

Project Proposal
Mercusuar
Agen Permainan Berbasis Minimax dengan Alpha-Beta Pruning



INFORMATICS

Amadeus Eugene Dirgantara (0706012410063)
adirgantara@student.ciputra.ac.id

Universitas Ciputra Surabaya
Oktober 2026

Daftar Isi

Daftar Isi.....	2
1. Pendahuluan.....	3
1.1 Latar Belakang.....	3
1.2 Rumusan Masalah.....	4
2. Pendekatan yang Diusulkan.....	5
2.1 Metode yang Diusulkan.....	5
Aturan Permainan.....	5
Algoritma.....	6
Kompleksitas Waktu dan Skalabilitas.....	6
Fungsi Evaluasi.....	9
Tingkat Kesulitan.....	10
PEAS.....	11
Sifat Lingkungan.....	11
Input dan Output.....	11
Arsitektur Software.....	12
2.2 Metode Evaluasi.....	15
2.3 Jadwal Pelaksanaan.....	17
3. Glosarium.....	18
Riwayat Revisi.....	18
Istilah.....	18
4. Daftar Pustaka.....	19

1. Pendahuluan

Proposal ini mengusulkan pembuatan Mercusuar, permainan papan berbasis web antara satu pemain manusia dan agen kecerdasan buatan. Agen memilih langkah menggunakan algoritma minimax dengan alpha-beta pruning dan menyediakan lima tingkat kesulitan. Permainan dirancang agar satu sesi selesai dalam waktu singkat sehingga dapat digunakan sebagai pos permainan dalam kegiatan rally.

1.1 Latar Belakang

Mercusuar adalah permainan papan dua pemain pada papan 7×7 . Kedua pemain bergantian membangun mercusuar di sel kosong. Setiap mercusuar menyinari baris dan kolomnya ke empat arah sampai cahayanya terhalang mercusuar lain. Sel yang hanya diterangi cahaya seorang pemain menjadi milik pemain tersebut, dan setelah semua mercusuar dibangun, pemain dengan sel terbanyak menang.

Aturannya singkat, tetapi satu langkah dapat mengubah pemilik banyak sel sekaligus. Mercusuar baru menerangi sel kosong dan sekaligus menghalangi cahaya lawan, sehingga sel milik lawan dapat menjadi tidak dimiliki atau berpindah pemilik. Untuk memilih langkah yang baik, pemain perlu memperhitungkan balasan lawan beberapa langkah ke depan.

Hasil permainan hanya ditentukan oleh langkah kedua pemain. Seluruh papan terlihat oleh kedua pemain, dan hasil setiap langkah dapat dihitung. Jumlah mercusuar setiap pemain tetap, sehingga satu permainan selalu berakhir setelah 14 langkah.

Domain ini dipilih karena permainan antara manusia dan komputer merupakan contoh langsung dari adversarial search yang dibahas dalam mata kuliah ini [1]. Jumlah langkah yang tetap dan papan yang kecil membuat durasi permainan dapat dibatasi, sehingga permainan dapat digunakan sebagai pos rally.

1.2 Rumusan Masalah

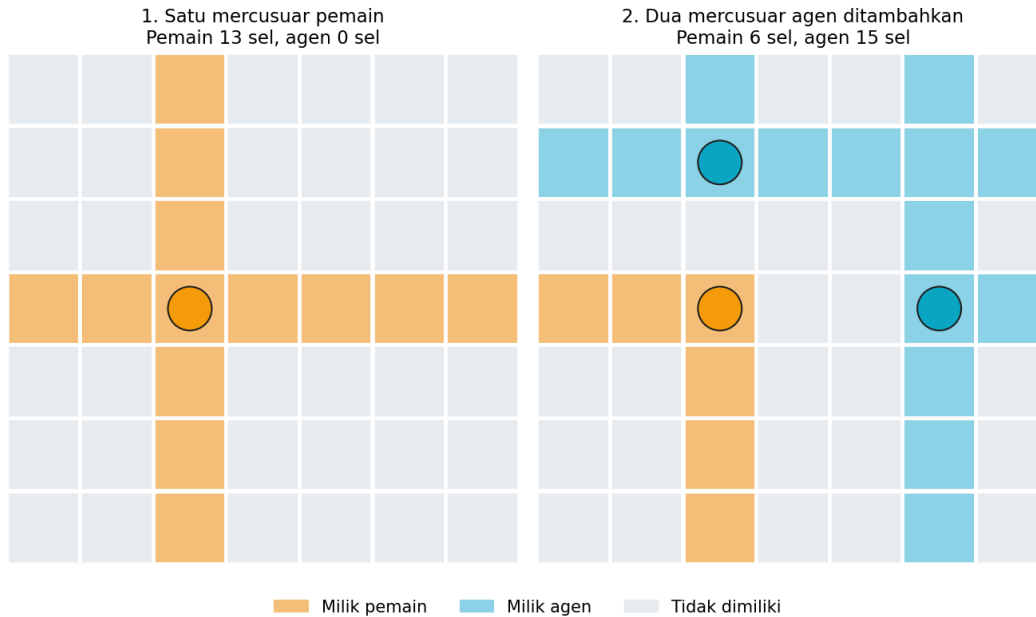
Diberikan state permainan yang terdiri atas posisi mercusuar kedua pemain pada papan 7×7 dan giliran pemain, agen harus memilih sel kosong untuk membangun mercusuar yang menghasilkan nilai terbaik bagi dirinya dengan asumsi lawan manusia selalu memilih balasan terbaik.

2. Pendekatan yang Diusulkan

2.1 Metode yang Diusulkan

Aturan Permainan

- Papan berukuran 7×7 sel, dan setiap pemain memiliki 7 mercusuar. Satu langkah berupa membangun satu mercusuar di sel kosong. Agen membangun lebih dahulu, lalu agen dan pemain manusia bergantian, sehingga pemain manusia membangun mercusuar terakhir. Pembangun terakhir dapat menghalangi cahaya lawan tanpa dapat dibalas, sehingga giliran terakhir diberikan kepada pemain manusia.
- Mercusuar menyinari sel di baris dan kolomnya ke empat arah. Cahaya berhenti pada mercusuar berikutnya, milik pemain mana pun (Gambar 1).
- Sel yang berisi mercusuar milik pembangunnya. Sel kosong milik seorang pemain jika hanya cahaya pemain tersebut yang mencapainya. Sel kosong yang diterangi kedua pemain atau tidak diterangi sama sekali tidak dimiliki siapa pun.
- Pemilik setiap sel dihitung ulang setelah setiap langkah. Permainan berakhir setelah 14 mercusuar dibangun. Pemain dengan sel lebih banyak menang, dan jumlah sel yang sama menghasilkan seri.
- Pemain manusia memiliki timer total 4 menit yang hanya berjalan saat permainan menunggu langkahnya. Jika waktu habis, pemain manusia kalah.
- Pemain manusia dan agen masing-masing melangkah 7 kali.



Gambar 1. Cahaya mercusuar dan pemilik sel

Algoritma

Agen akan menggunakan algoritma minimax [1], [3]. Algoritma ini membangun game tree dari state saat ini. Agen memilih langkah dengan nilai tertinggi, dengan asumsi pemain manusia memilih langkah yang memberi nilai terendah bagi agen.

Alpha-beta pruning [2] diterapkan untuk melewati cabang yang terbukti tidak dapat mengubah keputusan, sehingga hasil keputusan sama dengan minimax biasa dengan jumlah node yang diperiksa lebih sedikit. Pada setiap pencarian, langkah terbaik dari kedalaman sebelumnya diperiksa lebih dahulu, lalu sel lain diperiksa berurutan dari tengah papan ke tepi.

Setiap langkah adalah satu sel kosong, sehingga pada awal permainan setiap node memiliki 49 cabang, dan jumlah cabang berkurang satu setiap kali mercusuar dibangun. Permainan selalu berakhir tepat setelah 14 langkah, sehingga game tree memiliki kedalaman paling besar 14.

Kompleksitas Waktu dan Skalabilitas

Waktu berpikir agen ditentukan oleh dua besaran. Branching factor b adalah jumlah langkah legal pada satu state, yaitu jumlah sel kosong. Pada permainan ini b bernilai 49 pada langkah pertama dan 36 pada langkah terakhir. Kedalaman pencarian d adalah jumlah langkah ke depan

yang diperiksa.

Minimax biasa memeriksa setiap kemungkinan langkah hingga kedalaman d [1]. Jumlah leaf node N , kompleksitas waktu T , dan kebutuhan memori M ditunjukkan pada Persamaan (1).

$$N_{\text{minimax}} = b^d, \quad T = O(b^d), \quad M = O(b \cdot d) \quad (1)$$

Penilaian setiap leaf node membutuhkan waktu konstan untuk ukuran papan yang tetap, karena fungsi evaluasi memeriksa paling banyak empat arah cahaya dari setiap sel. Pencarian dilakukan secara depth-first, sehingga memori hanya bertambah sebanding dengan $b \cdot d$.

Alpha-beta pruning menghasilkan keputusan yang sama dengan minimax biasa, tetapi jumlah leaf node yang diperiksa bergantung pada urutan langkah. Jika langkah terbaik selalu diperiksa lebih dahulu, jumlahnya mengikuti Persamaan (2) [2].

$$N_{\text{ideal}} = b^{\lceil d/2 \rceil} + b^{\lfloor d/2 \rfloor} - 1, \quad T = O(b^{d/2}) \quad (2)$$

Jika langkah terbaik selalu diperiksa paling akhir, tidak ada cabang yang di-prune dan jumlahnya sama dengan Persamaan (1). Jumlah leaf node sebenarnya berada di antara Persamaan (2) dan Persamaan (1), bergantung pada urutan langkah.

Agen akan menggunakan iterative deepening. Pencarian dijalankan berturut-turut pada kedalaman 1, 2, 3, dan seterusnya hingga kedalaman maksimum tingkat tersebut atau hingga akhir permainan. Langkah terbaik dari setiap kedalaman diperiksa lebih dahulu pada kedalaman berikutnya agar lebih banyak cabang dapat di-prune. Pengulangan pada kedalaman yang lebih dangkal menambah waktu. Pada minimax biasa, setiap kedalaman memiliki sekitar b kali lebih banyak node daripada kedalaman sebelumnya, sehingga tambahan waktunya mengikuti Persamaan (3) [1].

$$\frac{b^1 + b^2 + \dots + b^{d-1}}{b^d} \approx \frac{1}{b-1} = \frac{1}{48} \approx 2\% \quad (3)$$

Dengan alpha-beta, pertumbuhan jumlah node per kedalaman lebih kecil sehingga tambahannya dapat lebih besar.

Kedalaman terbesar, yaitu 4 pada tingkat Expert, dipilih karena jumlah leaf node pada Persamaan

(1) bertambah cepat seiring kedalaman. Menjelang akhir permainan, sisa langkah dapat lebih sedikit daripada kedalaman tingkat, sehingga pencarian berhenti pada terminal state.

Fungsi Evaluasi

Jika pencarian berhenti sebelum permainan selesai, state s dinilai dengan fungsi evaluasi $E(s)$ pada Persamaan (4), dengan $S_{\text{agen}}(s)$ dan $S_{\text{pemain}}(s)$ adalah jumlah sel milik agen dan milik pemain manusia. Batas nilainya berasal dari jumlah sel papan, yaitu 49.

$$E(s) = S_{\text{agen}}(s) - S_{\text{pemain}}(s), \quad -49 \leq E(s) \leq 49 \quad (4)$$

Terminal state dinilai dari hasil akhirnya seperti pada Tabel 1. Kemenangan agen bernilai 1000 ditambah $E(s)$, kekalahan bernilai -1000 ditambah $E(s)$, dan seri bernilai 0. Karena nilai mutlak $E(s)$ paling besar 49, state menang atau kalah selalu bernilai lebih ekstrem daripada state yang belum selesai. Dengan aturan ini agen memilih kemenangan dengan selisih terbesar dan kekalahan dengan selisih terkecil.

State	Nilai
Permainan belum selesai	$E(s)$
Agen menang	$1000 + E(s)$
Agen kalah	$-1000 + E(s)$
Seri	0

Tabel 1. Nilai state

Tingkat Kesulitan

Kekuatan agen diatur dengan dua parameter. Keduanya berupa aturan tetap yang ditentukan sebelum permainan.

Kedalaman pencarian menentukan berapa langkah ke depan yang diperiksa agen. Kedalaman paling besar sama dengan sisa langkah dalam permainan.

Softmax temperature T menentukan seberapa sering agen memilih langkah selain langkah terbaik. Setiap langkah m di root search tree diberi nilai minimax $v(m)$, lalu dipilih secara acak dengan peluang $P(m)$ pada Persamaan (5), dengan v^* adalah nilai langkah terbaik.

$$P(m) = \frac{\exp((v(m) - v^*)/T)}{\sum_{m'} \exp((v(m') - v^*)/T)} \quad (5)$$

Langkah yang nilainya mendekati langkah terbaik sering dipilih, sedangkan langkah yang jauh lebih buruk jarang dipilih. Pada $T = 0$ agen selalu memilih langkah terbaik. Softmax membutuhkan nilai yang tepat untuk setiap langkah, sedangkan alpha-beta hanya menghasilkan batas nilai untuk langkah selain langkah terbaik. Karena itu, pada tingkat dengan $T > 0$ setiap langkah di root dicari dengan window alpha-beta penuh. Nilai pada Tabel 2 merupakan nilai awal yang dapat disesuaikan setelah program diuji.

Tingkat	Kedalaman	Softmax temperature (T)
Beginner	1	20
Easy	1	2
Medium	2	1
Hard	2	0
Expert	4	0

Tabel 2. Parameter tingkat kesulitan

PEAS

Komponen	Isi
Performance	Menang, seri, atau kalah, selisih jumlah sel, dan waktu berpikir agen
Environment	Papan 7×7 , mercusuar kedua pemain, pemain manusia, dan timer pemain
Actuators	Membangun mercusuar di sel kosong yang dipilih
Sensors	Posisi mercusuar di papan dan giliran pemain

Tabel 3. Deskripsi PEAS agen

Sifat Lingkungan

Sifat	Nilai	Alasan
Observability	Fully observable	Kedua pihak melihat seluruh papan
Agen	Multi-agent, kompetitif	Manusia melawan agen
Determinisme	Deterministic	Hasil setiap langkah ditentukan oleh aturan. Langkah acak pada tingkat rendah berasal dari pilihan agen
Episode	Sequential	Setiap langkah memengaruhi langkah berikutnya
Dinamika	Semi-dynamic	Papan tetap selama pemain berpikir, tetapi timer pemain berjalan dan pemain kalah jika waktunya habis, seperti catur dengan jam [1]
State space	Discrete	49 sel dan paling banyak 49 pilihan langkah
Pengetahuan aturan	Known	Aturan permainan tetap dan diketahui agen

Tabel 4. Sifat lingkungan

Input dan Output

- Input: tingkat kesulitan yang dipilih, sel yang diklik pemain, dan sisa waktu pemain.
- Output: langkah agen, papan yang diperbarui beserta pemilik setiap sel, jumlah sel kedua pemain, dan hasil akhir permainan (menang, seri, atau kalah).

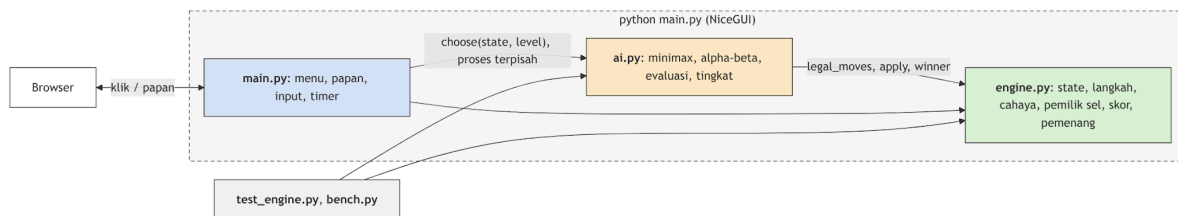
Program akan dibangun menggunakan Python dan dijalankan sebagai aplikasi web sehingga dapat dimainkan dari browser laptop atau smartphone.

Program akan terdiri atas dua halaman, yaitu halaman pemilihan tingkat kesulitan dan halaman permainan. Halaman permainan menampilkan papan, jumlah sel dan sisa mercusuar kedua pemain, dan timer pemain.

Arsitektur Software

Seluruh program akan ditulis dalam Python. UI akan dibangun dengan NiceGUI [4], library Python untuk membuat UI web. Program akan berjalan sebagai server Python dengan perintah `python main.py`, dan pemain membuka permainan melalui browser di laptop atau smartphone. Papan, tombol, dan timer akan ditulis dalam Python. Di pos rally, program akan dijalankan dan dibuka di laptop yang sama. Setiap tab browser mendapat permainan sendiri, sehingga beberapa pemain dapat bermain bersamaan. Library yang dibutuhkan hanya NiceGUI.

Program akan dibagi menjadi tiga lapisan, seperti ditunjukkan pada Gambar 2 dan Tabel 5. Lapisan aturan (`engine.py`) dan lapisan agen (`ai.py`) hanya memakai Python standar, sehingga keduanya dapat diuji dan diukur kinerjanya secara terpisah dari UI.



Gambar 2. Arsitektur software

Modul	Tanggung jawab
engine.py	State permainan, langkah legal, pembangunan mercusuar, cahaya dan pemilik sel, jumlah sel, dan pengecekan pemenang
ai.py	Minimax dengan alpha-beta pruning, fungsi evaluasi, urutan langkah, iterative deepening, dan parameter tingkat kesulitan
main.py	Menu, tampilan papan, input pemain, timer, dan layar hasil
test_engine.py	Pengujian aturan permainan dan kesamaan nilai alpha-beta dengan minimax biasa
bench.py	Pengukuran jumlah node dan waktu berpikir agen

Tabel 5. Rencana pembagian modul program

State permainan akan disimpan sebagai tuple yang tidak dapat diubah. Tuple ini berisi 49 sel papan dan giliran pemain. Pemilik sel dihitung dari posisi mercusuar setiap kali dibutuhkan, menggunakan daftar sel di keempat arah dari setiap sel yang dihitung sekali di awal. Langkah akan direpresentasikan sebagai bilangan 0 sampai 48, yaitu indeks sel.

UI akan memanggil agen melalui satu fungsi, `choose(state, level)`, yang mengembalikan langkah agen. Fungsi ini akan dijalankan di proses terpisah dengan `run.cpu_bound` dari `NiceGUI` agar tampilan tetap dapat diperbarui selama agen berpikir. Proses terpisah dipakai karena pada Python, thread yang menjalankan perhitungan berat menghambat thread lain. State berupa tuple, sehingga dapat langsung dikirim ke proses tersebut. Timer pemain dihitung dengan `time.monotonic()` dan hanya berjalan saat permainan menunggu langkah pemain manusia.

Permainan dimulai dengan langkah agen, yaitu langkah 3 dan 4 di bawah. Alur satu giliran adalah sebagai berikut:

1. Pemain menekan sel kosong. UI memeriksa bahwa langkah tersebut ada di `legal_moves(state)`, menghentikan timer pemain, lalu menghitung state baru dengan `apply(state, move)`.
2. UI menampilkan papan baru, lalu memeriksa `winner(state)`. Jika permainan selesai, layar hasil ditampilkan.
3. UI menampilkan status menunggu dan memanggil `choose(state, level)` di proses terpisah.

4. Langkah agen diterapkan dengan `apply(state, move)`, papan baru ditampilkan, lalu `winner(state)` diperiksa kembali.
5. Timer pemain berjalan lagi dan UI menunggu langkah berikutnya.

2.2 Metode Evaluasi

Label setiap permainan adalah hasil akhirnya, yaitu menang, seri, atau kalah. Pengujian yang direncanakan:

1. Kebenaran aturan. Pengujian otomatis memeriksa cahaya pada baris dan kolom, cahaya yang terhalang mercusuar, sel yang diterangi kedua pemain, perhitungan jumlah sel, akhir permainan setelah 14 mercusuar, menang dan seri, serta urutan giliran.
2. Kebenaran pruning, dengan minimax biasa sebagai baseline. Pada 200 posisi yang belum selesai, yang dibentuk dari 2 sampai 10 langkah acak dari papan kosong, alpha-beta dan minimax biasa dijalankan dengan kedalaman 3 dan urutan langkah yang sama. Keduanya harus menghasilkan nilai minimax yang sama untuk setiap posisi. Jumlah node dan waktu eksekusi keduanya dicatat dan dibandingkan.
3. Kecepatan. Waktu berpikir agen setiap tingkat dicatat pada 200 posisi uji yang sama.
4. Durasi permainan. Paling sedikit lima penguji memainkan satu permainan masing-masing melalui smartphone, lalu durasi setiap permainan dicatat.

2.3 Jadwal Pelaksanaan

Pengerjaan dibagi ke dalam tahapan berikut. Urutan dan durasi setiap tahap dapat disesuaikan dengan kemajuan pengerjaan, dan beberapa tahap dapat berjalan bersamaan.

Tahap	Perkiraan waktu	Kegiatan
Perancangan	Awal pengerjaan	Menetapkan aturan permainan, struktur data papan, dan rancangan UI
Implementasi aturan	Minggu pertama	Logika papan, cahaya mercusuar, pemilik sel, pengecekan pemenang, dan pengujian otomatisnya
Implementasi agen	Minggu pertama hingga kedua	Minimax, alpha-beta pruning, dan fungsi evaluasi
UI dan fitur permainan	Minggu kedua	Tampilan web, tingkat kesulitan, dan timer pemain
Pengujian dan penyesuaian	Menjelang akhir pengerjaan	Menjalankan pengujian evaluasi dan menyesuaikan parameter
Finalisasi	Sebelum batas pengumpulan	Perbaikan akhir, laporan, dan persiapan presentasi

Tabel 6. Jadwal pelaksanaan

3. Glosarium

Riwayat Revisi

Versi	Tanggal	Keterangan
1.0	Oktober 2026	Penyusunan awal proposal

Tabel 7. Riwayat revisi dokumen

Istilah

Istilah	Definisi
Adversarial search	Pencarian dalam lingkungan di mana dua agen memiliki tujuan yang saling berlawanan
Alpha-beta pruning	Teknik untuk melewati cabang game tree yang tidak dapat mengubah keputusan minimax
Branching factor	Jumlah langkah yang tersedia pada satu state permainan
Fungsi evaluasi	Fungsi yang memberi nilai perkiraan pada state permainan yang belum selesai
Game tree	Tree yang node-nya berupa state permainan dan cabangnya berupa langkah
Iterative deepening	Teknik pencarian yang mengulang pencarian dengan kedalaman yang terus bertambah hingga batas kedalaman tercapai
Kedalaman (depth)	Jumlah langkah ke depan yang diperiksa oleh algoritma pencarian
Kompleksitas waktu	Ukuran pertumbuhan waktu eksekusi algoritma terhadap ukuran masalah, dinyatakan dengan notasi O
Mercusuar	Bangunan di satu sel yang menyinari baris dan kolomnya ke empat arah sampai terhalang mercusuar lain
Minimax	Algoritma yang memilih langkah terbaik dengan asumsi lawan selalu memilih langkah terburuk bagi agen
PEAS	Performance, Environment, Actuators, Sensors: framework untuk mendeskripsikan agen
NiceGUI	Library Python untuk membuat UI web yang dibuka melalui browser
Pemilik sel	Pembangun mercusuar di sel tersebut, atau satu-satunya pemain yang cahayanya mencapai sel kosong tersebut
Softmax	Pemilihan acak yang lebih sering memilih langkah bernilai tinggi
Terminal state	State permainan yang sudah berakhir dengan menang, kalah, atau seri

Tabel 8. Daftar istilah

4. Daftar Pustaka

- [1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Hoboken, NJ, USA: Pearson, 2020.
- [2] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artificial Intelligence*, vol. 6, no. 4, pp. 293–326, 1975.
- [3] C. E. Shannon, "Programming a computer for playing chess," *Philosophical Magazine*, vol. 41, no. 314, pp. 256–275, 1950.
- [4] Zauberzeug GmbH, "NiceGUI." [Online]. Available: <https://nicegui.io/> (accessed Oct. 15, 2026).