BAB 2
TINJAUAN PUSTAKA

2.1 *Game*

Mahardika (2013) menjelaskan bahwa *game* atau permainan adalah sesuatu dalam dunia buatan yang sudah memiliki ketentuan-ketentuan yang sudah ditetapkan. Ketentuan ini berfungsi untuk menentukan semua jeni saksi dan tindakan yang dilakukan dan yang dilarang oleh pemain. *Game* termasuk interaktif, partisifatif dan hiburan. Maksudnya bermain *game* akan merasa senang dan terhibur dengan berperan kedalam *game*nya secara aktif [4]. *Game* dikelompokan ke beberapa macam, yaitu terdiri dari jenis dimensi, jenis *genre*, dan jenis *platform* yang digunakan.

1. *Game* berdasarkan jenis dimensi

Dimensi adalah bentuk yang terdiri dari lebar, panjang dan sebagainya atau bentuk angka yang ada hubungannya dengan sifat metrik dari objek tertentu. Berdasarkan beberapa objek dari suatu jenis dimensi *game* terbagi kedalam beberapa macam, yaitu:

a. *Game* 2D

   Game berbentuk dua dimensi merupakan permainan yang memiiki objek satu bidang datar atau memiliki dua sisi. Pergerakannya berdasarkan titik koordinat sumbu X dan Y. pergerakan bisa secara keatas, kebawah, kekiri kekanan ataupun miring diagonal. Pergerakan kamera statis dan bergerak *side scrolling*. Pergerakan kamera statis yaitu latar tempat permainannya tidak dalam bergerak. Dan gerakan *side scrolling* yaitu kamera mengikut pergerakan karakter.

b. *Game* 2.5D

   Game 2.5 dimensi atau disebut juga 3D datar (3D *plane*) yaitu permainan termasuk 3D sebagian atau tidak sepenuhnya. Pergerakannya sama dengan 2D tapi terdapat beberapa objek yang sudah *rendering* 3D.

c. *Game* 3D

Game tiga dimensi adalah permainan yang memiliki tiga sisi titik koordinat X, Y dan Z. Jenis ini dapat melihat objek dari semua arah atau sisi.

2. *Game* berdasarkan jenis *genre*.

Game berdasarkan jenis *genre* adalah permainan yang tergolong bedasarkan interaksi,atau berdasarkan perbedaan dari segi model tantangan dan tata tertib dari *game*nya. Secara umum jenis ini dibagi dalam beberapa macam juga, yaitu:

a. *Sport Games*

Game yang berjenis olahraga yaitu yang memilki ketentuan, aturan dan tantangannya seperti olahraga asli. Contohnya olahraga sepakbola, voli dan sebagainya

b. *Strategy Games*

Game strategi yang berasal dari permainan dalam papan seperti othello, dam daman dan catur. Permainan ini harus berpikir mengatur dan merancang strategi untuk melewati tantangannya. Biasanya karakter lebih dari satu.

c. *Puzzle Games*

Game teka-teki adalah permainan dimainkan yang memiliki tujuan untuk memecahkan suatu permasalahan atau mencari suatu jawaban. Pada permainan teka-teki akan diberi beberapa arahan atau petunjuk yang dapat memudahkan dan membantu dalam memecahkan misterinya.

d. *Adventure Games*

Game petualangan adalah permainan yang memiliki suatu cerita interaktif tentang pemain atau karakter. Biasanya pada permainan ini akan ada tempat yang harus jelajah pemain. Dari tempat yang dijelajahinya pemain harus memiliki navigasi ke suatu tempat, atau melawan musuh, mencari suatu pesan rahasia dan lainnya.

e. *Action Games*

Game aksi adalah permainan yeng penuh dengan berbagai macam dan jenis aksi, pemain harus mempunyai suatu keterampilan terkait reaksi dan

respon cepat dalam menghadapi, atau melawan ataupun menghindari serangan lawan.

f. *Board Games*

Board games atau permainan berjenis papan yaitu sebuah permainan bisa dimainkan dengan memakai suatu alas atau tempat dalam bentuk persegi. Dan biasanya memiliki beberapa komponen pendukungnya berupa bentuk koin, bentuk bidak, bentuk kertas, kartu dan sebagainya. *Board games* biasanya dapat dimainkan lebih dari satu orang atau *Multiplayer*. Sehingga pemain dapat lebih berinteraksi langsung dengan para pemain atau musuh lainya.

3. *Game* bedasarkan jenis *platform*.

Platform adalah pengkombinasi atau gabungan dari suatu yang dinamakan perangkat lunak (*software*) dan suatu perangkat keras (*hardware*) yang digunakan dalam memainkannya. Berdasarkan *platform, game* dibagi kedalam beberapa jenis atau macam juga, yaitu:

a. *Arcade Games*, merupakan *game* mesin yang dirancang dan dibuat untuk suatu *video games* tertentu. Terdapat alat atau fitur yang bisa membuat bermain lebih serasa nyata seperti terdapat stir mobil, menggunakan pistol, ada sensor gerakan dan kursi khusus.

b. *PC Games*, merupakan permainan yang dapat dimainkan menggunakan benda komputer atau laptop.

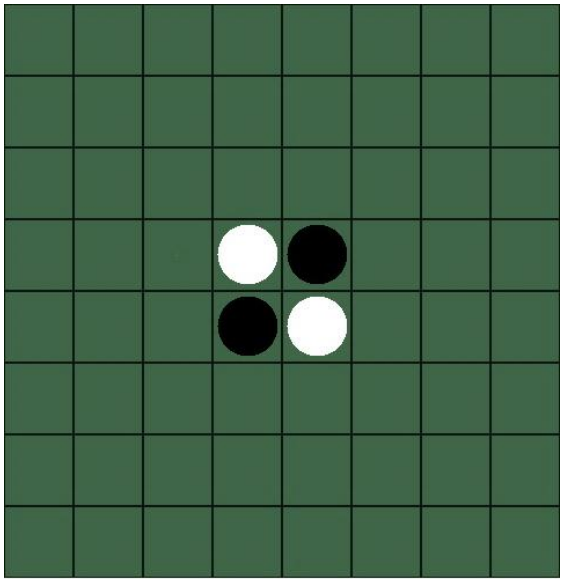
c. *Console Games*, merupakan permainan yang biasanya menggunakan suatu bentuk konsol tertentu dalam mencoba memainkannya. Seperti permainan *playstation, game Nintendo*, jenis *Xbox* dan sebagainya.

d. *Handheld Games*, merupakan permainan yang menggunakan suatu bentuk konsol khusus dapat dibawa oleh orang kemana-mana seperti laptop atau disebut juga *portable*. Seperti *Sony* PSP.

e. *Mobile Games*, merupakan permainan yang biasanya bisa atau dapat dimainkan pada *mobile phone*, jenis *smartphone* atau bentuk PDA.

2.2 Othello

Permainan Othello atau riversi merupakan salah satu jenis permainan papan yang berbasis strategi. Othello dimainkan oleh dua orang pemain diatas papan persegi yang biasanya berkuran 8 x 8. Di dalam papannya terdapat kotak-kotak kecil sebanyak 64 kotak. Setiap pemain, masing-masing mempunyai bidak atau koin sebanyak 64 buah dengan warna yag berbeda. Biasanya satu pemain dengan bidak warna putih dan yang satunya lagi warna hitam. Pada awal permainan akan diletakkan dua bidak hitam dan bidak koin putih pada tengah-tengah papan [1]. Untuk memperjelas lihat Gambar 2.1.



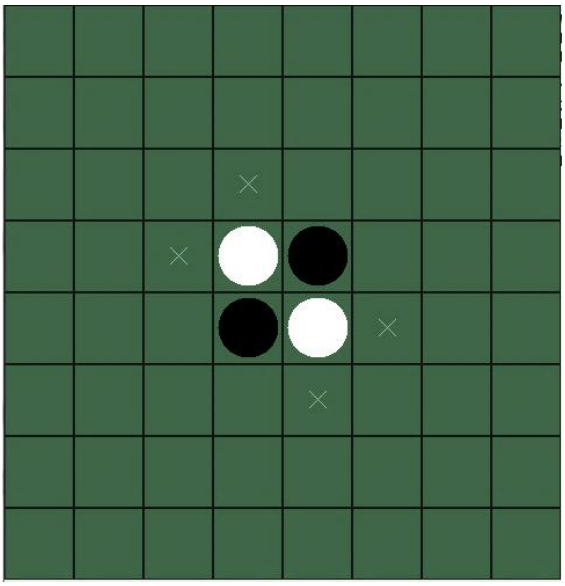
Gambar 2.1 Keadaan awal permainan othello

2.2.1 Aturan Permainan Othello

Menurut Arie Setiawan dkk. (2013) bahwa Aturan main permainan othello yaitu cara menggunakan bidak dalam memainkannya. Bidak-bidak tersebut nantinya akan saling mengapit dan melangkahi. Untuk lebih jelasnya berikut adalah aturan umum pada permainan othello:

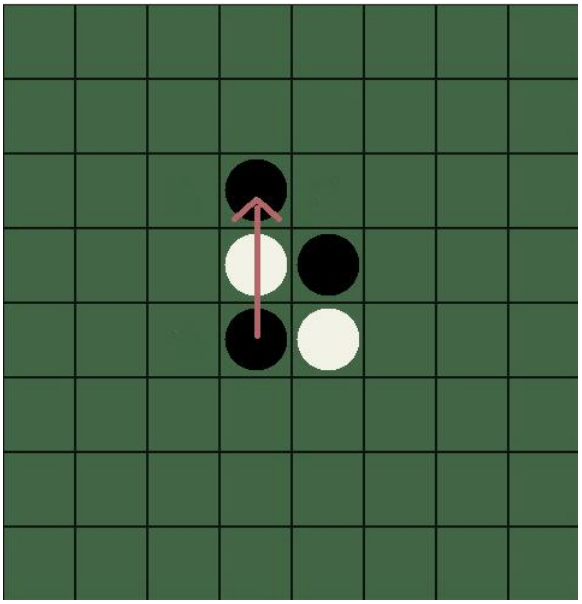
1. Pemain yang mulai pertama adalah pemain dengan bidak hitam, cara bermainnya bidak hitam harus melangkahi dan mengapit bidak putih pada papan. Kemudian bidak putih yang diapit akan berubah menjadi bidak hitam. Pemain hanya diperbolehkan satu kali menentukan langkah setelah itu bergantian giliran pemain lain dengan bidak putih.

2. Langkah yang bisa dilakukan adalah keatas, kebawah, kekiri, kekanan dan diagonal yang membentuk garis lurus. Dan langkah yang diambil harus mengapit bidak musuh. Berikut kondisi pengambilan langkah pada bidak hitam dapat dilihat pada Gambar 2.2.



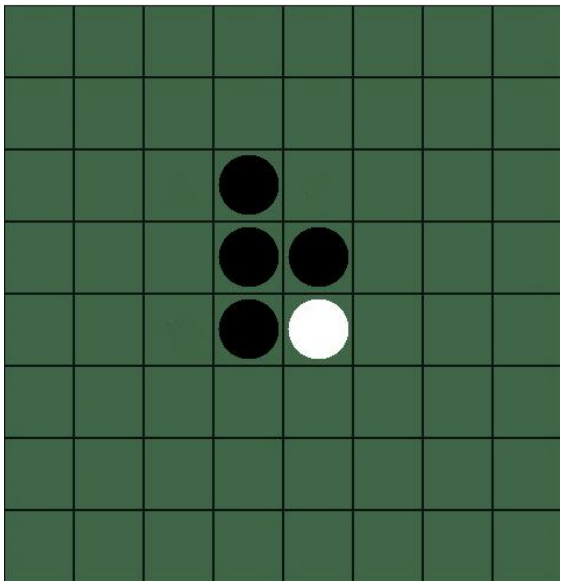
Gambar 2.2 Kotak yang mungkin (giliran bidak hitam)

Misalkan bidak hitam menentukan langkah ke atas untuk mengapit bidak putih diatasnya, lebih jelasnya dapat dilihat pada Gambar 2.3.



Gambar 2.3 Bidak hitam menentukan langkah

Kemudian bidak putih yang langkahi atau diapit akan berubah menjadi bidak hitam. Lebih jelas pada Gambar 2.4.



Gambar 2.4 Perubahan bidak putih jadi hitam yang diapit

3. Jika terdapat pemain tidak dapat melakukan pemilihan langkah dikarenakan tidak ada kotak legal, maka giliran diganti ke pemain lawan.
4. Jika keduanya sudah tidak ada pemilihan langkah lagi, maka bisa dikatakan permainan selesai atau sudah berakhir.
5. Permainan dikatakan selesai atau berakhir jika dipapan penuh diisi oleh bidak-bidak. Atau papan belum penuh oleh bidak tapi hanya terdapat bidak dalam satu jenis warna saja. Yang menang bisa ditentukan berdasarkan jumlah bidak pada akhir permainan. Jumlah bidak terbanyak adalah pemenangnya [1].

2.2.2 Strategi Permainan Othello

Permainan othello merupakan permainan yang bertipe strategi. Pemain harus bisa memikirkan dan menyusun berbagai strategi supaya menjadi pemenang [10]. Maka dibutuhkan beberapa strategi, dan berkit ini beberapa hal yang harus menjadi perhatian saat memainkannya:

1. Jumlah Bidak

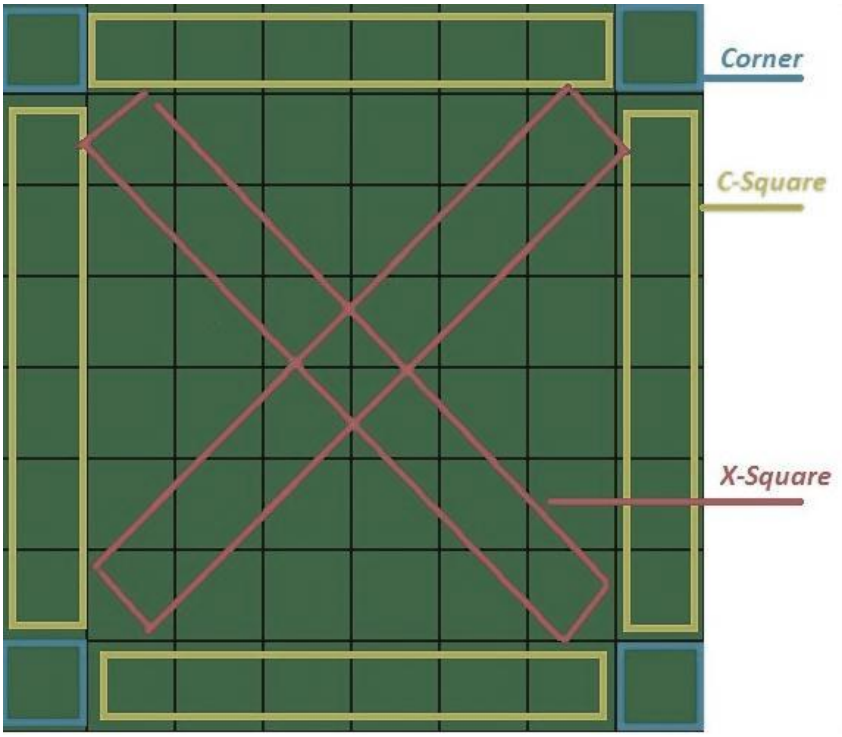
Tujuan utama permainan othello untuk menjadi pemenang adalah untuk mendapatkan jumlah bidak paling banyak diakhir. Oleh sebab itu, para pemain othello harus berusaha memilih langkah pada kotak dengan mendapat bidak lawan sebanyak mungkin. Strategi ini biasanya sering digunakan pada kondisi akhir untuk mengumpulkan jumlah bidak sebanyak mungkin, tapi jika digunakan pada kondisi awal dan kondisi tengah hanya akan mendapat kondisi buruk. Jika menggunakan strategi ini disarankan untuk tetap menjaga jumlah bidak sesedikit mungkin pada kondisi awal dan kondisi tengah dan tentunya harus bisa mendapat banyak bidak di kondisi akhir permainan.

2. Kotak pada Sudut (*corners*), Kotak *XSquares*, dan Kotak *CSquares*

   Jika pemain menaruh bidak pada kotak bagian sudut, maka bidak itu bisa dikatakan tidak akan bisa untuk dibalik lagi, dikarenakan sudak tidak bisa diapit lagi oleh siapapun. Bidak sudut ini dapat digunakan untuk membalik bidak-bidak lawan di sekitarnya. Maka kotak pada bagian sudut adalah kotak penting. Maka, Disarankan untuk memperoleh kotak bagian sudut di kondisi awal dan di kondisi tengah.

   Kemudian kotak *XSquare* merupakan kotak di yang berada pada sebelah diagonal kotak bagian sudut. Kotak ini bisa dikatakan kotak bahaya, dikarenakan jika pemain meletakan atau menaruh bidak pada bagian ini, lawan bisa menggunakan untuk kotak bagian sudut di tempat sebelahnya. Maka, saran agar tidak memilih beberapa langkah di kotak bagian ini pada kondisi awal dan kondisi tengah.

   Dan terakhir, kotak *CSquare* merupakan beberapa kotak di bagian sebelah kotak yang berada di sudut baik itu secara kanan, kiri atau bawah atas. Kotak ini bisa dikatakan juga kotak bahaya, karena sama seperti kotak *Xsquare* bisa dimanfaatkan untuk menaruh di kotak bagian sudut. Sarannya tidak memilih langkah di kotak *CSquare* pada kondisi awal dan kondisi tengah. Mengenai kotak sudut, *X-square* dan *C-square* lebih jelasnya dapat dilihat pada Gambar 2.5.



Gambar 2.5 Letak Kotak Corner, *C-Square* dan *X-Square*

3. Bidak Stabil

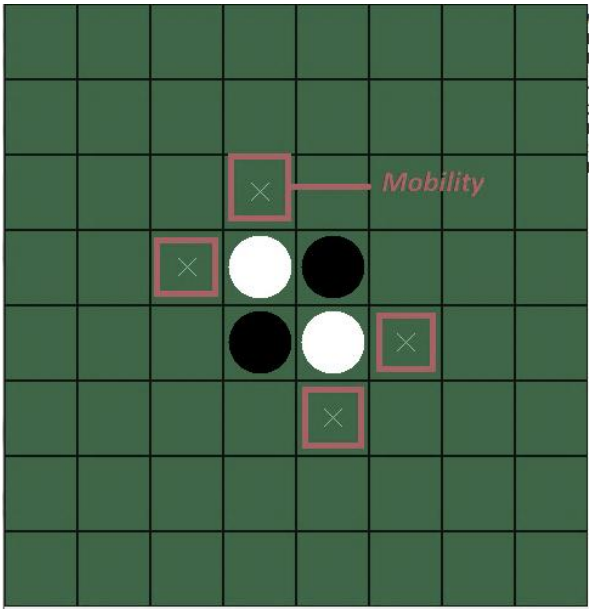
   Bidak stabil merupakan bidak yang tidak bisa dilangkahi oleh bidak lawan, sehingga tidak bisa untuk dibalik. Bidak stabil ini bisa dijadikan sebagai penentu dalam memenangkan pertandingan karena sudah tidak bisa lagi dipakai untuk menjadi bidak lawan. Maka, saran untuk mendapat jumlah bidak stabil di semua kondisi.

4. *Mobility*

   Di awal permainan bidak hitam dapat memilih langkah pada empat kotak. Kemudian bagian bidak putih dapat memilih langkah pada tiga kotak, dan seterusnya, yaitu yang bisa melangkahi bidak lawan, bisa berubah tergantung dari langkah bagian lawan pada sebelumnya. Jumlah atau beberapa kotak dapat dilangkahi ini dinamai *mobility*.

   Banyak *mobility*, maka akan mendapat beberapa kemungkinan pemilihan langkah yang baik, atau sebaliknya maka akan semakin sedikit akan mendapat langkah yang buruk. Sarannya untuk mendapat *mobility* sebanyak

mungkin di semua kondisi. Untuk lebih jelas , berikut bentuk *mobility* pada permainan othello dapat dilihat pada Gambar 2.6.



Gambar 2.6 *Mobility* bidak hitam pada permainan othello

5. Bidak bagian Tepi atau *Frontier Disc*

   Bidak bagian tepi adalah bidak yang terdapat pada tepi kotak yang kosong. Ketika jumlah bidak tepi banyak dipunyai, maka kemungkinan silawan bisa menetukan langkah di bagian tepi kotak yang kosong sampingnya, sehingga mengakibatkan *mobility* milik lawan dapat meningkat. Saran untuk mengambil lebih sedikit bidak tepi pada semua kondisi karena bidak tepi yang diambil lebih sedikit ini akan berusaha untuk bidaknya selalu bisa terhubung dengan yang lainnya dan terletak di kepungan bidak lawan.

2.3 Fungsi Evaluasi Permainan Othello

Fungsi evaluasi merupakan fungsi untuk menghitung skor pada satu langkah. Umumnya, fungsi ini digunakan pada othello jadi bahan pertimbangan dari berbagai strategi permainan yang digunakan [10]. Maka fungsi evaluasi perhitungan skor pada sebuah langkah permainan othello umumnya mengikuti persamaan 2.1 berikut.

$$evaluasi = wd * pieces + wc * corner + ws * stables + wm * mobility + wf * frontier \quad (2.1)$$

Keterangan:

wp = -3 (untuk bobot strategi *pieces*)

wc = 5 (untuk bobot strategi *corners*)

ws = 9 (untuk bobot strategi *stables*)

wm = 7 (untuk bobot strategi *mobility*)

wf = -5 (untuk bobot strategi *frontier*)

2.4 *Artificial Intelligence*

Artificial Intelligence atau dalam bahasa indonesia disebut kecerdasan buatan didefinisikan berbeda-beda oleh para ahli tergantung pada sudut pandangnya masing-masing. Ada yang didefinisikan tentang pola berpikir atau logika pada manusia saja, atau secara luasnya tentang sifat dari tingkah laku atau prilaku manusia. Kecerdasan buatan merupakan perangkat komputer yang bisa memahami dengan lingkungannya dan bisa mengambil suatu tindakan yang memaksimalkan keberhasilan di lingkungan tersebut untuk beberapa tujuan [5]. Pendekatan AI dikelompokan kedalam empat kategori yaitu:

1. *Thinking humanly*

Pendifinisian AI merupakan cara berpikir yang menyerupai berpikir manusia. Terdapat dua teknik yaitu melalui introspeksi maksudnya mengambil dari gagasan atau ide kita sendiri ketika melakukan proses berasumsi dan melalui percobaan atau eksperimen psikologi.

2. Acting humanly

Acting humancly melakukan pendekatan dengan menirukan prilaku seperti manusia yang diperkenalkan pada tahun 1950 oleh Alan turing dalam merancang cara kerja pengujian melalui teletype yaitu jika penguji (integrator) tidak dapat membedakan yang mengintrogasi antara manusia dan komputer maka komputer tersebut dikatakan lolosdari turning test.

3. Thinking rationally

Ada pendekatan pada teknik *thinking rationally*, yaitu kesatu susah dalam membentuk suatu pengetahuan jenis informal dan merubah ke pengetahuan formal dipakai untuk notasi pada logika. dan kedua memiliki beberapa perbedaan dari bisanya mencari solusi dalam bentuk prinsip dan menerjemahkannya di dunia nyata.

4. *Acting rationally*

Pendekatan *acting rationally* memiliki tujuan untuk mengembangkan agen yang rasional. Dimana agen adalah sistem komputer yang beroperasi secara otomatis, yang mampu mempersepsikan lingkungan disekitarnya, kemudian dapat beradaptasi terhadap perubahan, dan mampu membuat tujuan sekaligus berusaha untuk mencapainya.

2.5 *Swarm Intellingence*

Swarm intellingence merupakan salah satu jenis metode kecerdasan buatan yag dilihat dari tingkah laku kumpulan serangga pada suatu sistem desentralisasi dan bisa mengatur dirinya sendiri [18].

2.6 Algoritma *Max Min Ant System* (MMAS)

Max-Min Ant System (MMAS) merupakan Algoritma MMAS adalah varian dari algoritma kelompok serangga semut yaitu algoritma yang dilihat dari prilaku kumpulan semut dalam menemukan rute atau jalur terpendek dari sumber makanan ke tempat sarangnya atau sebaliknya.Pemilihan jalur terpendek dapat dilihat pada jejak kaki semut (*Pheromone*). *Pheromone* sendiri merupakan sejenis kimia yang bersumber dari suatu kelenjar yang dinamakan endokrin yang manfaatkan untuk mengenali dan mengetahui dari sesama jenis, kelompok, individu lain dan juga membantu dalam proses reproduksi (Agus Laksono, 2009).

Algoritma MMAS memiliki beberapa perubahan utama yang membedakan dengan pengembangan ACO lainnya. Berikut perubahan pada algoritma MMAS:

1. *Pheromone* dapat dilakukan pada garis dari lintasan terbaik yang dapat didapat pada pertama kali algoritma dipakai atau dilintasan paling baik yang didapat pada proses iterasinya. Atau bisa juga pada keduanya. Akan tetapi,

kemungkinan dapat mengakibatkan kondisi stagnasi dimana akibatnya beberapa semut melewati arah sama, dikarenakan kelebihan *pheromone* pada garis, walauun termasuk dari suatu lintasan yang terbaik.

2. Untuk mengatasi kondisi stagnasi, maka algoritma ini diberi sautu batasan pada nilai *pheromone* dengan interval [τ_{min} ,τ_{max}].
3. Proses insialisasi *pheromone* pada batas atas *pheromone*, Dimana secara seksama tingkatan evaporasi untuk *pheromone* kecil, ditingkatkan eksplorasi pada lintasan ketika pencarian dimulai.
4. Proses inisialisasi *Pheromone* diulang kembali ketika adan kondisi stagnasi atau jika tidak didapat lintasan berdasarkan iterasi yang diharapkan.

2.6.1 Langkah – langkah penyelesaian masalah Algoritma MMAS

Agus Leksono (2009: 31-32) menjelaskan bahwa terdapat langkah-langkah yang perlu dilakukan dalam peyelesaian masalah dengan menggunakan algoritma MMAS. Berikut adalah langkah – langkah penyelesaian masalahnya:

1. Menginisialisasikan parameter untuk menempatkan beberapa semut diawal pada beberapa atau n titik. Sebelum menentukan parameter, tentukan jarak(d_{rs}) antara sarang(r) ke tempat mencari makanan(s) terlebih dahulu dapat dihitung dengan persamaan 2.2.

$$drs = \sqrt{(x_r - x_s)^2 + (y_r - y_s)^2} \qquad (2.2)$$

setelah jarak diketahui maka jarak tersebut adalah dihitung *visibility*nya (η_{rs}) (invers dari jarak) dengan persamaan 2.3.

$$\eta_{rs} = \frac{1}{d_{rs}} \qquad (2.3)$$

Parameter MMAS terdiri dari α sebagai parameter yang mengontrol bobot relatif dari *pheromone*, β sebagai parameter pengendali jarak, ρ sebagai parameter penguapan *pheromone*, τ_0 sebagai *pheromone* awal, jumlah semut m sebagai jumlah semut dimana m sama dengan banyak titik(n), τ_{min} sebagai

parameter batas bawah nilai *pheromone* dan τ_{max} sebagai paramter batas atas nilai *pheromone* dihitung dengan persamaan 2.4.

$$\tau_{max} = \frac{1}{\rho C^{bs}} \qquad (2.4)$$

C^{bs} = panjang pada lintasan baik yang didapat pada dimulainya proses algoritma.

2. Kemudian semut akan memilih titik untuk selanjutnya berdasarkan dari persamaan 2.5.

$$P_{rs}^{k} = \begin{cases} \frac{[\tau_{rs}]^{\alpha}.[\eta_{rs}]^{\beta}}{\sum_{u \in j_r^k} [\tau_{ru}]^{\alpha}.[\eta_{ru}]^{\beta}} & untuk\ s \in j_r^k \\ 0 & untuk\ s\ lainnya \end{cases} \qquad (2.5)$$

P_{rs}^{k} = Nilai probabilitas pada anggota semut dengan tanda k di titik r(sarang) yang melalui dan bertujuan titik s(tempat makanan ditemukan).

τ_{rs} = nilai untuk *Pheromone* pada garis titik r ketempat titik s.

Dan j_r^k = himpunan dari beberapa titik yang siap dikunjungi anggota semut tanda k yang berada dilokasi titik r, dan di bentuk persamaan(2.1) dengan mengalikan *pheromone* pada garis(r,s) dengan suatu nilai *visibility* atau invers (η_{rs}). Teknik ini dapat menentukan garis yang paling terpendek dan mempunyai nilai suatu *pheromone* yang paling besar.

3. Semua semut sudah melakukan pemilihan lintasan, panjang lintasan semut akan dihitung dan didapat lintasan paling baik.
4. Dilakukan *update pheromone*, pada proses tahap ini penambahan *pheromone* dapat dilakukan di lintasan yang paling baik ketika mulainya algoritma atau pada lintasan paling terbaik yang didapat pada proses iterasinya, ditambah untuk keduanya. Sebelum melakukan *update pheromone* tentukan *delta* nya dengan persamaan 2.6.

$$\Delta\tau_{rs}^{best}\begin{cases}{}^{1}/_{C^{best}} & Jika\ semut\ menemukan\ best\ so-for-tour\\ {}^{1}/_{C^{ib}} & Jila\ semut\ menemukan\ iteration\ best-tour\end{cases} \quad (2.6)$$

Dan berikut aturan *update pheromone* pada MMAS menerapkan persamaan 2.7.

$$\tau_{rs} \leftarrow (1-\rho).\tau_{rs} + \Delta\tau_{rs}^{best} \quad (2.7)$$

C^{best} = panjang lintasan paling terbaik

C^{ib} = panjang iterasi paling terbaik pada lintasan.

5. Dilakukan pengujian di saat kondisi akhir sebagai tanda proses pencarian lintasan terpendek berhenti. Jika, semua anggota semut berhasil menemukan lintasan yang sama dan memiliki nilai jarak sama juga (konvergensi) atau Nilai NCmaxnya tercapai maka bisa dihentikan pencariannya, jika masih belum maka proses untuk melakukan pencarian terus dilanjutkan hingga sampai adanya kondisi konvergensi atau nilai NCmax tercapai.

2.7 Pemrograman yang Berorientasi Objek

Pemrograman objek merupakan jenis dari pembangunan untuk perangkat lunak yang didefinisikan perangkat lunak menjadi beberapa objek mengenai operasi dan Data di dalamnya. Pembangunan objek merupakan teknik pada sistem aplikasi yang akan dibangun dengan menggunakan pendekatan secara objek dan sistematis. Pembangunan ini dibentuk dari aktivitas menganalisis secara objektif, perancangan, pemrograman, dan pengujian bentuk objek [6].

Pemrograman berorientasi objek dibagi menjadi beberapa ciri utama, berikut adalah ciri-cirinya:

1. Kelas

Kelas atau *class* adalah bentuk abstrak pada objek yang dibangun dari penyederhanaan, atau bisa dikatakan juga kelas itu adalah bagian dari objek itu sendiri [6]. Yang membedakan kelas dengan objek, yaitu:

a. *Class* adalah rancangan dan *object* adalah perwujudan dari suatu *class*.

b. *Class* bersifat abstrak sementara object bersifat konkrit (atau nyata).

2. Objek

Sebuah objek pada pemrograman berorientasi objek dapat didefinisikan sebagai berikut:

a. Objek dipunyai oleh kelas, maka objek tidak adak jika tidak ada kelass.
b. Sebuah objek adalah pembungkusan (*encapsulation*) yang membungkus operasi dan data untuk melakukan pemrosesannya.
c. Atribut-atribut objek membantu untuk menyimpan dan menjaga status objek. Atribut-atribut ini menentukan apa yang berkaitan mengenai objek. *Method* objek adalah satu-satunya cara untuk mengakses data dan memodifikasi statusnya. Cara pengaksesan dan pemodifikasian data dilakukan dengan mengirimkan sebuah pesan ke objek tersebut [7].

2.8 UML (*Unified Modeling Language*)

Menurut Bambang (2004) bahwa *Unified modeling language* (UML) merupakan model berorientasi objek yang dipakai dalam menggambar, menspesifikasi, mendokumentasi dari kontruksi sistem dan pembangunan aplikasi. UML bisa dikatakan juga sebagai bahasa dalam bentuk visual untuk melakukan pemodelan dan cara berkomunikasi tentang sebuah sistem dengan memakai suatu diagram dan pendukung lainnya. Awal pengembangan UML yaitu pada tahun 1994 dengan melakukan kerja sama antara Grady Booch dan james rumbauh dalam mengkobinasi dua metode booch dan OMT. Dan pada tahun 1997, UML di usulkan kepada OMG (*Object Management Group*) yaitu standarisasi teknologi objek supaya dijadikan notasi dan bahasa pemodelan. Model UML bertujuan untuk:

1. Penyedia bahasa pemodelan grafis yang ekpresif yang siap pakai untuk pengembangan sistem.
2. Penyedia mekanisme dan spesifikasi mendapat konsep-konsep pada sistem
3. Penyedia jenis formal dalam memahami pemodelan sistem.
4. Mendukung konsep-konsep pengembangan derajat yang lebih tinggi seperti framework, pattern, kolaborasi dan komponen.

2.8.1 Diagram *Use Case*

Prabowo (2011) mengungkapkan bahwa *Use case* digunakan untuk menggambarkan atau mendeskripsikan bentuk fungsionalitas dari sebuah sistem. *Use case* berfungsi merepresentasikan hubungan antara bagian aktor dan bagian sistem. *Use case* sebagai berbagai pekerjaan pada sistem sedangkan aktor merupakan bentuk entitas pada mesin atau manusia yang melakukan interaksi pada sistem untuk mengerjakan suatu perintah tertentu. *Use case* digunakan untuk menyusun kebutuhan sebuah sistem, menjelaskan perancangan sistem pada klien [8].

Use case dapat meng*includekan* fungsi dari *use case* lain. Atau dapat di*include*kan lebih dari satu,maka dapat terhindar dari duplikasi fungsionalitas. *Use case* bisa meng*extend*kan ke *use case* lainnya dengan behaviournya sendiri.

2.8.2 *Activity Diagram*

Prabowo (2011) mengungkapkan bahwa diagram yang menjelaskan alur dari aktivitas pada perancangan sistem, bagaimana setiap alur berawal, proses apa saja yang dilakukan, dan bagaimana alurnya berakhir. Diagram ini menjelaskan proses jenis paralel pada setiap eksekusi. *Activity diagram* lebih menjelaskan mengenai beberapa proses dan beberapa jalur aktivitas di setiap level atas [8].

2.8.3 *Class Diagram*

Prabowo (2011) mengungkapkan bahwa diagram ini menjelaskan mengenai deskripsi dan struktur pada setiap kelas, objek dan *package* serta keterkaitan antara satu dengan yang lainnya seperti agregasi, *inherintence*, asosiasi dan sebagainya [8].

2.8.4 *Sequence Diagram*

Prabowo (2011) mengungkapkan bahwa *Sequence diagram* menggambarkan hubungan antar objek yang berada di dalam sistem. Diagram *Sequence* juga terdiri diantar suatu dimensi bentuk vertikal(waktu) dan dimensi bentuk horizontal(objek yang terhubung). *Sequence diagram* biasanya dipakai dalam menjelaskan

rangkaian dari langkah-langkah untuk rangsangan dari kejadian dalam mendapatkan keluaran tertentu. [8].

2.9 Java

Java termasuk kedalam bahasa pemrograman yang berjenis orientasi objek, yang artinya pemodelan yang berdasarkan objek. Objek sendiri didefinisikan sebagai konsep atau benda yang memiliki batasan, identitas dan arti yang berada dilingkungan sekitar [9]. Java awalnya dibuat ketika perusahaan Sun Microsystem mengerajakan *Green Project* pada tahun 1991. Dan dirilis pada tahun 1995 dengan versi 1.0.2. Nama java sendiri berasal dari jenis kopi kesukaan Gosling yang biji kopinya digiling langsung. Bahasa Pemrograman java memiliki beberapa karakteristik, yaitu:

1. Bersifat sederhana
2. Bersifat orientasi objek
3. Terdistribusi dengan mudah
4. Dijalan kan dengan *Interpreter*
5. Bersifat *robust* artinya memiliki realibilitas tinggi
6. Aman
7. *Architecture Neutral*
8. Sangat portabel
9. *Multithreaded*
10. Dinamis

2.10 Studi Literatur

Studi literatur yang dilakukan akan dijadikan referensi yang berkaitan dengan penelitian, berikut hasilnya sebagai berikut:

1. Terdapat penelitian permainan othello yang membandingkan dengan menggunakan metode *minimax* dan menggunakan metode *alpha-beta pruning*. Kedua metode ini diterapkan pada permasalahan memilih langkah dalam suatu tingkat kedalaman dan membandingkan kinerja waktu yang diperoleh. Konsep kedua metode tersebut akan mencari langkah yan membuat

musuh mendapat kerugian maksimum dan untuk diri sendiri akan mendapat kerugian minimum. Pengujiannya dilakukan dengan tiga kali pengujian. Pengujian pertama diuji dalam beberapa kedalaman dimana dpertandingkan *minimax vs alpha beta*. Hasilnya *minimax* menang di kedalaman 2 dan 3 sedangkan *alpha beta pruning* menang dikedalaman 4, 5 dan 6. Kemudian dibalik *alpha beta vs minimax* dengan hasil *alpha beta* menang dikedalaman 5 sedangkan *minimax* menang pada kedalaman 2, 3, 4 dan 6. Dipengujian kedua sama tetapi dicatat juga waktu dalam menentukan langkahnya. Dimana *alpha beta* unggul dari segi waktu dibandingkan *minimax*. Ini disebabkan *minimax* tetap melakukan pencarian nilai terbesar dan terkecil pada setiap pohon pencarian. Sedangkan *alpha-beta pruning* ketika dapat nilai terbesar atau terkecil maka pencarian berhenti. Dan pengujian ketiga *minimax vs alpha beta* dengan 25 pertandingan, dimana *minimax* 13kali menang (52%) dibanding *alpha beta* 12 kali menang(48%). Dibalik *alpha beta* vs *minimax* dengan hasil *alpha beta* dan minimax sama-sama mendapat 12 kali menang (48%) dan dapat satu kali imbang dengan presentase 4% [1].

2. Penelitian Keiji Kamei dan Yuuki Kakizoe pada permainan othello bahwa terdapat tiga algortima yang digunakan pada penelitiannya yaitu pertama algoritma *Self Organizing Maps* (SOM) untuk mengetahui dan menghapal posisi bidak pada papan, menentukan neuron terbaik untuk mentukan posisi terbaiknya. Kedua algoritma Temporal *Difference Learning* (TD) untuk mempelajari langkah yang optimal dan melanjutkan permainan ke pihak lawan. Dan ketiga algoritma *monte carlo learning* untuk mengoleksi banyak bidak yang didapat dan mempelajari pergerakan yang optimal. Pada percobaan yang telah dilakukan SOM menentukan posisi bidak berdasarkan warna bidak dan mendapat lebih dari 20% tingkat kemenangan dibanding metode lain. Proses pembelajaran gerakan optimal diterapkan algoritma *TD Learning* dan *monte carlo* dengan hasil yang dilakukan pada 4 langkah terakhir algoritma TD *learning* menjadi kinerja terbaik dengan tingkat kemenangan 60% sementara *monte carlo* mendapat 58% [12].

3. Terdapat penelitian yang dilakukan untuk Othello dengan metode *Negascout* dan metode MTDF (*Memory enhanced Test Driver valuef*). *Negascout* merupakan salah satu metode pencarian *minimax* dengan langkah pertama yang diperoleh ialah langkah terbaik, sedangkan sisanya langkah terburuk. Kemudian MTDF yaitu metode pencarian yang memakai *upperbound* dan *lowerbound* dengan melakukan pemanggilan *alpha-beta with memory* secara berulang-ulang. Percobaan dengan menguji algoritma *negascout* melawan MTDF, yang dievaluasi menggunakan strategi *edge table, mobility, potential mobility* atau *frontier* dan penguasaan *corner*. Pengujian dilakukan dengan mempertandingkan *Negascout* dan MTDF dengan permainan othello yang telah didownload dengan nama cyclog. Dimana hasil *negascout* vs cyclog dengan 12 kali pertandingan didapat *negascout* menang 10 kali dan cyclog 2 kali. Sedangkan MTDF vs cyclog mendapat hasil MTDF menang 11 kali dan cyclog menang 1 kali [13].
4. Dikutip pada penelitian rancang bangun permainan othello menggunakan algoritma *Minimax* bahwa Algoritma ini merupakan algoritma jenis *depth first search* yang cocok diterapkan pada permainan yang terdiri dari dua pemain. Algoritma *minimax* mencari nilai maksimum dan minimum dari setiap langkah yang ada. Algoritma *minimax* diterapkan pada komputer untuk melawan manusia dalam mencari posisi yang mempertimbangkan nilai paling maksimum dari lawannya. Pada algoritma ini, pemain komputer akan menganalisis dari setiap kondisi pohon permainan sehingga nantinya komputer akan menentukan sebuah langkah yang mempunyai nilai keuntungan terbesar untuk dirinya dan mendapat nilai kerugian terbesar untuk lawannya. Namun cara kerja *minimax* berisifat rekursif yang mengakibatkan proses pencariannya berjalan lambat [14].
5. Terdapat penelitian yang dilakukan pada *Game Othello* dengan menggunakan algoritma *Iterative Deepening A Star*(IDA*). Algoritma IDA* adalah algoritma A* yang kedalamannya dibatasi pada setiap iterasi pencariannya. kompleksitas waktu IDA* lebih tinggi dibanding dengan Algoritma A*, tapi IDA* tidak membutuhkan memori yang sebesar dibanding Algoritma A*.

IDA* menggunakan persamaan fungsi *f(n)=g(n)+h(n). g(n)* adalah jarak sebenarnya dari simpul awal menuju suatu simpul *n.* Setiap level pada pohon evaluasi merepresentasikan jumlah langkah ke-i dari posisi awal, maka tingkat level tersebut merupakan jarak sebenarnya *g(n)*. Dan *h(n)* adalah jarak perkiraan dari simpul *n* (kotak legal *n* yang dipilih). Pengujian dilakukan pada ukuran papan 6x6 dengan menerapkan 3 iterasi. Iterasi ke-1 didapat kemungkinan langkah yang bisa dipilih adalah *f(n)*=14 dan *f(n)*= 16. Iterasi ke-2 didapat kemungkinan langkah yang bisa dipilih adalah *f(n)*=15 dan *f(n)*= 17. Dan iterasi ke-3 didapat kemungkinan langkah yang bisa dipilih adalah *f(n)*=16 dan *f(n)*= 18.

6. Dikutip dari jurnal Delpiah Wahyuningsih tentang perancangan sistem penjadwalan akademik STMIK Atma Luhur Pangkal bahwa terdapat permasalahan dalam mengatur jadwal diantaranya dosen yang mengajar harus sesuai dengan kelompok yang telah dibagikan, tidak dapat mengajar dalam satu waktu dari satu kelompok, tidak dapat mengajar dalam waktu yang sama dalam satu matkul, satu kelas yang sama tidak bisa dipakai lebih dari satu kelompok dalam waktu yang berbarengan dan kelompok berbeda tidak bisa ikut ke satu kelas di waktu yang berbarengan. Mengatasi permasalahan ini maka diterapkan algoritma *Max Min Ant System* dalam pembuatan jadwal akademik dengan paramater yang ditentukan terdapat 35 slot waktu dari 7 sesi yang dikali selama lima hari. Untuk pembentukan node nya diterapkan derajat penguapan *pheronome* sebesar 0.3, alfa sebagai nilai derajat kepentingan antara node yang berdekatan sebesar 0,9 beta 0,09, *max* siklus 1000 dan sebagai nilai penguapan sebesar 0.1. Maka didapat hasil 11.025 *node* pada penjadwalan akademik di STMIK Atma Luhur Pangkalpinang kemudian berhasil diterapkan dan menghasilkan jadwal yang optimal, tidak terjadi bentrokan antara matakuliah setiap kelompok sesinya dengan waktu yang efektif [3].
7. Menurut jurnal Rug-Tzuo Liaw dan kawan-kawan yang berjudul “Solving The Selevtive Pickup and Delivery Problem Using Max-Min Ant System” bahwa permasalahan dalam memilih pickup dan pengiriman dapat diatasi

dengan menggunakan MMAS dengan membangun rute terpendek dengan mempertimbangkan jumlah node pada pickup yang dipilih dan semua node pengiriman. Dan dilakukan pengujian kinerja MMAS yang dibandingkan dengan algoritma Genetik dan algoritma memetik dengan hasil kualitas MMAS lebih unggul dari algoritma Genetika pada segi kualitas solusi, dan dari kecepatan konvergensinya MMAS lebih cepat dari algoritma memetik [11].

8. Pada penelitian William dan kawan kawan mengenai algoritma *Max Min Ant System*, bahwa algoritma ini merupakan algoritma yang dibangun sebagai pendekatan alternatif dan perbaikan untuk algoritma koloni semut *(Ant Colony Optimization)*. Algoritma ini memiliki prinsip pengorganisasian yang mandiri seperti dilakukan pada semut sesungguhnya pada populasi *artifcial agent* untuk menyelesaikan masalah komputasional. Masalah penjadwalan dapat diselesaikan dengan menempatkan kombinasi dari kelas dan mata kuliah pada suatu *timeslot* dan ruang sehingga memenuhi *constraint* yang telah ditentukan. Terdapat dua jenis *constraint* yang berfungsi sebagai pembatas suatu keluaran solusi, sekaligus sebagai fungsi evaluasi dari solusi yang dihasilkan. *Constraint* pertama disebut *hard constraint* yaitu *constraint* bersifat tegas atau tidak boleh dilanggar. Kedua *soft constraint* yang digunakan untuk menyatakan tingkat keseimbangan suatu solusi. Pengujian dilakukan pada penjadwalan Program Studi Teknik Informatika UMN untuk semester genap. Pengujian dilakukan terbagi menjadi 3 kategori berdasarkan jumlah ruangan yang disediakan, yaitu kategori besar, sedang, dan kecil. Jumlah dan spesifkasi ruangan yang disediakan, yaitu ruangan untuk teori dan praktik. Dari hasil pengujian Implementasi algoritma yang dilakukan belum mampu menghasilkan solusi yang optimal, yaitu saat nilai *unplace* dan SCP sama dengan nol. Yang mengakibatkan jadwal yang seimbang belum dapat dicapai [15].
9. Penelitian Erick Alfons L dan Phie Chyan yang menerapkan algoritma semut untuk memecahkan dari permasalahan permainan *slidding Puzzle* bahwa pada representasinya papan berbentuk matriks satu dimensi. Inisialisasi feromon

berkaitan pada suatu tabel yang dinamakan pergerakan dari bentuk indeks matriks. Tabel ini mempunyai nilai banyak gerakan tertinggi yang diperoleh dari indeks berbentuk kotak tersebut serta ada kemungkinan beberapa strategi pergerakan bisa dilakukan. Banyaknya semut di setiap tahapan iterasi memiliki sifat dinamis yaitu bergantung pada banyaknya gerakan tertinggi yang diperbuat dari indeks perkotak. Kemudian ada suatu aturan kotak pergerakan diperbaiki dengan memanfaatkan set gerakan kombinasi dari sumber Finkelstein dan Markovitch. Yang bertujuan untuk mengatasi kondisi stagnasi perubahan pada feromon. Untuk penerapan algoritmanya dinyatakan berhasil dengan nilai feromon terbaik digunakan 0.15 dan 0.2 menghasilkan solusi dengan iterasi sedikit [16].