BAB 2 LANDASAN TEORI

Bab ini akan menjelaskan tentang landasan teori yang digunakan pada penelitian tugas akhir ini. Pemaparan ini bertujuan untuk memberikan pemahaman lebih mendalam kepada penulis serta pembaca laporan, mengenai teori-teori yang melandasi proses penyelesaian masalah pada penelitian tugas akhir ini.

2.1 Teknologi Jaringan Komunikasi

Perangkat teknologi komunikasi adalah segala perlengkapan dan fasilitas yang dapat digunakan oleh manusia dalam rangka melakukan proses perhubungan atau komunikasi jarak jauh. Teknologi ini sangat erat hubungannya dengan teknologi informasi karena salah satu hal yang paling sering dibahas pada komunikasi adalah bagaimana cara memindahkan informasi dari suatu tempat ke tempat lain yang saling berjauhan. Supaya dapat menghubungkan berbagai lokasi pada jarak berjauhan ini, seringkali dibutuhkan media yang mampu memindahkan informasi secara cepat dan handal. Kebutuhan akan media yang sesuai inilah yang kemudian menentukan tren perkembangan teknologi di bidang komunikasi.

Secara garis besar, tren perkembangan teknologi komunikasi terbagi atas dua kelompok besar berdasarkan media yang digunakannya sebagai sarana pemindahan informasi, yaitu perangkat teknologi berkabel atau dikenal dengan istilah *wired communication*, serta perangkat teknologi tanpa kabel atau dikenal juga dengan istilah *wireless communication*.

Perangkat teknologi berkabel, sesuai dengan namanya, menggunakan media kabel sebagai sarana utama untuk memindahkan informasi dari suatu tempat ke tempat yang lain. Media ini dapat menjangkau tempat yang berjauhan dengan mudah karena bentuk fisiknya yang tipis dan panjang. Dengan memanfaatkan sifat konduktor komponen kabel yang digunakan, informasi yang dibawa dalam bentuk sinyal elektromagnetik dapat dipindahkan dalam waktu yang cepat. jenis teknologi ini pun populer digunakan selain karena juga handal dan mudah untuk digunakan karena biaya kabel yang tidak terlalu mahal.

Perangkat teknologi nirkabel merupakan jenis teknologi komunikasi lainnya yang tidak menggunakan media kabel sebagai sarana pemindahan informasi. Pada teknologi berkabel prinsip yang digunakan adalah konduksi pada benda padat, sedangkan pada teknologi jenis ini informasi dipindahkan melalui radiasi gelombang elektromagnetik melalui perangkat-perangkat yang dapat menangkap frekuensi yang bersesuaian. Perangkat teknologi komunikasi nirkabel sebenarnya sudah lama dikenal sejak radio pertama kali diciptakan pada abad 19. Dengan tidak menggunakan media kabel sebagai sarana untuk memindahkan informasi dari suatu tempat ke tempat lain, teknologi ini menawarkan peluang untuk digunakan pada teknologi komunikasi secara luas. Dewasa ini, teknologi jaringan nirkabel banyak digunakan pada perangkat-perangkat komunikasi lain, seperti telepon selular, jaringan internet, GPS, PDA, satelit, televisi, serta perangkat teknologi lain yang tidak memungkinkan penggunaan kabel sebagai mediator utama untuk memindahkan informasi.

2.2 Robotika dan Kecerdasan Buatan

Teknologi robotika adalah teknologi yang membahas tentang proses desain, perancangan serta pembuatan robot, hingga aplikasi-aplikasi yang di dalamnya melibatkan penggunaan robot. Berbicara mengenai robot, sangat erat kaitannya dengan konsep agen cerdas dan sistem kecerdasan buatan.

Robot sendiri sebenarnya tidak terbatas pada perangkat keras cerdas yang dapat melaksanakan suatu tujuan tertentu, tetapi juga termasuk perangkat lunak yang dapat digunakan untuk menyelesaikan permasalahan yang diberikan secara cerdas. Walaupun demikian, istilah robot pada saat sekarang sering dikaitkan dengan perangkat elektrik-mekanik hasil rekayasa manusia yang memiliki bentuk fisik serta memiliki sifat cerdas, dengan berbagai karakteristik seperti dapat bergerak dan berpindah tempat, dapat berkomunikasi dengan lingkungan, serta mengubah keadaan lingkungannya melalui proses interaksi yang dilakukannya tersebut, atau terkadang dapat meniru perilaku cerdas hewan dan manusia.

Berikut akan dijelaskan lebih lanjut tentang kecerdasan buatan, agen cerdas, serta teknologi lain yang berkaitan dengannya.

2.2.1 Intelligent Agent

Berikut adalah beberapa pendapat yang mengatakan kecerdasan buatan itu adalah:

- Usaha-usaha yang dilakukan dalam rangka membuat komputer atau mesin berpikir dan mempunyai pikiran [Haugeland, 1985].
- Pembelajaran mengenai kecakapan yang bersifat kejiwaan melalui penggunaan berbagai model komputasi [Charniak & McDermott, 1985].
- Seni membangun mesin yang dapat melaksanakan beberapa fungsi yang jika dilakukan oleh manusia akan membutuhkan tingkat intelejensi [Kurzweil, 1990].
- Merupakan pembelajaran mengenai perilaku cerdas dalam benda-benda hasil rekayasa manusia [Nilsson, 1998].

Pada dasarnya, konsep kecerdasan itu sendiri mengacu kepada perilaku berpikir dan bertindak seperti manusia atau secara rasional. Sampai sekarang belum ada teori absolut yang dapat menjelaskan cara manusia berpikir sehingga penjelasan yang paling tepat untuk mendeskripsikan kecerdasan buatan adalah sistem hasil rekayasa manusia yang mampu bertindak secara rasional. Agen yang menjadi aktor dalam sistem ini disebut sebagai agen cerdas atau agen rasional.

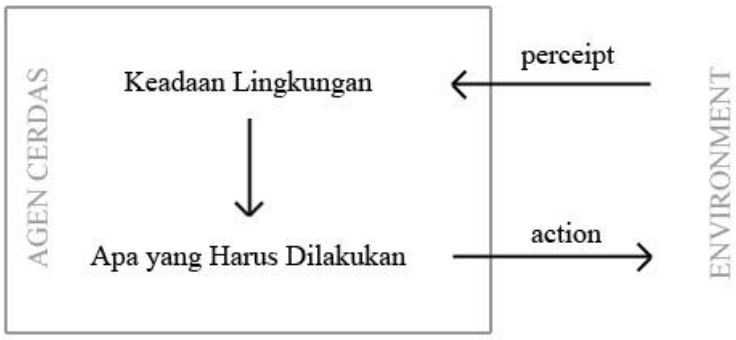
Dalam melaksanakan tugasnya, agen cerdas selalu berupaya melakukan pekerjaannya untuk mendapatkan hasil yang paling baik. Untuk itu, agen cerdas perlu mengetahui beberapa parameter tertentu untuk dapat menyelesaikan tugasnya. Russell & Norvig mendefinisikan parameter ini dalam konsep P.A.G.E, yaitu *Percept*, *Actions*, *Goal*, dan *Environment*, yang dijabarkan sebagai berikut.

- *Percept*, merupakan cara bagi suatu agen cerdas untuk dapat menerima masukan dari lingkungannya. Contoh *percept* adalah sensor cahaya, sensor suara, kamera, GPS, dan sebagainya.

- *Actions*, merupakan semua tindakan yang dapat dilakukan oleh agen cerdas sebagai bentuk respon terhadap kondisi lingkungan yang ditemuinya, dalam rangka menyelesaikan tugas tertentu yang diberikan. Contoh *actions* adalah suatu agen cerdas akan bergerak mundur dan berputar jika ditemui halangan di depan jalannya.
- *Goal*, merupakan tugas atau misi yang harus diselesaikan oleh suatu agen cerdas, berikut parameter yang menentukan keberhasilan pelaksanaan tugas yang diberikan tersebut. Contoh *goal* dari agen mobil otomatis adalah untuk mengantarkan penumpang sampai suatu tujuan tertentu.
- *Environment*, merupakan lingkungan tempat agen cerdas beroperasi, dimana agen tersebut dapat berkomunikasi dan berinteraksi bahkan mengubah kondisi lingkungan tersebut sesuai tindakan yang dilakukannya.

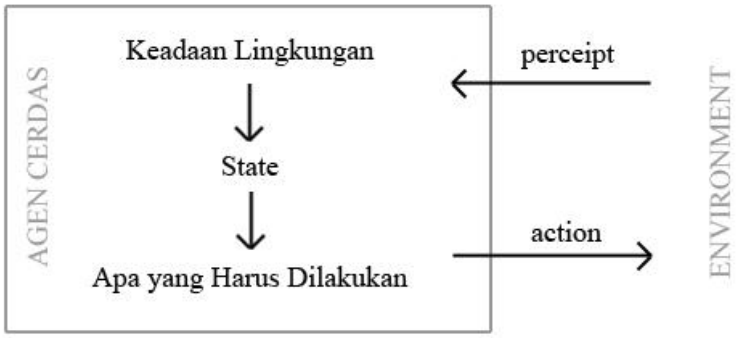
Berdasarkan karakteristik dalam mengerjakan tugasnya, agen cerdas atau agen rasional kemudian dapat dikelompokkan dalam beberapa jenis sebagai berikut.

1. *Simple Reflex Agent*, yaitu agen cerdas yang tindakannya ditentukan oleh respon terhadap masukan terakhir yang diberikan. Gambar 2.1 di bawah ini merupakan cara kerja dari agen jenis ini.



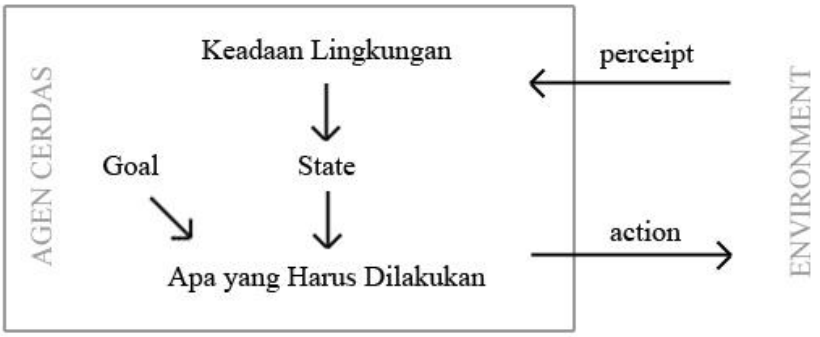
Gambar 2.1 - Simple Reflex Agent

2. *Model Based Agent*, yaitu agen cerdas yang memiliki representasi internal mengenai keadaan pada lingkungannya. Gambar 2.2 di bawah ini merupakan cara kerja dari agen jenis ini.



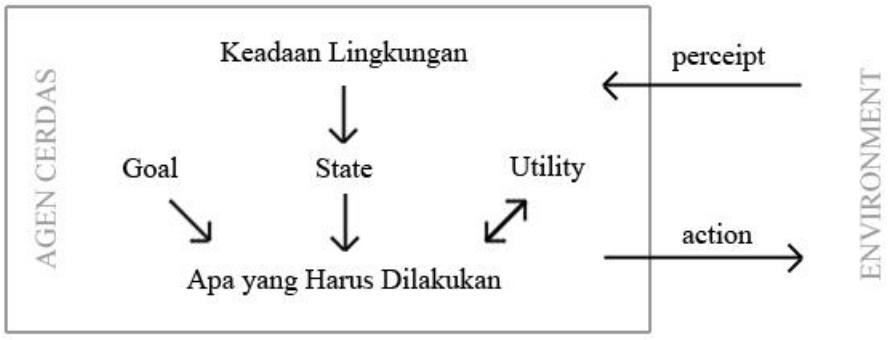
Gambar 2.2 - Model Based Agent

3. *Goal Based Agent*, yaitu agen cerdas yang memiliki tujuan serta dapat memilih tindakan yang dapat dilakukannya dalam rangka mencapai tujuan tersebut. Gambar 2.3 di bawah ini merupakan cara kerja dari agen jenis ini.



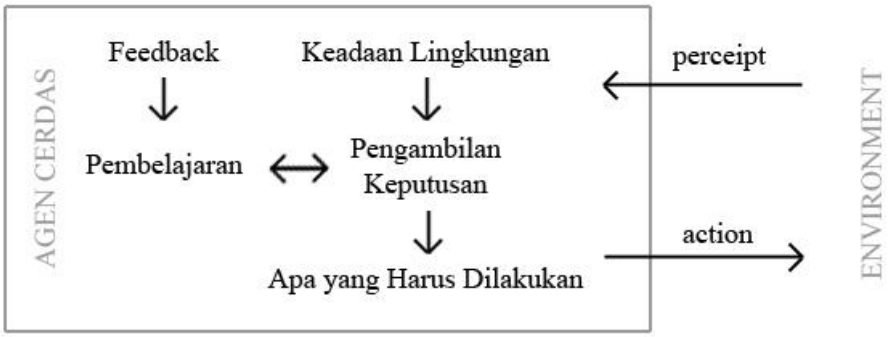
Gambar 2.3 - Goal Based Agent

4. *Utility Based Agent*, yaitu agen cerdas yang mempunyai penilaian kualitatif terhadap hasil yang ditemukannya pada suatu langkah tertentu, kemudian penilaian ini dimanfaatkan untuk menentukan tindakan yang diambil pada langkah berikutnya. Gambar 2.4 di bawah ini merupakan cara kerja dari agen jenis ini.



Gambar 2.4 - Utility Based Agent

5. *Learning Agent*, yaitu agen cerdas yang dapat belajar dalam proses penyelesaian tugasnya. Selain itu, agen ini juga menggunakan sistem feedback dalam rangka meningkatkan performa kerjanya. Gambar 2.5 di bawah ini merupakan cara kerja dari agen jenis ini.



Gambar 2.5 - Learning Agent

2.2.2 Autonomous Robot

Autonomous robot atau robot otonom merupakan salah satu jenis robot yang dapat melakukan proses pengambilan keputusan sendiri tanpa adanya campur tangan dari pihak luar, baik dari manusia maupun komponen lain seperti komputer, *controller*, dan sebagainya. Oleh karena itu, setiap perintah dan komputasi yang dilakukan oleh robot otonom dikompilasi dan dijalankan secara langsung pada mesin robot yang bersangkutan.

Terdapat beberapa jenis robot otonom, seperti *fully autonomous*, yaitu robot yang bersifat otonom penuh serta *semi autonomous*, yaitu robot yang tidak bersifat otonom penuh karena masih membutuhkan pengaruh dari perangkat luar dalam rangka melaksanakan tugas yang dilakukannya. Adapun kedua jenis robot ini dapat digunakan secara efektif untuk memudahkan pekerjaan manusia, sesuai dengan kebutuhannya. Pada sistem yang bersifat *decentralized* robot *fully autonomous* cocok digunakan karena sifat tidak tergantung kepada perangkat lain dan dapat menyelesaikan pekerjaan sendiri.

2.2.3 Swarm Robotics

Swarm Robotics merupakan salah satu tren pada perkembangan teknologi robotika yang memanfaatkan robot dalam suatu koloni untuk menyelesaikan suatu tujuan tertentu. Penggunaan banyak robot dalam kelompok ini diharapkan untuk memudahkan pekerjaan, di mana semua robot saling bahu membahu untuk mencapai tujuan bersama. Adapun prinsip yang digunakan pada *swarm robotics* adalah bahwa suatu pekerjaan yang besar dan kompleks dapat dipecah menjadi beberapa pekerjaan yang lebih kecil dan sederhana. Dengan demikian, setiap robot dalam koloni ini diharapkan dapat menyelesaikan tugas masing-masing yang secara bersama-sama akan mencapai tujuan yang lebih besar tersebut. Kegiatan ini disebut juga sebagai *task division* pada koloni robot.

Salah satu permasalahan yang sering dibahas pada teknologi *swarm robotics* adalah mengenai faktor komunikasi dan koordinasi masing-masing robot dalam koloni. Hal ini dilakukan karena penerapan *task division* yang membagi suatu pekerjaan menjadi bagian-bagian yang lebih kecil dalam suatu kelompok menuntut suatu proses integrasi pada tahap akhirnya. Dengan demikian, solusi yang ditemukan oleh masing-masing pihak nantinya dapat merepresentasikan solusi permasalahan besar yang sedang dicari. Hal ini tentu mustahil dilakukan jika robot bekerja sendiri-sendiri tanpa adanya komunikasi.

2.2.4 Robotic Network

Teknologi jaringan robot merupakan salah satu aplikasi yang menggunakan robot untuk membangun suatu jaringan perhubungan. Hal ini mungkin dilakukan karena dewasa ini robot sering dikembangkan dengan menggunakan fungsi komunikasi yang memungkinkan terjadinya perpindahan data dan informasi dari suatu robot kepada robot lain. Dengan mengoperasikan robot secara massal dalam suatu kelompok, dapat dibangun suatu jaringan yang menggunakan robot sebagai perangkat yang saling berkomunikasi sebagai sarana utama untuk memindahkan informasi di dalam jaringan tersebut.

Dengan memanfaatkan jaringan robot seperti ini dapat diperoleh berbagai keunggulan, misalnya kemampuan robot untuk menyelesaikan pekerjaan dengan cerdas serta penggunaan robot untuk menjelajahi daerah-daerah yang sulit dijangkau menggunakan perangkat biasa.

2.3 Graph dan Tree

Graph merupakan suatu struktur diskrit yang terdiri atas sekumpulan vertex dengan edge yang saling menghubungkan antara satu vertex dengan vertex lain. Pada umumnya, graph dapat digunakan untuk memodelkan permasalahan yang menyangkut keterhubungan antara suatu entitas dengan entitas lain dalam suatu himpunan. Setiap entitas di dalam himpunan direpresentasikan oleh suatu vertex. Jika terdapat suatu hubungan tertentu antara dua atau lebih entitas dalam himpunan, maka digambarkan vertex-vertex tersebut terhubung dalam edge-edge dengan weight yang merepresentasikan nilai keterhubungan mereka tersebut. Jika semua vertex saling terhubung di dalam suatu graph yang sama, maka graph ini disebut sebagai *connected graph*. Suatu vertex dapat terhubung secara langsung maupun tidak langsung dengan vertex-vertex yang lain.

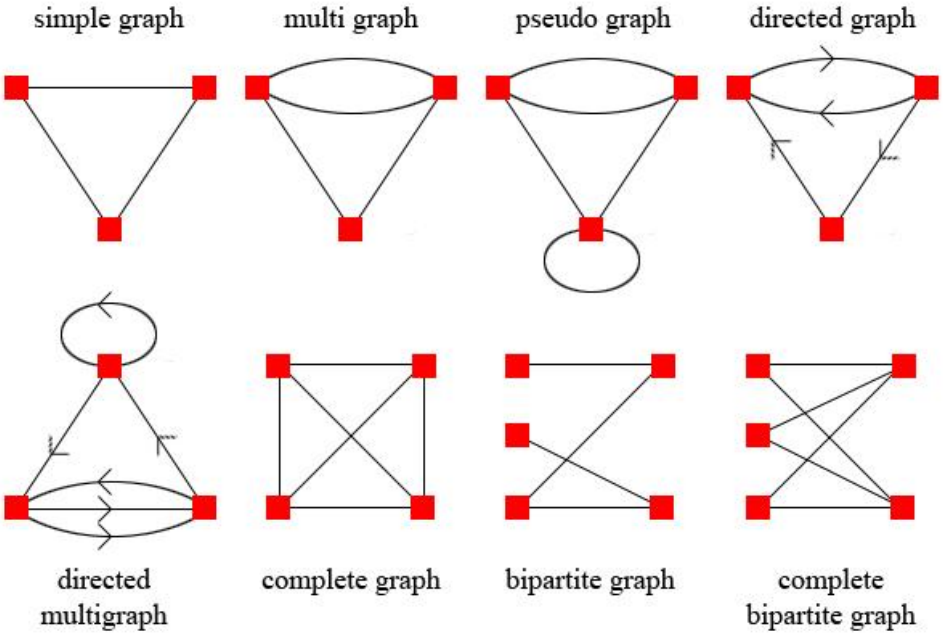
Berikut ini adalah beberapa jenis dari graph yang dapat digunakan untuk memodelkan berbagai permasalahan dalam kehidupan nyata. Solusi permasalahan kemudian dapat ditemukan dengan menyelesaikan permasalahan graph tersebut.

- *Simple graph*, merupakan graph di mana tidak terdapat dua edge yang berbeda sebagai penghubung pasangan dua vertex yang sama atau suatu pasangan vertex hanya dihubungkan oleh satu edge tertentu saja.
- *Multi graph*, merupakan graph di mana terdapat dua edge yang berbeda yang menghubungkan pasangan dua vertex yang sama atau suatu pasangan vertex bisa saja dihubungkan oleh dua edge yang berbeda.
- *Pseudo graph*, merupakan graph di mana mungkin terdapat edge yang menghubungkan suatu vertex dengan dirinya sendiri. *Pseudo graph* sering disebut sebagai *loop*.
- *Directed graph*, merupakan graph di mana setiap edge yang diberikan memiliki arah keterhubungan dari suatu vertex tertentu menuju vertex yang lainnya, tetapi belum tentu memiliki hubungan yang sebaliknya.
- *Directed multigraph*, merupakan graph di mana selain edge yang diberikan memiliki arah keterhubungan, serta mungkin terdapatnya dua edge berbeda dengan arah yang sama menghubungkan dua pasangan vertex yang sama.

Selain itu, terdapat beberapa jenis graph spesial dengan karakteristik tertentu yang membedakannya dari graph lainnya. Adapun graph-graph dengan karakteristik khusus ini antara lain sebagai berikut.

- *Complete graph*, merupakan graph di mana setiap vertex terhubung secara langsung melalui suatu edge tertentu dengan semua vertex lain.
- *Bipartite graph*, merupakan graph di mana semua vertex dapat dibagi menjadi dua kelompok tertentu dengan setiap vertex dari suatu kelompok hanya boleh terhubung dengan vertex lain pada kelompok yang berbeda.
- *Complete bipartite graph*, merupakan graph yang memenuhi sifat *complete graph* dan *bipartite graph* secara bersama-sama.

Untuk lebih jelasnya, pada gambar 2.6 berikut di bawah akan ditampilkan representasi grafis dari jenis-jenis graph yang telah disebutkan di atas secara berurutan dimulai dari kiri atas hingga kanan bawah.

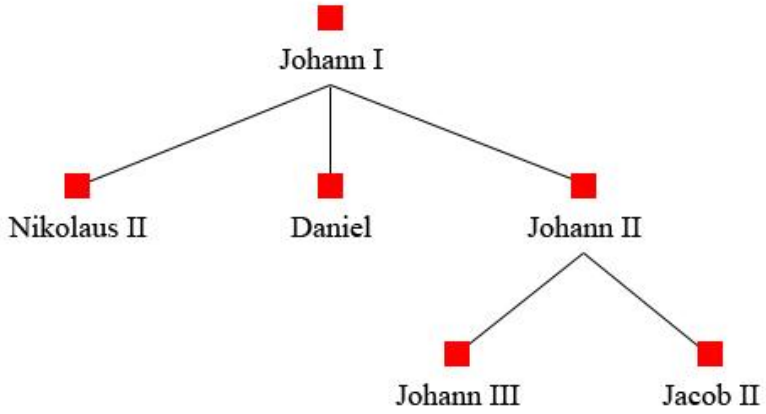


Gambar 2.6 - Jenis-Jenis Graph

Selain jenis-jenis graph yang disebutkan di atas, terdapat satu jenis graph khusus yang sering digunakan pada pemodelan masalah dalam kehidupan sehari-hari. Graph khusus tersebut adalah tree, yaitu graph terhubung yang tidak memiliki *loop* yang menghubungkan suatu vertex dengan vertex-vertex lain yang nantinya akan terhubung dengan vertex tersebut kembali. Loop ini disebut *circuit* sehingga tree dapat juga didefinisikan sebagai graph yang tidak memiliki circuit.

Tree sering digunakan sebagai pemodelan permasalahan keterhubungan entitas-entitas dalam suatu himpunan yang bersifat hirarkikal atau bertingkat. Permasalahan yang dapat dimodelkan melalui tree misalnya silsilah hubungan keluarga, kemungkinan langkah yang dapat diambil oleh pemain catur pada setiap giliran, hubungan atom-atom dalam struktur kimia suatu zat tertentu, ataupun banyak permasalahan lainnya.

Gambar 2.7 berikut di bawah ini merupakan salah satu contoh representasi tree yang dapat digunakan untuk memodelkan silsilah keluarga Bernoulli.



Gambar 2.7 - Contoh Aplikasi Tree

Vertex pada tree dikenal juga dengan istilah node, di mana node dengan hirarki tertinggi disebut sebagai root. Node dengan hirarki terendah disebut sebagai leaf yang merupakan node terluar dari konfigurasi tree tersebut karena node ini tidak terhubung lagi dengan node lain yang hirarkinya lebih rendah. Selain itu, jarak antara leaf terjauh dari root disebut sebagai kedalaman tree, di mana node-node yang berjarak sama dari root dikatakan berada pada level yang sama.

Berikut akan dijelaskan beberapa permasalahan umum yang sering dimodelkan dengan menggunakan graph dan tree. Ketika dijumpai permasalahan pada kehidupan nyata yang dapat dimodelkan dengan pemodelan yang serupa, solusi terhadap masalah tersebut akan dapat ditemukan karena sudah banyak algoritma yang dikembangkan untuk berbagai permasalahan ini.

2.3.1 Shortest Path Problem

Shortest Path Problem merupakan permasalahan di mana dengan diberikan suatu konfigurasi graph terhubung yang terdiri atas sekumpulan vertex dan edge dengan weight tertentu, kemudian diminta untuk menemukan jalur atau rute terpendek yang dapat ditempuh dari suatu vertex kepada vertex-vertex lainnya.

Salah satu algoritma yang dapat menyelesaikan permasalahan ini dengan baik adalah Algoritma Djikstra yang pertama kali diperkenalkan oleh seorang ahli matematika dari Belanda, Edsger Djikstra pada tahun 1959. Prinsip yang

digunakan pada algoritma ini adalah menentukan jarak terpendek dari suatu vertex dimulai dari vertex-vertex yang terdekat dari vertex awal tersebut, hingga jarak ke semua vertex berhasil ditemukan. Berikut adalah langkah penyelesaian masalah pada algoritma ini.

1. Tentukan suatu vertex yang akan menjadi posisi awal, dari mana akan dicari jalur atau rute terpendek terhadap semua vertex yang lain.
2. Tentukan jarak dari vertex awal tersebut ke vertex-vertex yang terhubung dengannya. Vertex yang tidak terhubung dengan vertex awal diasumsikan berjarak tidak berhingga.
3. Ambil salah satu vertex dengan jarak terpendek terhadap vertex awal, dari semua kemungkinan vertex yang terhubung dengan vertex awal. Vertex yang sudah dikunjungi ini dimasukkan ke dalam suatu himpunan yang disebut sebagai himpunan penyelesaian sementara. Simpan jarak yang menghubungkan vertex dengan vertex awal sebagai nilai jarak.
4. Ulangi kembali dari langkah 2, kali ini jarak ditentukan dari himpunan penyelesaian sementara terhadap vertex-vertex yang belum dikunjungi. Lakukan hal ini secara berulang-ulang, hingga semua vertex selesai dikunjungi dan masuk ke dalam himpunan penyelesaian sementara. Himpunan ini kemudian dapat disebut sebagai himpunan penyelesaian.
5. *Shortest Path* dari vertex awal terhadap vertex-vertex yang lain merupakan nilai weight setiap edge yang disimpan pada saat suatu vertex tertentu dimasukkan ke dalam himpunan penyelesaian. Jika vertex awal tidak terhubung secara langsung dengan vertex tersebut, maka nilai jarak terpendek merupakan hasil akumulasi weight dari setiap edge yang harus dilalui dari vertex awal hingga vertex tujuan.

2.3.2 Minimum Spanning Tree

Minimum Spanning Tree merupakan permasalahan dimana dengan diberikan suatu konfigurasi graph terhubung yang terdiri atas sekumpulan vertex dan edge

dengan weight tertentu, kemudian diminta untuk melakukan konstruksi tree di mana total weight yang diperoleh nantinya akan berjumlah minimum.

Salah satu algoritma yang dapat menyelesaikan permasalahan ini dengan baik adalah algoritma yang pertama kali diperkenalkan oleh seorang ahli matematika dari Amerika Serikat, Robert Prim pada tahun 1921. Prinsip yang digunakan pada algoritma ini adalah mengambil semua edge dengan nilai weight terkecil selama tidak terjadi *cycle*, hingga semua vertex terhubung di dalamnya. Berikut adalah langkah penyelesaian masalah pada algoritma ini.

1. Tentukan suatu vertex yang akan menjadi posisi awal, dari mana proses konstruksi tree akan dimulai.
2. Tentukan semua edge yang menghubungkan vertex tersebut, kemudian ambil edge yang memiliki nilai weight yang paling kecil. Simpan vertex yang dihubungkan oleh edge yang diambil tersebut.
3. Ulangi dari langkah 1, kali ini dengan menggunakan vertex yang disimpan pada langkah sebelumnya. Lakukan hal ini secara berulang-ulang. Jika selama proses konstruksi tree, edge yang diambil akan menimbulkan suatu *cycle*, maka ambil edge dengan nilai weight terendah berikutnya.
4. Berhenti ketika semua vertex sudah disimpan. Semua edge yang diambil pada kegiatan ini merupakan solusi *spanning tree* yang diharapkan.

Selain itu, terdapat algoritma lain yang dapat digunakan untuk menyelesaikan permasalahan ini, yaitu algoritma yang diperkenalkan oleh Joseph Kruskal pada tahun 1928. Berikut adalah langkah penyelesaian masalah pada algoritma ini.

1. Pilih suatu edge dengan nilai weight yang paling kecil dari konfigurasi graph yang diberikan. Simpan kedua vertex yang dihubungkan oleh edge yang dipilih tersebut.
2. Lanjutkan pemilihan edge secara berulang hingga semua vertex kembali terhubung dalam konfigurasi graph atau tree yang sama. Selama proses pengambilan edge ini, hindari pengambilan edge yang dapat menimbulkan terjadinya *cycle* pada konfigurasi yang baru.

3. Konfigurasi tree yang didapatkan pada saat semua vertex telah terhubung satu sama lain merupakan solusi dari *spanning tree* yang diharapkan dari permasalahan yang diberikan.

2.3.3 Searching Tree

Searching Tree merupakan pemodelan suatu masalah dengan mendefinisikan bentuk solusi yang diharapkan melalui bentuk suatu konfigurasi tree, kemudian dilakukan penelusuran terhadap semua atau sebagian node pada tree tersebut untuk menemukan solusi yang diharapkan. Ada beberapa cara yang dapat digunakan pada penelusuran tree dalam rangka menemukan solusi yang diharapkan yaitu sebagai berikut.

1. *Uninformed search*, yaitu penelusuran tree dengan hanya memanfaatkan informasi yang ada pada tree tersebut tanpa adanya tambahan informasi dari luar yang dapat membantu proses penelusuran menjadi lebih cepat dilakukan. Beberapa algoritma yang termasuk jenis *uninformed search* ini adalah sebagai berikut.

    a. *Bread-First Search* atau BFS, yaitu proses penelusuran tree dimulai dari root, kemudian diteruskan kepada semua node yang terhubung dengan root sebelum dilanjutkan kepada node-node pada level berikutnya. Dengan kata lain, penelusuran node pada tree dilakukan secara berurutan berdasarkan level kedalamannya.

    b. *Depth-First Search* atau DFS, yaitu proses penelusuran tree dimulai dari root, kemudian diteruskan kepada salah satu node yang terhubung dengan root, terus dilanjutkan kepada salah satu node di level berikutnya hingga pada akhirnya akan mencapai suatu leaf tertentu sebelum dilakukan backtrack yang akan melanjutkan penelusuran terhadap node-node yang belum dikunjungi. Dengan kata lain, penelusuran node pada tree dilakukan dengan menelusuri kedalaman tree terlebih dahulu.

c. *Iterative Deepening Search*, yaitu proses penelusuran tree yang serupa dengan algoritma DFS, akan tetapi dilakukan dengan cara membatasi kedalaman tree secara iteratif. Hal ini berguna untuk mengatasi permasalahan yang tidak dapat diselesaikan oleh algoritma DFS pada saat kedalaman tree yang diberikan bernilai tidak terbatas.

2. *Informed search*, yaitu penelusuran tree dengan memanfaatkan informasi dari luar yang dapat membantu proses penelusuran menjadi lebih cepat dilakukan. Informasi tambahan dari luar ini bisa berupa suatu nilai yang dapat dijadikan sebagai suatu *heuristic* tertentu untuk mengetahui kemungkinan node mana saja yang lebih pantas dikunjungi dan mana yang tidak. Penelusuran kemudian dapat dilakukan dengan mengunjungi node yang memiliki nilai *heuristic* yang lebih baik.

2.4 Geometri dan Ruang Euclid

Geometri merupakan bagian dari ilmu matematika yang membahas tentang posisi, kedudukan, dan ukuran suatu benda pada ruang berdimensi tertentu. Adapun beberapa topik yang dibahas pada geometri adalah jarak suatu benda dari benda yang lain, kedudukan benda relatif terhadap suatu posisi, ukuran sudut, sampai dengan persamaan garis yang terkait dengan ilmu kalkulus.

Terdapat dua cara untuk mendefinisikan posisi suatu benda di dalam suatu ruang berdimensi. Cara pertama, yaitu dengan menggunakan suatu posisi tertentu sebagai titik acuan pada ruang berdimensi. Posisi suatu benda dirumuskan sebagai jarak antara benda terhadap posisi acuan. Jarak benda terhadap suatu posisi dapat dijabarkan pada masing-masing dimensi tempat benda berada, sehingga ukuran jarak ini dapat disajikan dalam bentuk urutan angka, sebagai berikut.

$$\mathbf{P(x_1, x_2, x_3, \ldots, x_n), \text{ dengan } x_i = \text{jarak terhadap dimensi i}} \qquad \mathbf{(2.1)}$$

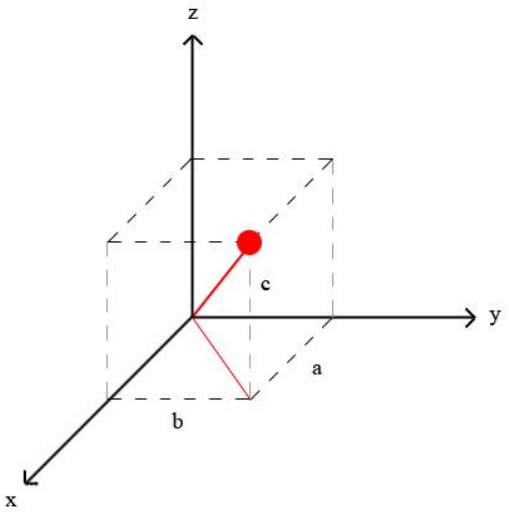
Dengan demikian pada ruang berdimensi-n, posisi benda dapat dinyatakan oleh urutan *n* angka. Dalam keadaan yang sebenarnya, benda-benda berada pada ruang berdimensi 3 sehingga posisi benda di dunia ini dinyatakan sebagai berikut.

P(x, y, z)	**(2.2)**

Untuk dapat merepresentasikan kedudukan benda ini secara benar, kemudian posisi benda digambarkan melalui suatu diagram koordinat, di mana pada tiap-tiap dimensi yang diwakilinya akan terdapat suatu garis acuan yang disebut sebagai garis sumbu. Pada ruang berdimensi 3, digunakan acuan sumbu *x*, sumbu *y*, serta sumbu *z* untuk mendefinisikan dimensi panjang, lebar dan tinggi. Titik pertemuan ketiga sumbu ini kemudian akan menjadi posisi acuan P(0, 0, 0). Konvensi penentuan posisi benda dengan cara yang seperti ini disebut sebagai sistem koordinat absolut.

Cara kedua untuk menentukan posisi benda dalam ruang berdimensi adalah dengan menetapkan posisi benda tersebut sebagai posisi acuan. Dengan kata lain, tiap-tiap benda dipandang memiliki sumbu koordinat masing-masing sehingga posisi suatu benda terhadap benda lain akan bersifat relatif terhadap benda tersebut. Setiap benda memandang dirinya berada pada posisi P(0, 0, 0). Konvensi penentuan posisi benda dengan cara yang seperti ini disebut sebagai sistem koordinat relatif.

Gambar 2.8 berikut di bawah merupakan representasi posisi suatu dalam ruang berdimensi 3.



Gambar 2.8 - Posisi Benda pada Ruang Dimensi 3

Pada gambar tersebut, posisi benda dinyatakan sebagai P(a, b, c).

2.4.1 Jarak Antara Dua Titik

Salah satu hal yang sering dibahas pada ilmu geometri adalah permasalahan jarak antara dua titik. Untuk dapat menentukan jarak antara dua titik pada posisi $P_x(x_1, x_2, \ldots, x_n)$ dan $P_y(y_1, y_2, \ldots, y_n)$ dalam ruang berdimensi n, dapat dilakukan dengan menggunakan rumus sebagai berikut.

$$\textbf{distance} = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2 + \ldots + (y_n - x_n)^2} \qquad \textbf{(2.3)}$$

2.4.2 Persamaan Garis Antara Dua Titik

Dengan memanfaatkan informasi posisi dua titik yang berbeda P_1 dan P_2, dapat ditentukan persamaan garis lurus yang menghubungkan keduanya. Persamaan garis ini kemudian dapat digunakan untuk menentukan setiap posisi yang berada di antara kedua posisi P_1 dan P_2. Seringkali persamaan garis ini merujuk pada ruang berdimensi 2 karena garis lurus merupakan benda berdimensi 2.

Untuk menentukan persamaan garis pada ruang dimensi 2 yang menghubungkan posisi $P_1(x_1, y_1)$ dan $P_2(x_2, y_2)$ dapat dilakukan dengan rumus sebagai berikut.

$$(y - y_1) / (y_2 - y_1) = (x - x_1) / (x_2 - x_1) \quad (2.4)$$

Selain itu, persamaan garis lurus antara dua titik dapat juga diturunkan dengan menggunakan kemiringan garis yang akan dijelaskan pada bagian berikutnya.

2.4.3 Kemiringan Garis

Kemiringan garis ditentukan oleh perbandingan antara posisi garis pada sumbu *y* terhadap posisi garis pada sumbu *x*. Oleh karena itu, untuk dapat menentukan kemiringan garis, dapat dilakukan dengan mengikuti rumus sebagai berikut.

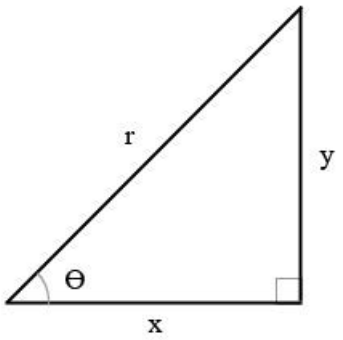
$$\text{Kemiringan garis} = (y_2 - y_1) / (x_2 - x_1) \quad (2.5)$$

Dengan informasi kemiringan garis ini dapat juga diturunkan persamaan garis lurus yang melalui posisi $P_1(x_1, y_1)$ dengan menggunakan rumus sebagai berikut.

$$y - y_1 = \text{kemiringan garis} * (x - x_1) \quad (2.6)$$

2.4.4 Trigonometri dalam Geometri

Salah satu konsep yang paling penting dalam ilmu geometri adalah trigonometri. Konsep ini menjelaskan hubungan antara garis dan sudut dalam segitiga tegak lurus atau segitiga dengan salah satu sudut 90^0. Adapun konsep trigonometri ini akan dijelaskan oleh hubungan pada gambar 2.9 sebagai berikut.



Gambar 2.9 - Konsep Trigonometri

Dimana kemudian hubungan antara nilai *x, y, r* dan θ dapat ditentukan melalui hubungan sebagai berikut.

sin θ = y / r cos θ = x / r tan θ = y / x	**(2.7)**
y = sin θ * r x = cos θ * r	**(2.8)**
θ = arc tan (y / x) = arc sin (y / r) = arc cos (x / r)	**(2.9)**

2.5 Cooperative Object Tracking with Mobile Robotic Sensor Network

Penelitian ini dilakukan oleh J. Takahashi, K. Sekiyama dan T. Fukuda dari Department Micro-Nano Systems Engineering, Nagoya University Jepang. Penelitian ini menjadi salah satu rujukan pada penelitian tugas akhir karena kemiripan permasalahan yang diteliti. Permasalahan utama yang akan diselesaikan pada penelitian Takahashi dkk ini adalah perancangan jalur komunikasi nirkabel dengan menggunakan jaringan robot untuk keperluan pengawasan objek. Berikut

akan dijelaskan spesifikasi robot yang digunakan, serta algoritma yang diajukan untuk menyelesaikan masalah.

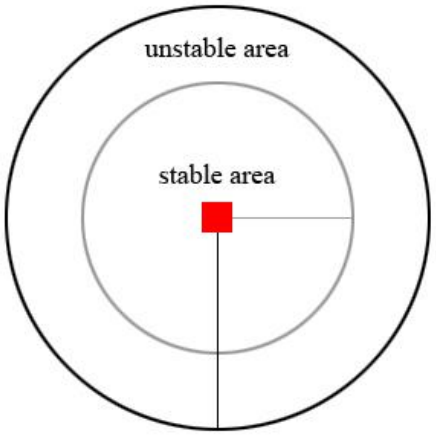
2.5.1 Spesifikasi Robot yang Digunakan

Berikut akan dijelaskan mengenai spesifikasi robot otonom yang digunakan.

1. Robot otonom yang digunakan adalah model robot beroda Amigobot dari MobileRobots Inc. Dalam melaksanakan fungsinya, robot ini didukung oleh dua modul lainnya, yaitu komponen PC model VAIO type-U dari Sony Corp. sebagai modul pengambilan keputusan, serta Perangkat ZigBee untuk modul komunikasi Peer-to-Peer.
2. Robot otonom memiliki program *controller* yang berfungsi untuk menentukan proses pengambilan keputusan yang dilakukan oleh masing-masing robot dalam simulasi. Program ini dijalankan pada modul PC dan terdiri atas beberapa *thread*, seperti *Sensing*, *Communication*, *Movement* dan *Console*. Masing-masing *thread* ini dijalankan secara *multithread* atau berjalan pada *thread* sendiri-sendiri.
3. Seperti yang disebutkan sebelumnya, robot otonom ini memiliki fungsi *sensing* untuk pendeteksian objek di sekitar robot, *communication* untuk komunikasi antar robot, *computation* untuk setiap proses komputasi yang diperlukan, serta *locomotion* untuk pergerakan robot.
4. Untuk fungsi *communication*, terdapat dua jenis komunikasi yang dapat dilakukan oleh robot pada algoritma ini, yaitu sebagai berikut.
    a. *Stable communication*, yaitu jenis komunikasi antarrobot yang menjamin bahwa data dan informasi yang dipindahkan akan berlangsung secara sempurna, artinya bebas dari error yang memungkinkan terjadinya perubahan pada data dan informasi. Rentang area dimana robot dapat melakukan komunikasi jenis ini disebut sebagai *stable communication area*, yaitu area dengan jarak radius tertentu dari pusat robot.

b. *Unstable communication,* yaitu jenis komunikasi antarrobot yang tidak dapat menjamin bahwa data dan informasi yang dipindahkan akan berlangsung secara sempurna, artinya mungkin saja terjadi error dapat menyebabkan perubahan data dan informasi. Rentang area dimana robot dapat melakukan komunikasi jenis ini disebut sebagai *unstable communication area,* yaitu area dengan jarak radius tertentu dari batas terluar *stable communication area*.

Untuk lebih jelasnya, representasi dari kedua *communication area* ini dapat dilihat pada gambar 2.10 berikut di bawah ini. Perlu diingat bahwa karena *sensing area* hanya diperlukan untuk mencegah benturan, maka pada spesifikasi robot yang digunakan pada penelitian Fukuda dkk ini *sensing area* berarti sama dengan *collision area*.



Gambar 2.10 - Communication Area

Selain itu, diasumsikan bahwa robot hanya diperbolehkan melakukan pertukaran data dan informasi pada saat mereka sudah saling terhubung dalam *stable communication area,* sedangkan *unstable communication area* hanya digunakan untuk menemukan posisi robot-robot lain yang berada di sekitar suatu robot otonom tertentu.

2.5.2 Algoritma Penyelesaian Masalah

Berikut akan dijelaskan mengenai algoritma penyelesaian masalah yang diajukan.

1. Robot otonom memulai simulasi dengan posisi awal yang acak, sedangkan perangkat pengawas dan objek yang diawasi harus diketahui posisinya.
2. Robot menyimpan informasi yang dibutuhkannya untuk melakukan kegiatan eksplorasi, yaitu informasi mengenai posisi dirinya masing-masing beserta informasi mengenai posisi objek yang akan diawasi dan perangkat pengawas selama proses simulasi. Akan tetapi, robot tidak mengetahui posisi robot-robot lainnya dalam simulasi. Informasi ini diberikan kepada robot otonom selama fase inisialisasi.
3. Robot otonom saling berkomunikasi satu sama lain untuk mengetahui posisi masing-masing. Hal ini dilakukan dengan cara sebagai berikut.
   a. Robot-robot otonom melakukan penyiaran informasi mengenai posisinya masing-masing secara *broadcast* ke dalam ruang kerja.
   b. Setiap robot otonom mempunyai rentang *unstable communication area* yang terbatas sehingga informasi ini hanya dapat mencapai jarak maksimal sejauh rentang *communication area* robot tersebut.
   c. Ketika suatu robot otonom masuk ke dalam rentang *unstable communication area* robot yang lain, robot akan dapat mengetahui posisi robot lain tersebut berdasarkan penyiaran informasi secara *broadcast* sebelumnya. Dengan melakukan proses yang sama, robot tersebut juga kemudian dapat memberitahukan informasi posisinya pada robot-robot lain dalam ruang kerja.
4. Robot otonom melakukan pengelompokan seluruh informasi mengenai posisi objek-objek dalam ruang kerja simulasi menjadi lima bagian, yaitu:
   a. *Sink*, merupakan informasi posisi perangkat pengawas.
   b. *Target*, merupakan informasi posisi objek yang diawasi. Pada penelitian ini, pengujian dibatasi terhadap pengawasan satu objek dalam simulasi.

c. Kelompok N^i, merupakan himpunan informasi posisi robot-robot yang berada dalam rentang *unstable communication area* robot otonom *i*.

d. Kelompok N_s^i, merupakan subset kelompok N^i dengan anggota himpunan merupakan informasi posisi robot-robot yang lebih dekat terhadap *Sink* daripada robot *i*.

e. Kelompok N_t^i, merupakan subset kelompok N^i dengan anggota himpunan merupakan informasi posisi robot-robot yang lebih dekat terhadap *Target* daripada robot *i*.

5. Robot otonom menentukan arah pergerakannya berdasarkan informasi lokasi yang sudah dilakukan pengelompokan seperti di atas. Terdapat dua prioritas pergerakan yang akan ditentukan dengan cara sebagai berikut.

   a. Prioritas pertama, adalah robot yang berada pada jarak paling dekat dengan robot tersebut dari anggota kelompok N_s^i. Jika kelompok ini merupakan himpunan kosong, maka prioritas pergerakan robot adalah *Sink*.

   b. Prioritas kedua, adalah robot yang berada pada jarak paling dekat dekat dengan robot tersebut dari anggota kelompok N_t^i. Jika kelompok ini merupakan himpunan kosong, maka prioritas pergerakan robot adalah *Target*.

6. Robot otonom bergerak berdasarkan aturan prioritas yang sudah ditentukan sebelumnya. Robot pertama kali akan bergerak menuju prioritas pertama hingga robot dan prioritasnya tersebut terhubung dalam rentang *stable communication area*. Selanjutnya robot akan bergerak menuju prioritas kedua sambil tetap mempertahankan rentang *stable communication area* dengan prioritas pertama. Robot akan berhenti jika kondisi simulasi sudah tidak memungkinnya untuk melakukan pergerakan.

Ketika semua robot sudah saling terhubung satu sama lain dalam rentang *stable communication area*, maka akan terbentuk jaringan yang menghubungkan *Sink* dan *Target*. Hal ini dapat terjadi karena setiap robot pasti memiliki prioritas

pertama dan prioritas kedua untuk menentukan pergerakannya. Prioritas pertama adalah robot terdekat lainnya dari robot tersebut yang juga berada lebih dekat terhadap posisi *Sink* atau perangkat pengawas. Kemudian, prioritas kedua adalah robot terdekat lainnya dari robot tersebut yang juga berada lebih dekat terhadap posisi *Target* atau objek yang diawasi. Ketika prioritas pertama tidak ditemukan, berarti robot tersebut adalah robot yang paling dekat dengan posisi *Sink*. Robot ini pun akan bergerak mendekati *Sink* untuk membangun jalur komunikasi langsung terhadap *Sink*. Begitu juga ketika prioritas kedua tidak ditemukan, berarti robot tersebut adalah robot yang paling dekat dengan posisi *Target*. Robot ini akan bergerak mendekati *Target* untuk membangun jalur komunikasi langsung terhadap *Target* sehingga akan terbentuk suatu jaringan yang lebih luas.

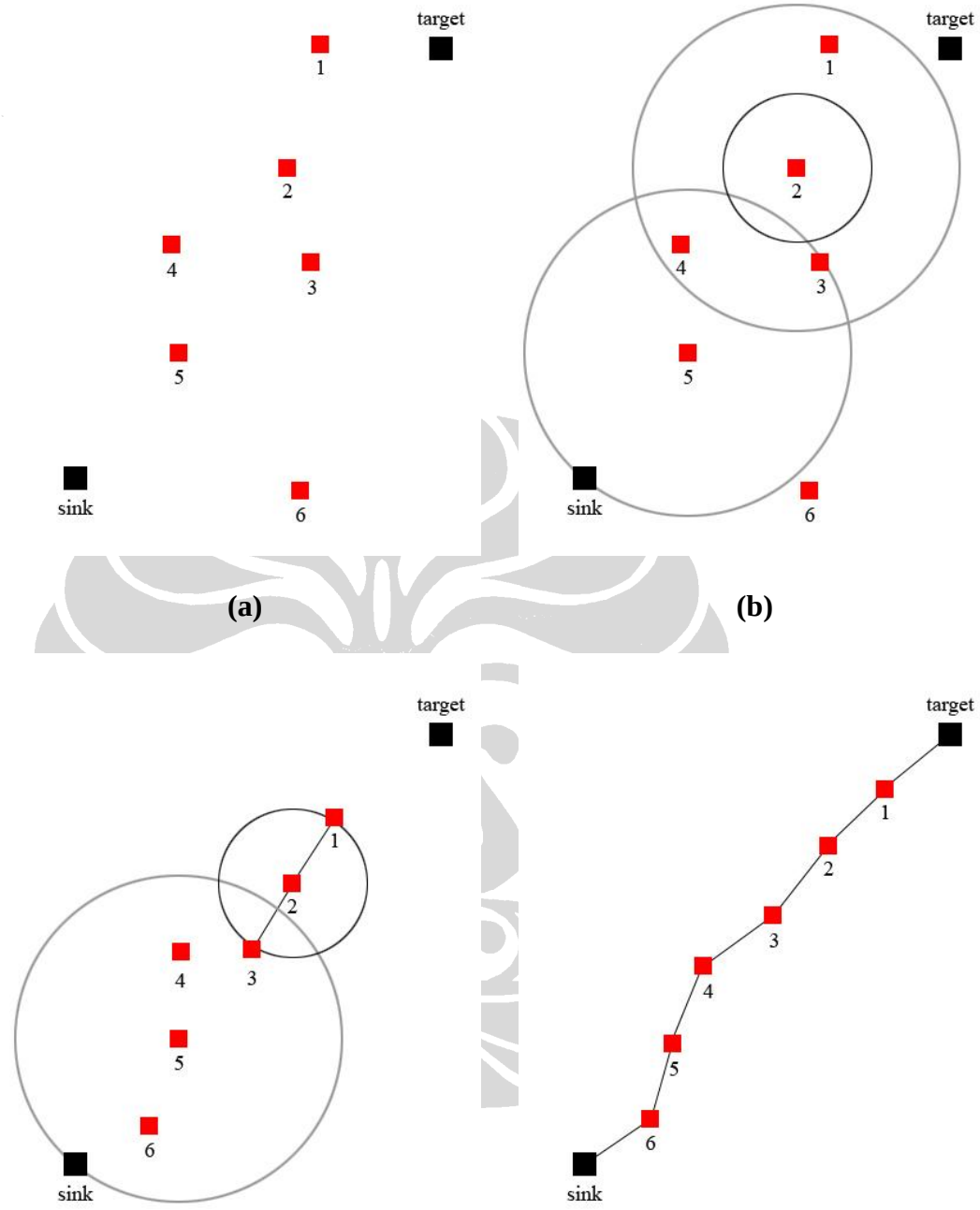
Penerapan algoritma ini dapat dijelaskan oleh contoh pada gambar 2.11 berikut di bawah. Pada contoh gambar 2.11 tersebut, objek-objek yang terlibat dalam simulasi mencakup satu perangkat pengawas atau *sink*, satu objek yang akan diawasi atau *target*, serta enam robot otonom yang akan membangun jaringan.

Gambar 2.11.a menunjukkan posisi awal masing-masing objek pada saat simulasi dimulai. Misalkan iterasi pada saat ini sedang berlangsung pada robot nomor 2 seperti yang ditunjukkan oleh Gambar 2.11.b. Robot 2 akan menyimpan informasi yang diketahuinya, sebagai berikut.

1. Posisi dirinya sendiri, *sink*, dan *target* dalam bentuk informasi koordinat.

$$\mathbf{P_2 = (x_2, y_2),\ \textit{Sink} = \{\ (x_s, y_s)\ \},\ \textit{Target} = \{\ (x_t, y_t)\ \}} \qquad \mathbf{(2.10)}$$

2. Posisi robot otonom lain dalam rentang *unstable communication area* sebagai anggota kelompok N^2, yaitu posisi robot 1, 3 dan 4 dalam bentuk informasi koordinat.



Gambar 2.11 - Penerapan Algoritma pada Penelitian

$$N^2 = \{ (x_1, y_1), (x_3, y_3), \text{dan } (x_4, y_4) \} \quad (2.11)$$

3. Posisi robot otonom yang termasuk ke dalam kelompok N_s^2, yaitu posisi robot 3 dan 4 dalam bentuk informasi koordinat.

$$N_s^2 = \{ (x_3, y_3), (x_4, y_4) \} \quad (2.12)$$

4. Posisi robot otonom yang termasuk ke dalam kelompok N_t^2, yaitu posisi robot 1 dalam bentuk informasi koordinat.

$$N_s^2 = \{ (x_3, y_3), (x_4, y_4) \} \quad (2.13)$$

5. Dengan mengikuti aturan yang dijelaskan sebelumnya, dapat ditentukan prioritas pertama bagi robot 2 adalah robot 3, sedangkan prioritas kedua adalah robot 1. Robot 2 kemudian akan bergerak mendekati robot 3 dengan mengikuti algoritma pergerakan robot.

Pada gambar 2.11.b dapat dilihat contoh kasus lain yang dialami robot 5, di mana pada saat simulasi robot ini tidak mengenali robot lain yang lebih dekat terhadap *sink* dibandingkan dirinya sendiri, maka prioritas pertama bagi robot 5 adalah *sink*. Namun ketika robot 5 bergerak mendekati *sink*, robot 6 kemudian memasuki *unstable communication area* robot 5 sehingga perlu dilakukan perubahan terhadap prioritas pertama robot 5. Kejadian seperti ini dapat terjadi pada saat yang tidak ditentukan sehingga setiap melakukan gerakan masing-masing robot kembali melakukan pengecekan informasi mengenai objek-objek di sekitarnya dan melakukan perubahan jika diperlukan.

Gambar 2.11.c merupakan kondisi ketika semua robot otonom bergerak mendekati prioritas pertama mereka masing-masing. Dapat dilihat bahwa

meskipun robot 1 sudah terhubung dalam rentang *stable communication area* dengan prioritas pertamanya yang dalam hal ini robot 2, robot 1 masih harus bergerak mendekati *sink* sebelum bergerak mendekati prioritas keduanya (dalam hal ini *target*). Hal ini disebabkan karena robot 1 masih harus mempertahankan rentang *stable communication area* dengan robot 2 sehingga ia akan tertarik oleh pergerakan robot 2 yang juga dipengaruhi robot-robot lain yang mendekati *sink*.

Ketika semua robot sudah mencapai posisi prioritas pertamanya, robot-robot otonom pun mulai bergerak mendekati prioritas keduanya. Hal ini akan menyebabkan pergerakan berantai semua mendekati posisi *target* karena pasti ada salah satu robot dalam simulasi yang prioritas keduanya adalah *target*. Robot-robot otonom kemudian akan berhenti bergerak ketika *sink* dan *target* sudah terhubung dalam suatu jalur jaringan yang utuh. Tentu saja sebelumnya harus dipastikan bahwa jumlah robot yang ada dalam simulasi mencukupi jumlah minimal robot yang diperlukan untuk membangun jaringan ini. Gambar 2.11.d merupakan contoh solusi yang diharapkan dari contoh kasus seperti ini.

Adapun maksud dari masing-masing robot harus mempertahankan rentang *stable communication area* adalah supaya nantinya ketika dilakukan pemindahan informasi dilakukan di dalam jaringan, informasi yang dikirim dan diterima dapat dijamin benar. Karena seperti yang sudah dijelaskan, komunikasi yang terjadi pada rentang *communication area* ini akan dijamin bebas dari error.