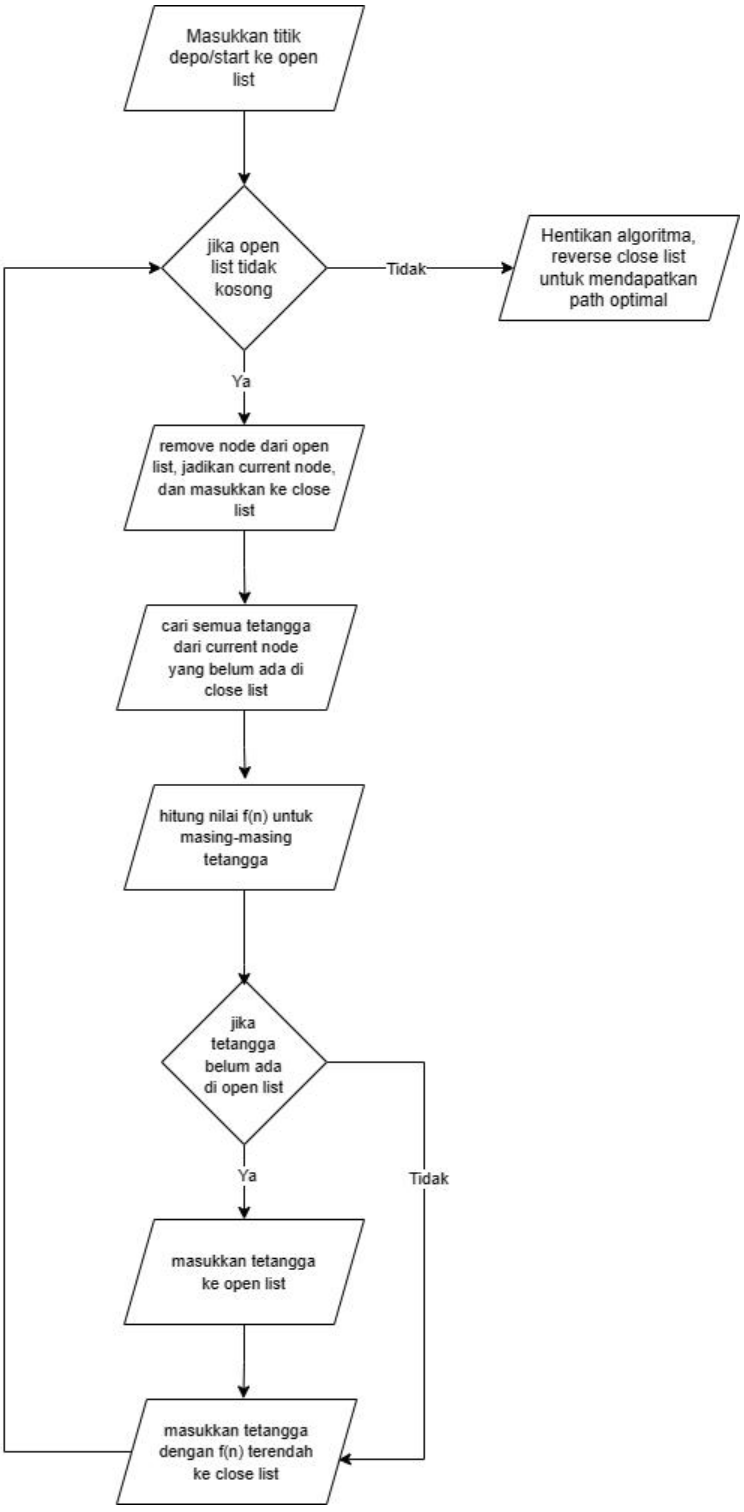
Algorithm 1 Pathfinding Algorithm

```
1:  procedure Pathfinding
2:  open_list contains nodes that have not been traversed
3:  close_list contains nodes that have been traversed
4:  path contains the sequence of nodes that form the order picking route
5:  add start node to path
6:  WHILE open_list is not empty and length of close_list is not equal to length of
    open_list
7:      IF start node is in open_list THEN
8:          remove start node from open_list
9:      END IF
10:     FOR each neighbor in nodes
11:         IF neighbor is not in close_list THEN
12:             calculate f(n) value
13:         END IF
14:         IF f(n) is less than best_f(n) THEN
15:             update best_distance and best_cost
16:             set best_neighbor to neighbor
17:         END IF
18:     END FOR
19:     add best_neighbor to path
20:     update current_capacity
21:     WHILE current_capacity is greater than max_capacity
22:         IF current_capacity is greater than max_capacity THEN
23:             add best_neighbor to path
24:             update current_capacity
25:         END IF
26:     END WHILE
27:     update current node to best_neighbor
28:     add current node to close_list
29: END WHILE
30: IF last node in path is not finish node THEN
31:     add finish node to path
```

32: **END IF**

33: **RETURN** path

Berikut adalah diagram alur algoritma *Pathfinding* :

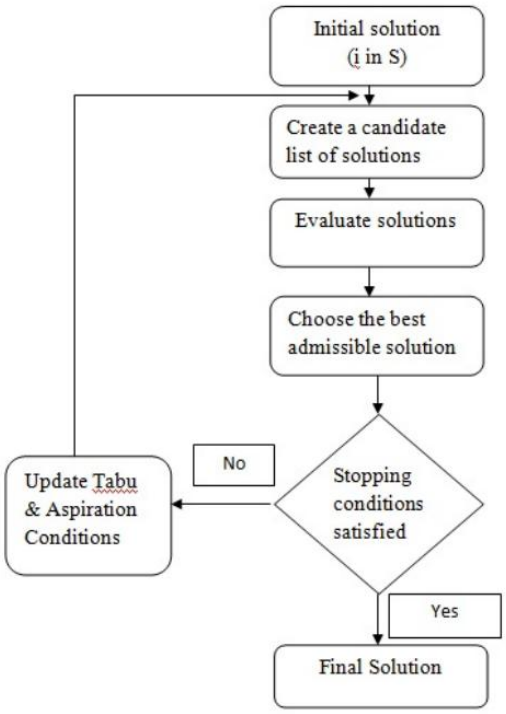


Gambar 2.1 Flowchart Algoritma *Pathfinding*

2.1.3. Algoritma *Tabu Search*

Tabu Search adalah teknik optimisasi berdasarkan *local search*, dimana proses pencarian berpindah dari satu solusi ke solusi berikutnya dengan memilih solusi terbaik yang tidak tergolong sebagai solusi yang dilarang (tabu) (Ariantini, M.S. & Dirgayusari, A.M., 2021). Pada algoritma ini, semua kemungkinan solusi akan dicoba dan ketika menemukan hasil yang lebih baik dari sebelumnya, maka solusi ini akan dimasukkan ke dalam *Tabu List*, dimana isi dari *Tabu List* tidak diubah lagi. Hal inilah yang membuat algoritma *Tabu Search* menjadi lebih efisien dalam segi usaha dan waktu yang dibutuhkan untuk menyelesaikan permasalahan (Suryanto, 2010). Dalam konteks skripsi ini, ukuran *Tabu List* adalah sebesar sepertiga dari jumlah *node* yang ada.

Berikut flowchart dari alur algoritma *Tabu Search* :



Gambar 2.2 Flowchart Algoritma *Tabu Search*

Sumber : Prajapati, et. al., 2020

2.1.4. Algoritma A*

Algoritma A* adalah algoritma pencarian yang digunakan untuk menemukan jalur terpendek antara titik awal dan akhir (Trivusi, 2023). Keunggulan algoritma A* dalam mencari jalur terpendek menjadikannya pilihan yang populer dalam pengembangan sistem yang membutuhkan perencanaan lintasan atau navigasi yang cerdas. Dalam penelitian yang

membandingkan algoritma A* dan Dijkstra, disimpulkan bahwa A* mempunyai waktu komputasi yang lebih singkat (Candra, et.al., 2020).

Algoritma A* menggabungkan pendekatan heuristik dan pencarian graf untuk mempercepat proses pencarian. Algoritma ini menghitung perkiraan jarak dari current *node* ke goal *node* dengan menggunakan persamaan berikut (Yan, 2023):

$$f(n) = g(n) + h(n) \quad (2.2)$$

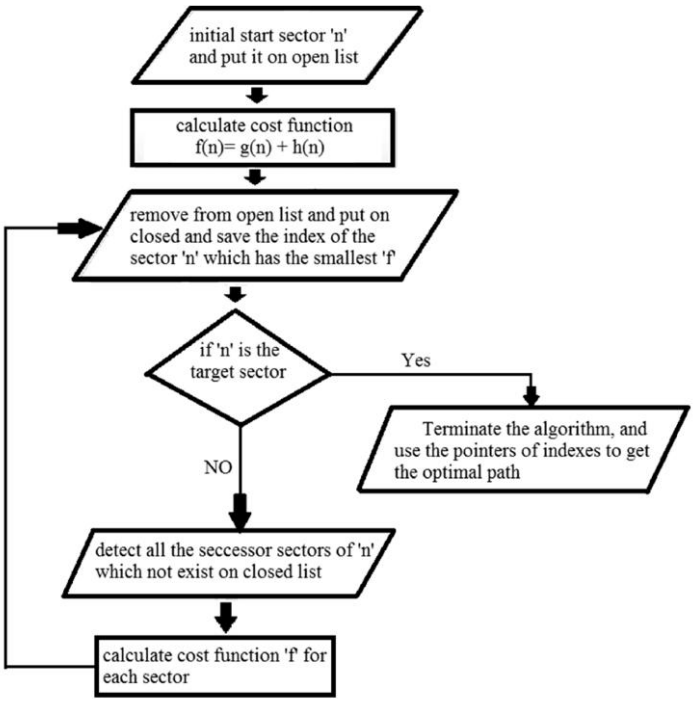
Keterangan :

- f(n) merupakan jarak dari *node n* menuju *node* akhir
- g(n) merupakan jarak/biaya sesungguhnya dari *node* awal menuju *node n*
- h(n) merupakan perkiraan jarak dari *node n* menuju *node* akhir

Pada penelitian ini, teknik heuristik yang digunakan untuk menghitung nilai h(n) adalah Euclidean Distance. Berikut adalah persamaan untuk menghitung nilai dari Euclidean Distance :

$$\mathrm{D} = \sqrt{(\mathrm{x}_2 - \mathrm{x}_1)^2 + (\mathrm{y}_2 - \mathrm{y}_1)^2} \quad (2.3)$$

Berikut adalah diagram alur algoritma A* :



Gambar 2.3 Flowchart Alur Algoritma A*

Sumber : Zidane, I. M. & Ibrahim, K.A., 2018

2.2. Tinjauan Studi

2.2.1. *Membuat Rute Order Picking di Warehouse Cawang PT Global Digital Niaga dengan Menggunakan Metode Routing Heuristics* (Ariffien & Putra, 2020)

- Masalah yang diangkat oleh penelitian ini adalah belum adanya penerapan metode untuk menentukan rute *order picking* pada gudang PT Global Digital Niaga. Tujuan dari penelitian ini adalah menghasilkan sistem yang mampu membantu dalam menghasilkan rute *order picking* bagi perusahaan tersebut.
- Metode yang diusulkan dalam penelitian ini adalah *Routing Heuristics,* yang terdiri dari *S-Shape, Return Back, dan Midpoint.* Penulis menggunakan masing-masing sub-metode tersebut untuk menentukan mana yang paling efisien dalam melakukan proses order picking.
- Hasil penelitian tersebut cukup baik. Penulis menyimpulkan bahwa sub-metode *Midpoint* merupakan metode yang paling efisien. Sistem yang dibuat juga mampu membantu proses order picking pada *warehouse* milik PT Global Digital Niaga.
- Perbedaan penelitian tersebut dengan yang dilakukan pada skripsi ini terletak pada algoritma yang digunakan. Selain itu, penelitian tersebut juga tidak memperhitungkan berat benda dan kapasitas maksimal pengangkut.

2.2.2. *Tabu Search for the Vehicle Routing Problem with Multiple Trips, Time Windows, and Simultaneous Delivery-Pickup* (Suprayogi & Priyandari, 2017)

- Masalah yang diangkat pada penelitian ini adalah *Vehicle Routing Problem with Multiple Trips, Time Windows, and Simultaneous Delivery-Pickup* (VRPMTTWSDP). Pada masalah ini, terdapat titik depo dan sejumlah konsumen dimana setiap konsumen mempunyai permintaan penjemputan dan pengantaran.
- Metode yang diusulkan dalam penelitian ini adalah menggunakan *Tabu Search* dimana solusi awalnya dibuat dengan menggunakan metode algoritma *Sequential Insertion* (SI).
- Hasil penelitian tersebut baik. Penulis menyimpulkan bahwa metode *Tabu Search* yang digunakan mampu memberikan hasil yang lebih baik dibandingkan algoritma *Local Search* dan *Genetic Algorithm.*
- Perbedaan antara penelitian ini dengan skripsi sebelumnya terletak pada penggunaan algoritma untuk memberikan solusi awal *Tabu Search.* Dalam skripsi ini, solusi awal *Tabu Search* diberikan menggunakan algoritma *Pathfinding.*

2.2.3. *Sistem Otomasi Rute Order Picking pada Gudang dengan Metode Simulated Annealing* (Cahyady et al., 2022)

- Masalah yang diangkat pada penelitian ini adalah proses *order picking* merupakan proses yang paling mahal dalam sistem pergudangan, dan perlu ditingkatkan keefisiensiannya. Tujuan dari penelitian ini adalah membuat sistem otomasi terkait persediaan barang dan menentukan rute *order picking* agar keseluruhan proses bisa menjadi lebih efisien.
- Metode yang diusulkan dalam penelitian ini adalah menggunakan RFID dan algoritma *Simulated Annealing*. *User* bisa mengetahui di rak mana barang disimpan serta rute mana yang paling efisien untuk mencapai rak tersebut.
- Hasil penelitian tersebut baik. Penulis menyimpulkan bahwa sistem yang dibuat bisa mendeteksi lokasi benda serta menentukan rute *order picking* yang paling efisien.
- Perbedaan penelitian tersebut dengan yang dilakukan pada skripsi ini adalah pada algoritma yang digunakan. Algoritma *Simulated Annealing* digunakan untuk menghasilkan urutan barang yang diambil, sementara rute antar barang pada denah gudang didapat dari inputan manual user. Pada penelitian ini, algoritma A* digunakan untuk menghasilkan rute antar barang yang lebih optimal.

Tabel 2.2

Perbandingan Penelitian yang Terkait dengan Penelitian Ini

Penulis	**Ariffien & Putra (2020)**	**Suprayogi & Priyandari (2017)**	**Cahyady et al., (2022)**	**Tjandra et al., (2023)**
Judul	Membuat Rute *Order Picking* di *Warehouse* Cawang PT Global Digital Niaga dengan Menggunakan Metode *Routing Heuristics*	*Tabu Search for the Vehicle Routing Problem with Multiple Trips, Time Windows, and Simultaneous Delivery-Pickup*	Sistem Otomasi Rute *Order Picking* pada Gudang dengan Metode *Simulated Annealing*	Penjadwalan Rute *Inventory Order Picking* dengan Menggunakan Kombinasi Algoritma *Pathfinding* dan *Tabu Search*

Masalah Penelitian	Belum adanya penerapan metode untuk menentukan rute *order picking* pada gudang PT Global Digital Niaga.	*Vehicle Routing Problem with Multiple Trips, Time Windows, and Simultaneous Delivery-Pickup* (VRPMTTWSDP)	Proses *order picking* merupakan proses yang paling mahal dalam sistem pergudangan, dan perlu ditingkatkan keefisiensiannya .	Proses *order picking* merupakan proses yang paling mahal dalam sistem pergudangan, dan perlu ditingkatkan keefisiensiannya
Metode	Routing Heuristics yang terdiri dari *S-Shaped, Midpoint,* dan *Return Back*	Tabu Search, Sequential Insertion	RFID, *Simulated Annealing*	*Tabu Search, Pathfinding,* dan A*
Hasil	Sistem *Order Picking*	Program yang mengimplementasikan *Tabu Search*	Sistem otomasi rute *order picking*	Aplikasi penjadwalan rute *order picking*
Perbedaan dengan penelitian pada skripsi ini	- Menggunakan metode *Routing Heuristics* - Tidak memperhitungkan berat benda dan kapasitas maksimal pengangkut	- Penggunaan algoritma *Sequential Insertion* untuk memberi solusi awal *Tabu Search*	- Menggunakan algoritma *Simulated Annealing* untuk menentukan urutan *order picking* - Rute antar barang didapat dari inputan	- Menggunakan metode *Tabu Search* dan Pathfinding untuk menentukan urutan *order picking* - Rute antar barang didapat

			manual *user*	dengan menggunakan algoritma A*