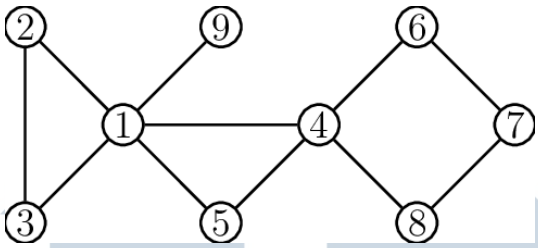
BAB 2
LANDASAN TEORI

Perkembangan sistem *multi-agent* untuk mendukung otomatisasi mendorong kajian mendalam terhadap *Multi-Agent Pathfinding* (MAPF), yaitu masalah pencarian jalur setiap agen untuk mencapai tujuan masing-masing tanpa konflik. Dalam praktiknya, lingkungan sering bersifat tidak pasti, sehingga muncul salah satu varian MAPF dengan *stochastic travel times* yang mempertimbangkan probabilitas *delay* dan ketidakpastian waktu pergerakan. Pendekatan penyelesaian MAPF terbagi menjadi metode berbasis *planning* dan *learning*, yang masing-masing memiliki kelebihan dan keterbatasan tergantung dengan dinamika lingkungan. Bagian ini menjadi dasar untuk pembahasan konsep, tantangan, dan metode dalam penelitian ini.

2.1 Multi-Agent Pathfinding

Secara umum, MAPF adalah tipe permasalahan *multi-agent planning* dengan fungsi untuk merencanakan jalur bagi para agen dalam suatu lingkungan, sehingga tidak terjadi konflik atau tabrakan antar agen [1]. Seperti terlihat pada Gambar 3.2, dalam permasalahan MAPF, sekumpulan agen sebanyak A bergerak pada suatu lingkungan yang dimodelkan sebagai graf $G = (V, E)$ dengan notasi sebagai berikut:

1. $G = (V, E)$ merepresentasikan model lingkungan tempat agen bergerak. Sebuah G terdiri dari sekumpulan V (*vertices*) dan E (*edges*).
2. V merupakan himpunan *vertices* yang merepresentasikan posisi atau *state* yang dapat ditempati oleh agen. Setiap V hanya dapat menampung suatu agen dalam suatu waktu.
3. E merupakan himpunan *edges* yang menunjukkan koneksi antar *vertex*. Suatu V dapat memiliki lebih dari satu E yang menunjukkan berbagai jalur yang dapat dilewati oleh agen.



Gambar 2.1. Visualisasi graf untuk *Multi-Agent Pathfinding*

Stern et al. [1] mendefinisikan *classical* MAPF dimana terdapat agen sebanyak k berupa array $[a_1,...,k]$ dengan karakteristik lingkungan yang berupa *tuple* $\langle G,s,g\rangle$ dengan s dan g berupa *array* $[s_1,...,k]$. Setiap agen a_i diasosiasikan dengan sepasang *state*/posisi yang berupa sumber s_i dan destinasi g_i. Setiap agen hanya dapat melakukan sebuah aksi dalam suatu waktu. Agen dapat menunggu di *vertex* tempat ia berada (*wait*), atau melangkah ke *vertex* lain yang terhubung melalui *edge* pada graf (*move*). Selama proses pergerakan, agen harus menghindari konflik dengan agen lain. Beberapa contoh konflik berupa situasi dimana lebih dari satu agen berada di suatu *vertex/edge* pada waktu yang sama (*vertex/edge conflict*) atau saling bertukar posisi pada waktu yang sama (*swapping conflict*). Berdasarkan pengetahuan bahwa kompleksitas jumlah *state* yang eksponensial terhadap jumlah agen, tingkat kesulitan pencarian solusi optimal dari permasalahan MAPF secara umum dikategorikan sebagai *NP-hard* [12].

Di samping hal itu, berbagai studi memiliki asumsi dan aturan yang berbeda dalam memodelkan permasalahan MAPF. Pada kasus *classical* MAPF versi Stern et al., solusi dideskripsikan sebagai π, dengan $\pi_i[t]$ menunjukkan posisi agen a_i setelah melakukan aksi pada waktu ke-t. Dalam penyelesaian masalah *classical* MAPF, solusi π tidak dapat memiliki konflik. Secara umum, agen a_i dan a_j dikatakan konflik pada waktu ke-t jika:

$$(\pi_{a_i}[t] = \pi_{a_j}[t]) \vee \big((\pi_{a_i}[t] = \pi_{a_j}[t+1]) \wedge (\pi_{a_i}[t+1] = \pi_{a_j}[t])\big) \tag{2.1}$$

Suatu solusi π yang tidak memiliki konflik akan dikategorikan sebagai *valid plan*. Namun demikian, tidak semua solusi yang valid memiliki kualitas yang sama. Dalam penilaian kualitas solusi dalam *classical* MAPF, dapat digunakan fungsi *makespan* dan *sum of costs* (SOC). Fungsi *makespan* digunakan untuk menghitung jumlah langkah waktu yang dibutuhkan hingga seluruh agen mencapai lokasi tujuan

masing-masing, yang secara matematis dapat dinotasikan sebagai:

$$\text{Makespan}(\pi) = \max_{i=1}^{n} |\pi_i| \tag{2.2}$$

Sementara itu, fungsi *sum of costs* (SOC) didefinisikan sebagai jumlah total langkah waktu yang dibutuhkan oleh seluruh agen untuk mencapai tujuan mereka, yang secara umum dirumuskan sebagai:

$$\text{SOC}(\pi) = \sum_{i=1}^{n} |\pi_i| \tag{2.3}$$

Pada permasalahan MAPF secara umum, setiap agen hanya disandingkan dengan satu tugas, yakni untuk mencari jalur dari sebuah titik awal sampai destinasi. Untuk meningkatkan kompleksitas permasalahan, Ma et al. [13] memperkenalkan model MAPF yang mempertimbangkan kemunculan tugas atau target baru secara konstan yang dikenal sebagai *Life-long Multi-Agent Pathfinding* (LMAPF). Pada model ini, agen beroperasi dalam lingkungan di mana tugas muncul secara berkelanjutan sepanjang waktu sampai operasi dihentikan. Himpunan tugas yang belum dieksekusi didefinisikan sebagai T. Setiap tugas $\tau_j \in T$ didefinisikan oleh titik awal $s_j \in V$ dan titik destinasi $g_j \in V$. Agen yang tidak sedang menjalankan tugas disebut *free*, sedangkan agen yang sedang menjalankan tugas disebut *occupied*. Agen *free* dapat ditugaskan untuk suatu tugas baru dengan bergerak menuju titik awal s_j dan kemudian menuju titik destinasi g_j. Tujuan sistem adalah menyelesaikan setiap tugas secepat mungkin dengan mempertimbangkan rata-rata langkah waktu yang dibutuhkan.

Model permasalahan MAPF klasik dan LMAPF menjadi dasar dari perkembangan berbagai variasi MAPF yang didasarkan terhadap tujuan dan objektif masing-masing. Sampai saat ini, metode pendekatan dalam pencarian solusi optimal untuk permasalahan model MAPF klasik maupun LMAPF masih menjadi perbincangan hangat. Kompleksitas permasalahan MAPF klasik meliputi waktu komputasi yang tinggi akibat pertumbuhan ruang pencarian yang juga selaras dengan kebutuhan menjamin solusi bebas konflik. Hal ini menimbulkan *trade-off* antara optimalitas dan efisiensi. Kompleksitas pun semakin rumit dalam kasus LMAPF dikarenakan tugas yang dinamis menuntut perencanaan ulang secara terus-menerus.

2.2 Stochastic MAPF

Permasalahan MAPF secara umum dirancang dalam ruang lingkup yang deterministik, dimana setiap aksi hanya menghasilkan sebuah hasil yang dapat diprediksi [2] [3]. Namun, pada praktek dunia nyata umumnya lingkungan menunjukkan perilaku yang *stochastic* dan tidak pasti. Berdasarkan studi dari Levy et al. [14], permasalahan MAPF pada *stochastic environment* dapat direpresentasikan sebagai *tuple* $\langle G, n, S, T, P, C \rangle$, dimana $G = (V, E)$ sebagai graf yang menggambarkan lingkungan eksperimen; n sebagai jumlah agen; $S = \langle s_0, \ldots, s_n \rangle$ sebagai himpunan titik awal; $T = \langle t_0, \ldots, t_n \rangle$ sebagai himpunan titik destinasi; Fungsi $P(v, a, v')$ sebagai fungsi transisi *stochastic* untuk menyatakan probabilitas suatu agen mencapai v' ketika melakukan aksi a saat di titik v, serta $C(v, a)$ sebagai fungsi untuk menyatakan biaya suatu agen ketika melakukan aksi a pada titik v.

Terdapat beberapa jenis pengaturan yang dapat diimplementasikan dalam model permasalahan *stochastic* MAPF, salah satunya adalah adanya *probabilistic delay*. Model MAPF dengan *probabilistic delay* di setiap *node* ini dapat direferensikan sebagai MAPF dengan ketidakpastian waktu tempuh atau *stochastic travel times* (STT-MAPF). Atzmon et al. [15] merepresentasikan *delay* dalam waktu yang diskrit, sebagai $\langle \pi, a_i, t \rangle$ dengan $t \geq 1$, yang menyatakan bahwa agen a_i tetap berada pada lokasi $\pi_i(t-1)$, dibandingkan melakukan aksi perpindahan dari lokasi $\pi_i(t-1)$ pada waktu $t-1$ menuju lokasi $\pi_i(t)$ pada waktu t. Suatu solusi π dikatakan *robust* terhadap *stochastic travel times* apabila probabilitas terjadinya konflik selama eksekusi tidak melebihi *threshold* yang telah ditetapkan, sehingga agen tetap dapat melanjutkan rencana pergerakan individualnya dengan tingkat risiko yang dapat diterima.

Dibandingkan dengan asumsi waktu diskrit, di mana *delay* direpresentasikan sebagai kejadian biner pada setiap *time-step*, studi lebih lanjut menggunakan pendekatan waktu kontinu dengan model *delay* sebagai variabel acak yang secara langsung memengaruhi durasi perjalanan. Pada pendekatan yang ditetapkan oleh Peltzer et al. [16], *delay* di *node* v yang ditempuh agen i dimodelkan sebagai $\tau_{v,i} \sim \text{Gamma}(n_v, \lambda)$. Waktu tempuh kumulatif sepanjang jalur dimodelkan sebagai distribusi Gamma, yang merupakan hasil agregasi dari parameter *delay* pada setiap segmen lintasan. Secara matematis, distribusi waktu tempuh kumulatif untuk agen i hingga langkah ke-k dapat terlihat pada Rumus 2.4.

$$\Delta_i(k) \sim \text{Gamma}\left(\sum_{j=0}^{k} n_{p_i(j)}, \lambda\right) \tag{2.4}$$

di mana $\sum_{j=0}^{k} n_{p_i(j)}$ merepresentasikan akumulasi parameter sepanjang lintasan yang dilalui agen, dan λ adalah parameter laju (*rate*) dari distribusi Gamma. Untuk menghitung total *delay* pada suatu waktu tertentu bisa dilakukan dengan menghilangkan indeks k dan mendefinisikan $n_1 = \sum_{j=0}^{k} n_{p_1(j)}$ dan $n_2 = \sum_{j=0}^{k} n_{p_2(j)}$, dimana *delay* kumulatif untuk masing-masing agen dinyatakan sebagai $\Delta_1 \sim \text{Gamma}(n_1, \lambda)$ dan $\Delta_2 \sim \text{Gamma}(n_2, \lambda)$.

Dengan mempertimbangkan waktu nominal t_1, t_2 serta *delay* kumulatif $\Delta_1 \sim \text{Gamma}(n_1, \lambda)$ dan $\Delta_2 \sim \text{Gamma}(n_2, \lambda)$, maka konflik pada *node* dan *edge* dinotasikan sebagai terlihat pada Rumus 2.5 dan Rumus 2.6.

$$\text{Conflict}_{\text{node}} \iff \begin{cases} \Delta_1 - \Delta_2 \geq t_2 - t_1 - t_e & (A) \\ \Delta_1 - \Delta_2 \leq t_2 - t_1 + t_e & (B) \end{cases}. \tag{2.5}$$

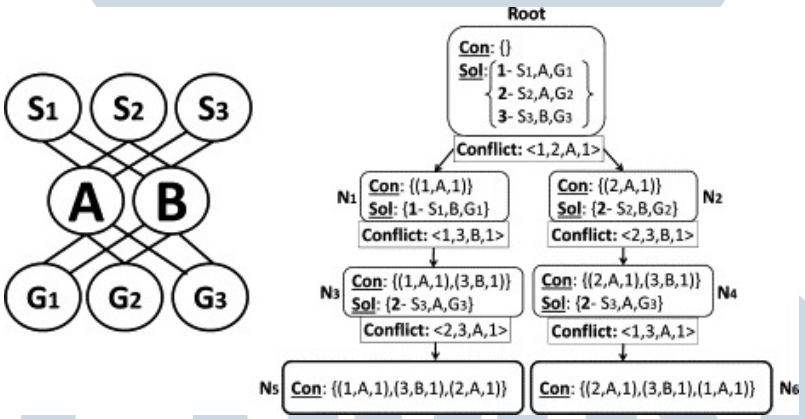
$$\text{Conflict}_{\text{edge}} \iff \begin{cases} \Delta_1 \geq \Delta_2 + t_2 - t_1 - t_e & (A) \\ \Delta_1 \leq \Delta_2 + t_2 - t_1 + t_e & (B) \end{cases}. \tag{2.6}$$

Probabilitas konflik pada suatu *node* dan edge pun masing-masing dapat dinotasikan dengan fungsi $P_{cv}(t_1 - t_2, n_1, n_2, \lambda, n_v)$ dan $P_{cv}(t_1 - t_2, n_1, n_2, \lambda, n_v)$, dengan kombinasi probabilitas sebagai $P_c = 1 - (1 - P_{cv})(1 - P_{ce})$.

2.3 Metode Planning vs Learning

Berbagai jenis metode, baik yang bersifat *planning* dan *learning* telah digunakan untuk menyelesaikan permasalahan MAPF dengan asumsi yang berbeda-beda. Secara umum, efisiensi dan performa dari suatu algoritma bergantung terhadap asumsi yang ditetapkan [6]. Namun dari berbagai studi sebelumnya, metode *planning-based* umumnya digunakan dengan asumsi bahwa kondisi lingkungan sepenuhnya dapat diamati serta dikendalikan secara terpusat/*centralized* [3] [17], sedangkan metode *learning-based* lebih difokuskan pada kasus lingkungan yang *decentralized* dan/atau *partially observable* [7] [18] [19].

Dalam penyelesaian permasalahan MAPF yang termasuk *NP-hard*, pendekatan dengan algoritma berbasis heuristik menjadi standar utama. Salah satu metode yang menjadi *state-of-the-art* yakni algoritma *Conflict-based Search* (CBS), yang bersifat *planning-based*. Algoritma CBS pertama dikemukakan oleh Sharon et al. [3] dengan tujuan mencari solusi optimal dari *classical* MAPF. CBS bekerja dengan terlebih dahulu merencanakan lintasan setiap agen secara individual, umumnya dengan algoritma *A-star* (A*), kemudian mendeteksi konflik (tabrakan) yang terjadi. Ketika konflik terdeteksi, keberadaan konflik akan disimpan dalam struktur data berbentuk *tree* yang disebut sebagai *constraint tree* (CT) seperti terlihat pada Gambar 2.2, kemudian perencanaan ulang lintasan setiap agen yang konflik dilakukan secara iteratif hingga diperoleh solusi yang bebas konflik. Algoritma CBS meningkatkan performa dengan cara meminimalisir ruang eksplorasi untuk pencarian jalur setiap agen secara individual dengan waktu komputasi yang linier terhadap ukuran graf.

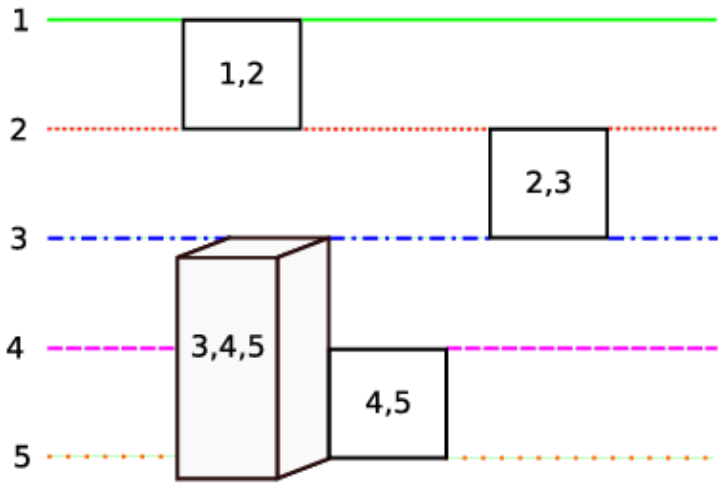


Gambar 2.2. Struktur *constraint tree* pada *Conflict-based Search*

Sumber: [5]

Berbagai studi telah memodifikasi algoritma CBS berdasarkan target yang berbeda-beda. Dibandingkan dengan model Sharon et al. yang memfokuskan terhadap meminimalisir *sum of costs*, model Barer et al. [20] memfokuskan dengan pencarian solusi *sub-optimal* dengan biaya komputasi minimal. Di sisi lain, studi dari Felner et al. [21] menunjukkan potensi dari variasi algoritma CBS dalam mencari solusi optimal dengan efisiensi komputasi yang tinggi. Selain itu, beberapa studi telah menciptakan variasi dari CBS yang disesuaikan dengan lingkungan tertentu, seperti Andreychuk et al. [22] yang memperkenalkan algoritma CBS lanjutan dikhususkan pada pencarian solusi di permasalahan MAPF dengan asumsi waktu yang bersifat kontinu.

Namun pendekatan penyelesaian MAPF tidak didasarkan hanya pada konsep algoritma CBS. Sebelum kemunculan algoritma CBS, Wagner et al. [23] mengemukakan pendekatan melalui algoritma M*, yang merupakan versi lanjutan dari algoritma A* dengan penekanan terhadap *subdimensional expansion*, dimana secara *default*, perencanaan jalur agen dilakukan secara individual, sedangkan permasalahan MAPF hanya dipraktikkan pada sekumpulan agen yang saling konflik selama proses pencarian, dengan tujuan meminimalisir ruang eksplorasi sesuai kebutuhan, contohnya dapat dilihat pada Gambar 2.3. Dalam upaya optimasi algoritma M*, berbagai variasi dari M* telah dikembangkan untuk meningkatkan efisiensi, contohnya *Recursive* M* [23] dan ODrM* [24].



Gambar 2.3. Lingkungan dengan 5 agen terbagi menjadi 4 submasalah MAPF sederhana melalui proses *subdimensional expansion*

Sumber: [23]

Dalam konteks permasalahan MAPF dengan mempertimbangkan ketidakpastian waktu tempuh, perkembangan variasi dari kedua algoritma CBS dan M* masih sangat aktif diteliti. Peltzer et al. [16] dan Wagner et al. [25] masing-masing merumuskan model algoritma *Stochastic Travel Times Conflict-based Search* (STT-CBS) dan *Uncertainty* M* (UM*) yang berfungsi meminimalisir kemungkinan terjadinya konflik antar agen dengan menghitung probabilitas konflik dan membatasi nilainya agar tetap berada di bawah *threshold* yang telah ditetapkan. Hal ini menunjukkan relevansi dari metode *planning-based* dalam menawarkan solusi yang kompetitif.

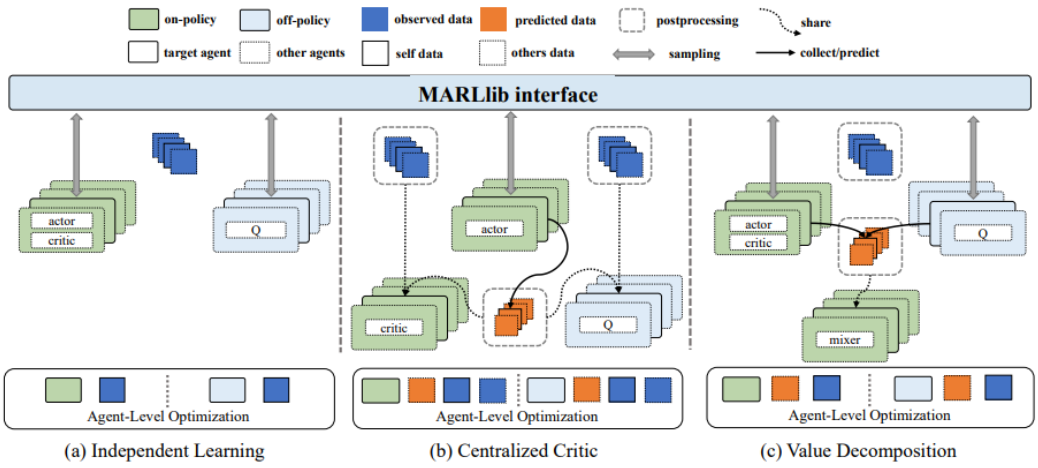
Namun seiring dengan perkembangan teknologi pembelajaran mesin, metode *learning-based* dengan pemanfaatan teknik pembelajaran mesin mulai

diterapkan dalam permasalahan MAPF, dengan tujuan untuk meningkatkan fleksibilitas dan adaptasi terhadap lingkungan yang kompleks. Pendekatan ini mendorong mesin untuk mempelajari fungsi heuristik dan secara langsung menentukan kebijakan pergerakan agen melalui *reinforcement learning* dan/atau *imitation learning*. Proses pengambilan keputusan agen dimodelkan sebagai *Markov Decision Process* (MDP), dimana pemilihan aksi dilakukan berdasarkan kebijakan yang telah dipelajari. Salah satu pelopor utama dalam penerapan metode *learning-based* adalah perancangan strategi bernama *Pathfinding via Reinforcement and Imitation Multi-Agent Learning* (PRIMAL) yang dikembangkan oleh Sartoretti et al. [8]. PRIMAL menggunakan fungsi *reward* yang kompleks serta menerapkan berbagai asumsi yang bergantung pada kondisi dan peta tertentu (*domain knowledge*). Dalam perkembangan metode *learning-based* dalam skenario yang lebih luas, Damani et al. [26] merancang metode lanjutan dari PRIMAL bernama PRIMAL2 yang memungkinkan koordinasi lebih lanjut antar agen dalam penyelesaian *lifelong* MAPF.

Lebih lanjut, dalam mempraktekkan metode *learning-based* dalam lingkungan stokastik, Wu et al. [7] merancang model berbasis *Multi-Agent Reinforcement Learning* (MARL). Dalam pendekatan ini, setiap agen belajar secara simultan melalui interaksi dengan lingkungan untuk memaksimalkan *reward* kumulatif, dengan mempertimbangkan keadaan dan tindakan agen lain. Secara umum, proses pembelajaran dalam MARL umumnya terbagi menjadi beberapa kategori, yaitu *Independent Learning* (IL), *Centralized Critic* (CC), dan *Value Decomposition* (VD) [27].

Pada pendekatan IL, setiap agen mempelajari kebijakannya secara mandiri dengan menganggap agen lain sebagai bagian dari lingkungan, sehingga metode ini relatif sederhana namun rentan terhadap permasalahan *non-stationarity*, yaitu kondisi ketika perubahan kebijakan agen lain menyebabkan lingkungan terus berubah selama proses pelatihan. Sementara itu, pendekatan CC menerapkan paradigma *Centralized Training and Decentralized Execution* (CTDE), di mana proses pelatihan memanfaatkan informasi global seluruh agen melalui sebuah *critic* terpusat, tetapi pada tahap eksekusi setiap agen tetap mengambil keputusan berdasarkan observasi lokalnya masing-masing, sehingga koordinasi dan stabilitas pembelajaran dapat meningkat. Di sisi lain, pendekatan VD berfokus pada dekomposisi fungsi nilai global menjadi kombinasi fungsi nilai individual tiap agen, sehingga agen dapat belajar berkontribusi terhadap tujuan kolektif secara terdesentralisasi sekaligus membantu mengatasi permasalahan *credit assignment*,

yaitu menentukan kontribusi masing-masing agen terhadap *reward* tim secara keseluruhan. Gambar 2.4 menunjukkan struktur alur data pada library MARLlib, untuk setiap metode pembelajaran.



Gambar 2.4. Struktur alur data pada MARLlib untuk tiga kategori pembelajaran: *independent learning*, *centralized critic*, dan *value decomposition*

Sumber: [27]

Dengan adanya perkembangan metode *learning-based*, jenis pendekatan semakin bervariasi dan tidak hanya mengandalkan algoritma *planning-based*, termasuk dalam mencari solusi optimal dalam kasus MAPF di pengaturan stokastik yang masih dipelajari secara intensif. Secara singkat, pendekatan *planning-based* dan *learning-based* masing-masing menawarkan keunggulan yang berbeda sehingga diperlukan eksplorasi yang lebih lanjut.

2.4 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) karya Coulom et al. [28] merupakan algoritma yang menggabungkan *Monte Carlo evaluation* dengan *tree search*. *Monte Carlo evaluation* dilakukan untuk memperkirakan hasil dari suatu skenario berdasarkan rata-rata nilai dari berbagai simulasi acak (*Monte Carlo simulation*), yang kemudian digunakan sebagai fungsi evaluasi pada daun pohon *min-max*. Akurasi dari *Monte Carlo evaluation* menjadi kunci utama dalam mencari solusi optimal, salah satunya dalam permasalahan MDP. Pada algoritma MCTS, metode untuk meningkatkan akurasi *Monte Carlo evaluation* adalah dengan algoritma *tree search*.

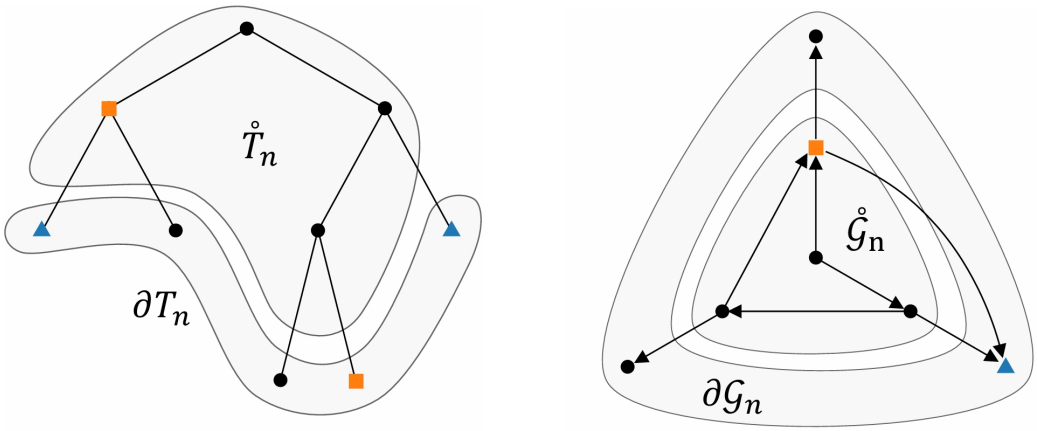
MCTS bekerja dengan cara membangun pohon pencarian secara iteratif melalui simulasi acak dari posisi awal (*root*). Setiap simulasi menghasilkan skenario yang digunakan untuk memperbarui informasi pada *node* yang dilalui. Berbeda dengan pendekatan *min-max* tradisional yang mengevaluasi seluruh cabang pohon secara eksplisit, MCTS menggunakan simulasi *Monte Carlo* untuk memperkirakan nilai suatu state berdasarkan hasil rata-rata simulasi yang telah dilakukan. Melalui mekanisme seleksi seperti *Upper Confidence Bound for Trees* (UCT) yang dikemukakan oleh Kocsis et al. [29], MCTS mampu menyeimbangkan eksplorasi terhadap langkah yang belum banyak dicoba (*exploration*) dan eksploitasi terhadap langkah yang memiliki rata-rata *reward* tinggi (*exploitation*). Node dengan performa baik akan lebih sering dipilih, sementara node yang masih jarang dikunjungi tetap diberikan kesempatan untuk dieksplorasi, sehingga kualitas keputusan dapat meningkat seiring bertambahnya jumlah simulasi.

Secara umum, pendekatan MCTS modern terdiri dari empat tahap utama, yaitu *selection*, *expansion*, *simulation*, dan *backpropagation* yang dilakukan secara iteratif hingga batas waktu atau jumlah iterasi tertentu tercapai. Pada tahap *selection*, algoritma menelusuri pohon pencarian dari *root node* menuju *child node* menggunakan kebijakan seperti *Upper Confidence Bound for Trees* (UCT) untuk menyeimbangkan eksplorasi dan eksploitasi. Selanjutnya, tahap *expansion* dilakukan dengan menambahkan *node* baru berdasarkan aksi yang tersedia pada *node* terpilih. *Node* hasil *expansion* kemudian digunakan pada tahap *simulation* atau *rollout*, yaitu proses simulasi menggunakan kebijakan acak maupun heuristik untuk memperoleh estimasi *reward*. Proses *rollout* akan menghasilkan informasi yang selanjutnya digunakan tahap *backpropagation* dengan memperbarui statistik *node* seperti jumlah kunjungan (*visit count*) dan total *reward*, sehingga kualitas estimasi nilai pada pohon pencarian meningkat seiring bertambahnya jumlah simulasi.

Dalam konteks *multi-agent exploration* berdasarkan model yang dikembangkan oleh Bone et. al [10], pendekatan MCTS dapat diterapkan secara *decentralized* dengan cara memodifikasi kerangka MCTS dengan memungkinkan setiap agen melakukan perencanaan secara independen, dibandingkan dengan pendekatan terpusat yang memerlukan perencanaan global dengan hasil berupa *joint action* untuk seluruh agen. Berdasarkan model ini, koordinasi dicapai dengan cara mengatur agen untuk saling berbagi informasi tentang lingkungan dan rencana masing-masing agen.

2.5 Monte Carlo Graph Search

Monte Carlo Graph Search (MCGS) karya Leurent et al. [11] merupakan improvisasi dari algoritma *Monte Carlo Tree Search* (MCTS) yang menjadi inovasi dalam permasalahan *Markov Decision Process* (MDP). Walaupun MCTS yang berbasis *tree-search* terlihat simpel dan efektif, terdapat limitasi berupa ketidakmampuan algoritma dalam memperhatikan kesamaan dari berbagai jalur. Studi oleh Ballesteros et al. [30] mengusulkan ide menggabungkan informasi antara *branch* sebuah *tree* dengan cara mengidentifikasi *state* yang identik dan membuat rencana dalam bentuk graf. State yang identik dikenal sebagai transposisi. Perbedaan dari struktur data berbentuk *tree* dan *graph* dapat dilihat pada Gambar 2.5.



Gambar 2.5. Struktur data *tree vs graph*
Sumber: [11]

Sebuah *tree* dillustrasikan sebagai sebuah *node* dengan kedalaman h yang merepresentasikan suatu urutan aksi $a \in A^h$. *Root* dari *tree* berkaitan dengan keadaan awal $s_0 \in S$, yang dilanjutkan dengan *branch* dibawahnya sebagai langkah aksi yang diterapkan. Pada iterasi ke-n, *tree* dinyatakan sebagai T_n. Himpunan *node* internal dinotasikan sebagai $\mathring{T}_n$, sedangkan himpunan *node* eksternal dinotasikan sebagai ∂T_n. Sebaliknya, graf pada iterasi ke-n dinyatakan sebagai $\mathcal{G}_n$, himpunan *node* internalnya sebagai $\mathring{\mathcal{G}}_n$, dan himpunan *node* akhirnya (*sink*) sebagai $\partial \mathcal{G}_n$.

Pada suatu *tree*, urutan aksi a memiliki asosiasi dengan keadaan akhirnya yang dinotasikan sebagai $\delta(a)$, namun asosiasi ini tidak bersifat unik, dimana beberapa langkah aksi dapat menghasilkan keadaan yang sama, yang akan

direpresentasikan beberapa kali di dalam *tree*. Sedangkan pada graf, *node* merepresentasikan *state* $s \in S$ pada suatu lingkungan, dimana tiap *node* memiliki *pointer* yang mengarah ke *parent state* serta *child state*. Dengan demikian, jika suatu aksi a yang diterapkan pada *state* s menghasilkan *next state* s' yang telah hadir di $\mathcal{G}_n$, sebuah *pointer* dapat ditambahkan antara s dengan s' dibandingkan dengan membuat *node* baru, seperti dalam struktur data *tree*. Penggabungan *similar state* atau transposisi memungkinkan statistik hasil simulasi dari keadaan yang identik untuk dibagikan dan dimanfaatkan kembali oleh beberapa jalur pencarian yang melewati *node* tersebut. Dengan menggabungkan statistik simulasi dari *state* yang identik, *state merging* menghasilkan estimasi nilai *node* yang lebih akurat, meningkatkan efisiensi pemanfaatan *simulation budget*, serta berpotensi menurunkan *regret bound* dibandingkan pencarian yang memperlakukan setiap kemunculan *state* sebagai *node* yang berbeda.

2.6 Research Gap

Dari kajian teoritis, berbagai studi menunjukkan potensi eksplorasi lebih lanjut pada permasalahan MAPF, khususnya dalam konteks lingkungan stokastik atau tidak pasti. Beberapa algoritma *planning-based* yang telah diuji termasuk STT-CBS yang berbasis algoritma CBS serta UM* yang berbasis algoritma M*, di mana secara harfiah, bekerja dengan menyederhanakan permasalahan dengan meminimalisir ruang eksplorasi melalui kriteria yang ditetapkan. Namun, metode ini umumnya bersifat *centralized* yang memerlukan koordinator pusat untuk mengontrol setiap agen. Di sisi lain, terdapat pendekatan *learning-based* seperti MARL yang memanfaatkan *reinforcement learning* untuk mempelajari lingkungan, kemudian secara implisit mencari solusi berdasarkan pengalaman yang diperoleh selama proses pelatihan. Contoh penelitian-penelitian terdahulu terkait penyelesaian masalah MAPF pada lingkungan stokastik terlihat pada Tabel 2.1.

Tabel 2.1. Perbandingan metode penelitian terdahulu

Judul Penelitian	Metodologi	Hasil Utama	Research Gap
Path Planning for Multiple Agents Under Uncertainty (Wagner et. al, 2017)	M*	Berbagai algoritma varian UM* menghasilkan perencanaan jalur multi-agent yang optimal dan *robust* terhadap konflik dalam permasalahan MAPF dengan pendekatan *belief-space planning* atau disebut sebagai MAPFU.	Metode ini memiliki keterbatasan karena bergantung pada informasi global dan rentan mengalami peningkatan *search space* yang signifikan seiring bertambahnya konflik. Selain itu, penelitian tersebut menggunakan model MAPF yang belum secara spesifik diterapkan pada lingkungan dengan pengaturan *probabilistic delay* berdasarkan suatu distribusi.
STT-CBS: A Conflict-Based Search Algorithm for Multi-Agent Path Finding with Stochastic Travel Times (Peltzer et. al, 2021)	Conflict-based Search	Algoritma STT-CBS menghasilkan solusi yang *robust* terhadap konflik hingga batas probabilitas atau *threshold* tertentu, meskipun waktu komputasi meningkat seiring ketatnya *threshold* yang ditetapkan.	Metode ini bergantung pada informasi global dan tidak terdesentralisasi. Oleh karena itu, penelitian selanjutnya dapat menerapkan serta membandingkan metode berbasis *planning* maupun *learning* lainnya pada lingkungan dengan pengaturan yang lebih spesifik.
Multi-Agent Reinforcement Learning-Based UAV Pathfinding for Obstacle Avoidance in Stochastic Environment (Wu et. al, 2023)	MARL-based algorithm	Metode MARL dengan pendekatan *partially decentralized execution* (CTPDE) mampu merencanakan jalur yang aman bagi setiap agen dalam kondisi keterbatasan komunikasi. Setiap agen pun mampu merencanakan jalur aman berdasarkan *policy* tanpa bergantung pada perencana terpusat.	Kinerja metode sangat bergantung pada kualitas proses pelatihan, desain *reward*, serta kemampuan generalisasi *policy* pada lingkungan yang berbeda. Selain itu, pengaturan MAPF dengan keterbatasan komunikasi masih menghadapi tantangan dalam menjaga koordinasi dan menghindari konflik antar agen secara desentralisasi pada lingkungan stokastik sekaligus mencari solusi optimal, sehingga topik ini masih memerlukan penelitian lebih lanjut.

Terlihat bahwa terdapat limitasi yang cukup jelas antara pendekatan berbasis *planning* dan *learning*. Algoritma *planning-based* umumnya bergantung pada asumsi lingkungan yang terpusat (*centralized planning*). Asumsi ini seringkali tidak realistis dalam skenario dunia nyata dengan lingkungan yang tidak dapat dikontrol secara terpusat. Di sisi lain, metode *learning-based* seperti MARL mampu beroperasi dengan paradigma CTDE, namun tidak selalu menjamin solusi yang optimal serta bergantung pada kualitas pelatihan dan kebijakan yang dipelajari.

Dengan mempertimbangkan keunggulan dan limitasi kedua metode, penelitian ini bertujuan untuk memanfaatkan metode *Monte Carlo Graph Search* (MCGS) sebagai algoritma pengambilan keputusan dalam paradigma *decentralized planning* untuk mencari solusi pada permasalahan *Multi-Agent Path Finding*

(MAPF) dengan *stochastic travel times*. Dalam pendekatan ini, setiap agen melakukan proses perencanaan secara mandiri berdasarkan informasi yang tersedia tanpa memerlukan koordinator terpusat. Penerapan MCGS dalam paradigma *decentralized planning* selanjutnya disebut sebagai *Decentralized Monte Carlo Graph Search* (Dec-MCGS). Perbandingan metode yang diterapkan dapat dilihat pada Tabel 2.2.

Tabel 2.2. Matriks perbandingan metode pada MAPF dengan *stochastic travel time*

No	Aspek / Fitur Utama	STT-CBS / UM*	MARL	Dec-MCGS
1.	*Stochastic-aware*	✓	✓	✓
2.	*Planning-based*	✓	×	✓
3.	*Decentralized*	×	✓	✓