Implementasi *Propositional Logic* dalam Membantu Penyelesaian *Game Minesweeper*

Muhammad Ferdinanta
Computer Science
IPB University
Bogor, Indonesia
md.ferdinanta@apps.ipb.ac.id

Tioninta Mandaika Maha
Computer Science
IPB University
Bogor, Indonesia
mahamaha@apps.ipb.ac.id

Khalid Zia Rabbani
Computer Science
IPB University
Bogor, Indonesia
khalidziarabbani@apps.ipb.ac.id

Muhamad Surya Fauzan
Computer Science
IPB University
Bogor, Indonesia
officiallysurya@apps.ipb.ac.id

Abstrak - Perkembangan teknologi informasi dan kecerdasan buatan telah memungkinkan implementasi entitas cerdas dalam menyelesaikan permainan Minesweeper. Minesweeper, sebagai permainan logika yang kompleks, dapat dipecahkan dengan efisien menggunakan propositional logic. Penelitian ini mengusulkan pendekatan berbasis logika proposisi untuk meningkatkan kinerja AI dalam menyelesaikan Minesweeper. Logika proposisi digunakan untuk menyatakan hubungan antar sel pada papan permainan, memungkinkan AI membuat keputusan cerdas. Metode ini diimplementasikan melalui library pygame untuk antarmuka grafis. Fitur "Show Inference" memvisualisasikan proses berpikir AI, menandai sel yang aman dengan warna merah muda, meningkatkan pemahaman pengguna. Pengembangan ini menciptakan pengalaman Minesweeper yang inovatif, menggabungkan elemen tradisional dengan fitur-fitur interaktif, mendukung pembelajaran AI, dan meningkatkan keterlibatan pengguna.

Keywords—Artificial Intelligence, Minesweeper, Propositional Logic, Puzzle, insert

I. Pendahuluan

Perkembangan teknologi informasi yang sangat pesat telah membawa perubahan dalam kehidupan manusia. Kini manusia dapat memecahkan berbagai masalah yang dihadapi dengan bantuan komputer. Hal ini dimungkinkan karena berkembangnya salah satu bagian dari ilmu komputer yaitu *Artificial Intelligence* atau kecerdasan buatan yang dapat membuat komputer berpikir dan bertindak seperti yang dilakukan oleh manusia. AI diciptakan tidak hanya untuk memahami namun serta menciptakan entitas cerdas [5]

Salah satu implementasi entitas cerdas adalah yang dapat menemukan solusi terhadap penyelesaian permainan Minesweeper. Minesweeper adalah permainan logika populer yang dimainkan oleh satu orang. Permainan ini dimulai dengan papan berukuran p x q yang terdiri dari pq kotak kosong. Di antara kotak-kotak tersebut terdapat M ranjau yang tersebar secara acak dan merata. Tugas pemain adalah membuka semua kotak yang tidak berisi ranjau. Pemain dapat membuka kotak dengan mengkliknya. Jika kotak tersebut berisi ranjau, maka pemain kalah. Sebaliknya, jika kotak tersebut tidak berisi ranjau, maka akan ditampilkan jumlah ranjau yang berada di sekitar kotak tersebut[6].

Penyelesaian permainan Minesweeper dapat dilakukan secara manual oleh manusia dengan menggunakan logika dan strategi. Namun, penyelesaian secara manual dapat menjadi sulit dan memakan waktu, terutama untuk papan permainan berukuran besar. Oleh karena itu, diperlukan metode yang lebih efisien untuk menyelesaikan permainan Minesweeper.

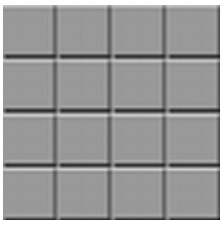
Propositional logic adalah salah satu cabang logika yang digunakan untuk mengekspresikan hubungan antara proposisi. Proposisi adalah pernyataan yang dapat dibuktikan benar atau salah[3]. Dalam konteks permainan Minesweeper, proposisi dapat digunakan untuk menyatakan hubungan antara nilai dua sel di papan permainan, seperti "sel A kosong" atau "sel B berisi bom".

Implementasi propositional logic dalam permainan Minesweeper dapat membantu pemain untuk menyelesaikan permainan ini dengan lebih efisien. Dengan menggunakan propositional logic, pemain dapat menganalisis hubungan antara proposisi-proposisi tersebut dan menarik kesimpulan tentang nilai sel-sel lain di papan. Misalnya, jika pemain mengetahui bahwa dua sel yang berdekatan memiliki jumlah bom yang sama, maka pemain dapat menyimpulkan bahwa sel-sel yang tersisa di sekitarnya pasti kosong atau berisi bom. Pengerjaan ini dilaksanakan dengan harapan mendapatkan wawasan lebih lanjut tentang penerapan kecerdasan buatan, sekaligus memberikan bantuan yang lebih efektif kepada pemain untuk menyelesaikan permainan dengan lebih mudah dan cepat.

II. Tinjauan Pustaka

A. Minesweeper

Minesweeper adalah permainan klasik *single-agent* dengan informasi tidak lengkap (*uninformed*), yang mendapatkan popularitasnya sebagai fitur Microsoft Windows pada tahun 1989 [4]. Tujuan dari permainan ini adalah untuk menentukan lokasi beberapa ranjau yang tersembunyi di dalam kotak persegi panjang. Pemain diminta untuk mengklik sel. Jika sel yang berisi ranjau diklik, permainan berakhir. Jika tidak, jumlah total ranjau di sel tetangga ditampilkan, dan informasi ini dapat digunakan untuk memandu keputusan selanjutnya tentang sel mana yang akan diklik. Permainan dimenangkan ketika semua ranjau telah ditemukan [1].



Gambar 1. Posisi awal

(a) kondisi kalah (b) kondisi menang

Gambar 2. Posisi akhir

B. *Propositional Logic*

Propositional Logic adalah cabang logika yang mempelajari cara menggabungkan pernyataan sederhana untuk membentuk pernyataan yang lebih kompleks [3]. Salah satu cara untuk menggabungkan pernyataan sederhana adalah dengan menggunakan konjungsi seperti kata “dan”. Ketika dua pernyataan digabungkan dengan "dan", pernyataan kompleks yang terbentuk hanya benar jika kedua pernyataan komponennya benar. Selain dengan menggunakan konjungsi, operator logika lainnya juga dapat digunakan untuk menggabungkan pernyataan sederhana lainnya. Seperti, operator logika negasi, disjungsi, implikasi, dan biimplikasi. Keterkaitan antara Minesweeper dan logika proposisi terletak pada penggunaan operator logika untuk menyimpulkan apakah sel tertentu aman atau tidak. Misalnya, jika pemain mengetahui bahwa ada dua bom di sel-sel yang berdekatan dengan sel tertentu, maka pemain dapat menyimpulkan bahwa sel tersebut aman.

C. *Artificial Intelligence*

Artificial Intelligence adalah sebuah cabang ilmu yang mempelajari, bagaimana cara membuat sebuah mesin yang mampu berpikir dan bertindak layaknya seorang manusia. *Artificial Intelligence* juga dapat didefinisikan sebagai kemampuan sistem untuk memahami dan menanggapi dunia di sekitarnya. Kemampuan ini meliputi kemampuan untuk belajar dari pengalaman, membuat keputusan yang tepat, dan memecahkan masalah. AI dapat diterapkan dalam berbagai bidang, seperti pendidikan, kesehatan, dan industri [4].

III. METODE

Metode yang diterapkan dalam pengembangan program ini menggunakan pendekatan berbasis *Propositional Logic*. Pendekatan ini memungkinkan *AI* membuat keputusan dengan mempertimbangkan *knowledge base* yang dimilikinya, serta menghasilkan *inference* berdasarkan pengetahuan tersebut. Dalam proses pengerjaan kode program, kami memanfaatkan library pygame untuk menampilkan graphical user interface (GUI) pada permainan Minesweeper. Adapun representasi pengetahuan *AI* terhadap permainan *Minesweeper* dilakukan dengan menjadikan setiap sel atau kotak pada papan permainan sebagai *propositional variable*. Variabel ini dianggap benar *(true)* ketika sel tersebut berisi ranjau dan dianggap salah *(false)* jika sebaliknya. Penggunaan library pygame membantu memperkaya pengalaman bermain dengan menyediakan antarmuka visual yang menarik.

IV. PEMBAHASAN

Program ini bermulai dengan menampilkan GUI menggunakan *library pygame*. Selanjutnya, sebagai user dapat memilih sel secara acak untuk memulai permainan. Ranjau pada program ini diacak setiap iterasi program ini dijalankan. Terdapat fitur utama pada program ini, yaitu *Show Inference.*

Fitur *Show Inference* menjadi inti dari pengalaman bermain *Minesweeper* pada program ini. Dengan memvisualisasikan *logical inferences* yang dimiliki *AI* selama bermain game. Fitur ini meningkatkan pemahaman pengguna tentang bagaimana *AI* mengidentifikasi sel yang aman dengan memberikan warna merah muda sebagai penanda sehingga membuat proses pengambilan keputusan menjadi transparan dan dapat dilihat pengguna.

```
def add_knowledge(self, cell, count):

    self.mark_safe(cell)
    self.moves_made.add(cell)
    ...
```

Gambar 3. Metode *add_knowledge*

```
def add_knowledge(self, cell, count):
    ...
    neighbors, count=self.get_cell_neighbors(cell,
count)
        sentence = Sentence(neighbors, count)
        self.knowledge.append(sentence)
    ...
```

Gambar 4. Metode *add_knowledge*

Metode *add_knowledge* pada Gambar 3 memainkan peran penting dalam proses ini. *Knowledge base* disimpan dengan menggunakan fungsi *add_knowledge.* Fungsi ini akan menyimpan nilai sel yang aman dan langkah yang telah dilakukan. Selanjutnya, mengambil sel tetangga untuk membuat *sentence* seperti pada gambar 4.

Setelah menandai sel yang aman dan memperbarui *knowledge base*, *AI* menggunakan pengetahuan yang baru untuk menarik kesimpulan tambahan. Kemudian memeriksa *knowledge base* dan mencari peluang untuk mengidentifikasi sel yang lebih aman atau yang berisi ranjau berdasarkan pola dan hubungan yang sudah ada.

```python
def get_cell_neighbors(self, cell, count):
    i, j = cell
    neighbors = []

    for row in range(i-1, i+2):
        for col in range(j-1, j+2):
            if (row >= 0 and row < self.height)\
            and (col >= 0 and col < self.width)\
            and (row, col) != cell \
            and (row, col) not in self.safes \
            and (row, col) not in self.mines:
                neighbors.append((row, col))
            if (row, col) in self.mines:
                count -= 1

    return neighbors, count
```

Gambar 5. Metode *get_cell_neighbors*

Fitur *show inference* melakukan pengecekan pada sel tetangga yang telah dipilih menggunakan metode Metode *get_cell_neighbors*. Metode ini memerlukan input sel yang berisikan sel saat ini dan *count* yang menandakan jumlah mines yang ada saat program dimulai. Metode ini akan mengembalikan koordinat sel tetangga yang berdasarkan semua sel yang telah kita pilih semuanya.

```python
# Conclusion
new_inferences = []
for s in self.knowledge:
    if s == sentence:
        continue
    elif s.cells.issuperset(sentence.cells):
        setDiff = s.cells-sentence.cells
        # Known safes
        if s.count == sentence.count:
            for safeFound in setDiff:
                self.mark_safe(safeFound)
        # Known mines
        elif len(setDiff) == s.count - sentence.count:
            for mineFound in setDiff:
                self.mark_mine(mineFound)
        # Known inference
        else:
            new_inferences.append(
                Sentence(setDiff, s.count - sentence.count)
            )
    elif sentence.cells.issuperset(s.cells):
        setDiff = sentence.cells-s.cells
        # Known safes
        if s.count == sentence.count:
            for safeFound in setDiff:
                self.mark_safe(safeFound)
        # Known mines
        elif len(setDiff) == sentence.count - s.count:
            for mineFound in setDiff:
                self.mark_mine(mineFound)
        # Known inference
        else:
            new_inferences.append(
                Sentence(setDiff, sentence.count - s.count)
            )
```

Gambar 6. Metode *add_knowledge*

Selanjutnya pada gambar 6 membuat inference baru dengan *knowledge* lama dan *neighbors*. Melakukan perbandingan antara knowledge lama dan baru, apabila nilai *count* sama antara knowledge lama dan knowledge baru akan dipastikan knowledge baru ini merupakan *safe moves* namun apabila terdapat perbedaan maka dipastikan terdapat *mines* diantara nilai neighbor ini

Fitur *Show inference* direpresentasikan secara visual dengan menandakan sel tetangga yang aman dengan warna merah muda. Sedangkan sel tetangga yang berisi ranjau ditandakan dengan warna merah. Isyarat visual ini berfungsi sebagai indikator yang jelas tentang proses pengambilan keputusan *AI*, sehingga pengguna dapat melihat bagaimana *AI* mengidentifikasi sel yang aman. Fitur *Show Inference* juga dibuat untuk menggantikan fitur klasik dalam game *minesweeper* dimana jika tiap angka dalam sel yang sudah terbuka ditekan kembali, maka setiap sel yang berada di sekelilingnya dan dianggap aman akan terbuka.

Fitur *Show Inference* meningkatkan keterlibatan pengguna dan pemahaman mendalam terhadap strategi yang diterapkan oleh *AI*. Dengan menampilkan proses berpikir *AI*, pengguna dapat belajar dari keputusan yang diambil *AI* dan mengasah keterampilan mereka sendiri. Ini menciptakan pengalaman bermain yang lebih interaktif dan mendidik, tidak hanya sekedar permainan keberuntungan semata.

Secara keseluruhan, pengembangan *Minesweeper* ini menggabungkan elemen-elemen tradisional dengan fitur-fitur inovatif, memberikan pengalaman bermain yang menarik dan mendalam.

V. KESIMPULAN

Permainan *Minesweeper* adalah permainan logika yang mengandalkan kondisi sel-sel pada papan. Kondisi sel-sel ini ditentukan oleh jumlah ranjau di sel-sel tetangga. Propositional Logic digunakan untuk menghitung jumlah ranjau di sel-sel tetangga, sehingga pemain dapat menentukan langkah selanjutnya dengan lebih aman.

AI dapat membantu pemain *Minesweeper* untuk menyelesaikan permainan dengan lebih mudah. Hal ini dikarenakan AI dapat membuat keputusan yang lebih tepat tentang sel mana yang harus dibuka selanjutnya. Pengetahuan yang dibangun oleh AI menggunakan logika proposisional, yang dapat digunakan untuk menghitung jumlah ranjau di sel-sel tetangga.

VI. SARAN

Untuk meningkatkan akurasi prediksi AI, diperlukan metode lain selain logika proposisional. Hal ini dikarenakan pengetahuan yang dibangun hanya berdasarkan pergerakan sebelumnya, yang dapat membatasi kapabilitas AI dalam membuat prediksi.

REFERENCES

[1] Chlond, M. J. (2011). Puzzle—Minesweeper Puzzles. INFORMS Transactions on Education, 11(2), 90-91.

[2] Dwiyanto, A. (2022). Potensi dan Kekurangan dari Adanya Robot Dengan AI (*Artificial Intelligence*).

[3] Klement, K. C. (2004). Propositional logic. Internet Encyclopedia of Philosophy.

[4] Liu, C., Huang, S., Naying, G., Khalid, M. N. A., & Iida, H. (2022). A solver of single-agent stochastic puzzle: A case study with Minesweeper. Knowledge-Based Systems, 246, 108630.

[5] Russell, S. J., & Norvig, P. (2010). Artificial intelligence a modern approach. London.

[6] Castillo, L. P., & Wrobel, S. (2003, August). Learning minesweeper with multirelational learning. In International Joint Conference On Artificial Intelligence (Vol. 18, Pp. 533-540). Lawrence Erlbaum Associates Ltd.