Alex Lu 03/31/2025

ESE 561 Artificial Intelligence
Assignment 2

Part 1: Algorithms

1. **A* Graph Search**

   (a) In what order are states expanded by A* graph search?
   $S \to A \to B \to D \to G \to C$

   (b) What path does A* graph search return?
   $S - A - D - G$

2. **Zero-Sum Game Tree**
   $A = 3$
   $B = 2$
   $C = 1$
   $D = 3$

3. **Expectiminimax Search**
   $A = 15$
   $B = 19$
   $C = 13$
   $D = 19$
   $E = 15$
   $F = 13$
   $G = 15$

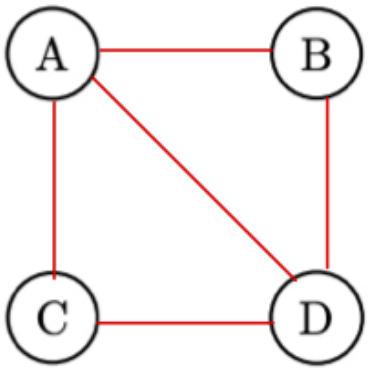
4. **Mini-max Tree Pruning**
   A, B, C, E are guaranteed to never be pruned.

5. **CSP: Apartment Assignment**

   Variables: A, B, C, D (people)
   Domains: $\{1, 2, 3\}$ (floors)
   Constraints:

   - $A \neq B$
   - If $A = C$, then $A = C = 2$
   - If $A \neq C$, then $A = 3$ or $C = 3$
   - $D \neq A, D \neq B, D \neq C$
   - $D > C$

(a)



$A - B : (A \neq B)$
$A - D : (D \neq A)$
$A - C : (if\ A = C : A = C = 2), (if\ A \neq C : A = 3\ or\ C = 3)$
$C - D : (D \neq C), (D > C)$

(b)

A	☒ 1	☐ 2	☐ 3
B	☐ 1	☐ 2	☐ 3
C		☐ 2	
D	☒ 1	☒ 2	☐ 3

When $C = 2$,
Since $D > C$, eliminates $\{1, 2\}$ from D's values.
Since if $A = C$ they must both equal to $\{2\}$, and if $A \neq C$ one of them must equal to 3, eliminates $\{1\}$ from A's values.

(c)

A	☐ 1	☐ 2	☐ 3
B	☐ 1	☐ 2	☐ 3
C	☐ 1	☐ 2	☒ 3
D	☐ 1	☐ 2	☐ 3

If $C = 3$, then D does not have values that satisfy the constraints.

(d)

When $A = 3$, $C = 3$ conflicts with the constraints, and $D = 3$ also conflicts with the constraints.$\rightarrow$ 2 conflicts
When $B = 1 \rightarrow 0$ conflict
When $C = 2$, $A = 3$ conflicts with constraints.$\rightarrow$ 1 conflict

When $D = 3$, $A = 3$ conflicts with constraints.→ 1 conflict

Since A has the most conflicts, reassign A.
When $A = 1$, $B = 1$ conflicts with constraints. When $A = 2$, no conflict occurs. Thus, reassigns A to $A = 2$.

Part 2: Programming

1. **AI for Checkers (Minimax & Alpha-Beta Pruning)**
   - Implement an AI agent that plays Checkers using Minimax with Alpha-Beta Pruning for efficiency.
   - Use an evaluation function to determine board strength.
2. **Scheduling AI (CSP & Local Search)**
   - Solve a university course scheduling problem using CSP.
   - Constraints:
     - No two exams at the same time for the same student
     - Professors have limited availability
   - Use Simulated Annealing or Genetic Algorithms for optimization.

AI for Checkers
Code: *Click here*

Analysis:
This report examines an implementation of a checkers game that employs two adversarial search algorithms—Minimax and Alpha-Beta Pruning—in an AI-driven simulation of optimal gameplay.

Minimax: Minimax exhaustively searches the game tree to a set depth, scoring leaves and returning the optimal move via *Maximize()* and *Minimize()*, tracked by *minimax_moves*. Minimax guarantees optimality and simple to implement, but has high complexity and inefficient for deep searches or large branching factors.

Alpha-Beta Pruning: An optimized Minimax, it prunes irrelevant branches using alpha and beta bounds in *AlphaBeta_Maximize()* and *AlphaBeta_Minimize()*, with *alphabeta_moves* counting the number of attempted moves. Alpha-Beta Pruning is more efficient and it retains optimality. However, the pruning depends on move order.

The Design of Evaluation Function: The evaluate() function scores the board: regular pieces

("b","w") are 1.0, kings ("B","W") are 2.0. AI pieces add the the score, opponent pieces subtract, ignoring empty (".") or unusable ("_") squares. Evaluate function provides a heuristic at depth limits (I set to 6) or game end, guiding the AI to maximize scores. Final scores determine outcomes: positive (AI wins), negative (opponent wins), zero (draw).

Depth: I set depth to 6 in these two implementation, is critical for managing the computational complexity of Minimax and Alpha-Beta Pruning in checkers. With a branching factor of 7 20 moves, exploring the full game tree is infeasible. This limit ensures efficient decision-making, as it provides a heuristic at depth limits via the evaluation function, balancing accuracy and resource use.

Performance comparison:
Both algorithms start the game by setting the board to its initial state, as shown in Figure 1. The performance disparity between Minimax and Alpha-Beta Pruning is pronounced. In *game_loop()*, Minimax's *minimax_moves* reaches 886,073 and takes 28.00 seconds to produce a result, as shown in Figure 2, due to its exhaustive search nature. In contrast, Alpha-Beta in *Prune_game_loop()* leverages pruning, reducing *alphabeta_moves* significantly to 29,144 and taking only 1.42 seconds, as shown in Figure 3. The output data in Figure 4 and Figure 5 further confirms Alpha-Beta's superiority in move count and execution time, highlighting its efficiency—particularly in checkers, given its large branching factor.

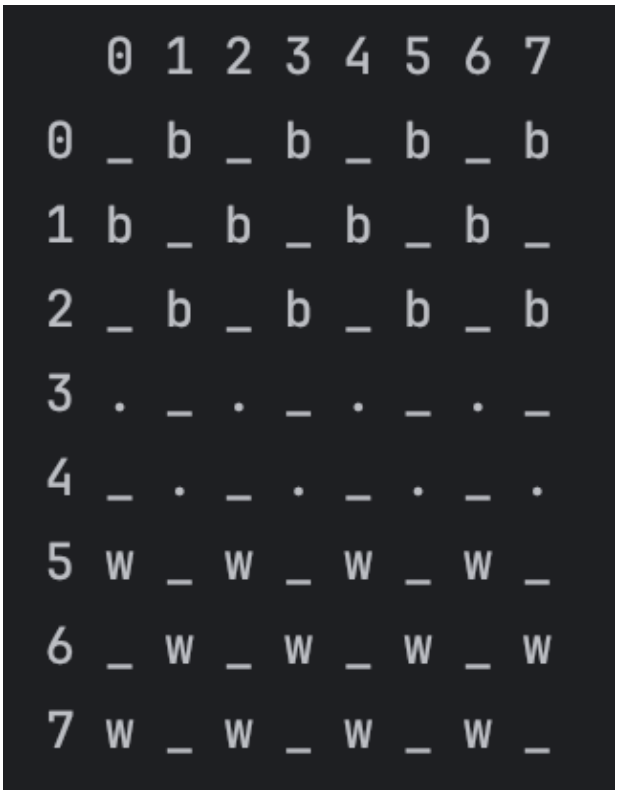


Figure 1: Initial Board State

```
time-consuming : 1.42 sec
Number of moves did <Alpha-Beta Pruning> attempt: 29144 times
```

Figure 3: Alpha-Beta Pruning Algorithm Performance

```
time-consuming : 28.00 sec
Number of moves did <Minimax> attempt: 886073 times
```

Figure 2: Minimax Algorithm Performance

```
It is b's turn.
Selected move: from (5, 6) to (7, 4), full path: [(5, 6), (7, 4)]
Capture positions: [(6, 5)]
  0 1 2 3 4 5 6 7
0 _ . _ b _ b _ b
1 . _ . _ b _ b _
2 _ . _ b _ . _ .
3 . _ . _ . _ . _
4 _ . _ b _ . _ .
5 . _ . _ . _ . _
6 _ . _ . _ . _ b
7 . _ B _ B _ . _

AI WINS!
```

Figure 4: Minimax Result

```
It is b's turn.
Selected move: from (5, 6) to (7, 4), full path: [(5, 6), (7, 4)]
Capture positions: [(6, 5)]
  0 1 2 3 4 5 6 7
0 _ . _ b _ b _ b
1 . _ . _ b _ b _
2 _ . _ b _ . _ .
3 . _ . _ . _ . _
4 _ . _ b _ . _ .
5 . _ . _ . _ . _
6 _ . _ . _ . _ b
7 . _ B _ B _ . _

AI WINS!
```

Figure 5: Alpha-Beta Result

Scheduling AI
Code: *Click here*

Analysis:
This report examines an implementation of a university exam scheduling problem using two distinct approaches—Constraint Satisfaction Problem (CSP) with Backtracking and Simulated Annealing (SA)—each with its own advantages and trade-offs.

CSP Constraints:
(1) Professor Availability Constraint:
Each course must be scheduled in a timeslot during which its assigned professor is available. If a professor is unavailable at a certain timeslot, the corresponding course cannot be scheduled at that time.

(2) Professor Conflict Constraint:
If a professor teaches multiple courses, those courses must be scheduled in different timeslots. This ensures that a professor is not required to supervise more than one exam at the same time.

(3) Student Conflict Constraint:
For any student enrolled in multiple courses, the exams for those courses must be scheduled in different timeslots. This prevents any student from having overlapping exam times.

CSP Scheduling Solution (Backtracking):
The algorithm works by sequentially assigning timeslots to each course and checking the constraints at every step. If an assignment violates any constraint, the algorithm backtracks—removing the previous assignment and trying a different timeslot. This method guarantees that if a solution exists, it will be found, and it will be the unique valid solution for the problem instance. CSP Scheduling Solution have following features:

(1) Deterministic Outcome:
Backtracking explores all combinations until a valid solution is found, ensuring a unique and correct result if one exists.

(2) Systematic Exploration:
The algorithm checks each course and iterates through all timeslots. Upon detecting a conflict, it immediately retracts the last assignment and tests alternatives, covering all possible options.

(3)Computational Complexity:
While guaranteeing a solution, the exhaustive search can become computationally expensive as the number of courses and constraints increases, potentially leading to a dramatic rise in backtracking

attempts in larger scenarios.

Simulated Annealing Scheduling Solution:
The simulated annealing (SA) approach, on the other hand, is a probabilistic local search algorithm that aims to find a near-optimal solution by minimizing a cost function that represents constraint violations:

Cost Function Design:

- Professor Availability: The cost function adds 5 points penalty.
- Professor Conflict: The cost function adds 5 points penalty.
- Student Conflict: A lower penalty (1 point) is added.

Algorithm Behavior:

- The algorithm starts with an initial random assignment of timeslots to courses.
- At each iteration, a neighboring solution is generated by randomly changing the timeslot for one course.
- The new solution is accepted if it improves the cost, or with a certain probability based on the current temperature (controlled by the exponential acceptance probability) even if the solution is worse.
- The temperature gradually decreases (cooling rate), reducing the probability of accepting worse solutions and allowing the algorithm to converge.

Performance Comparison:
CSP (Backtracking):

- Pros: Guarantees an optimal and valid solution when one exists; conceptually simple and straightforward to implement.
- Cons: Can be computationally expensive due to its exhaustive search, as seen by the increasing number of backtracking attempts with larger or more complex problem instances.

Simulated Annealing:

- Pros: More efficient in practice for large search spaces; capable of escaping local minima and quickly converging to a near-optimal solution.
- Cons: The quality of the solution depends on the parameter settings and the randomness inherent in the process; it does not ensure a completely constraint-satisfying solution if the problem is particularly challenging.

The final comparison result shows in Figure 6.

```
【CSP Scheduling Solution】
Final Assignment: {'Class1': 0, 'Class2': 1, 'Class3': 2, 'Class4': 3, 'Class5': 2, 'Class6': 3, 'Class7': 4, 'Class8': 5, 'Class9': 5, 'Class10': 6, 'Class11': 7}
assignment attempted:49 times

【Simulated Annealing Scheduling Solution】
Final Assignment: {'Class1': 3, 'Class2': 2, 'Class3': 1, 'Class4': 4, 'Class5': 4, 'Class6': 3, 'Class7': 2, 'Class8': 6, 'Class9': 5, 'Class10': 9, 'Class11': 6}
cost: 0
assignment attempted:22 times
```

Figure 6: Final Result