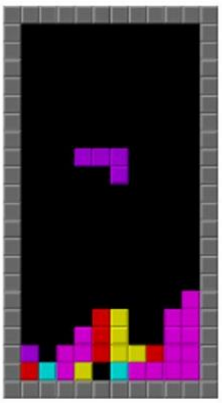
National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 3: Heuristics

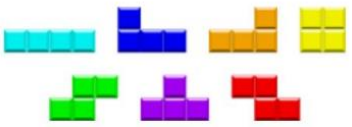
SOLUTIONS

---

1. Tetris (fill-the-board variant) is a tile-matching game in which pieces of different geometric forms, called Tetriminos, descend from the top of the field. During this descent, the player can move the pieces laterally (move left, move right) and rotate (rotate right, rotate left) them until they touch the bottom of the field or land on a piece that had been placed before it. The player can neither slow down the falling pieces nor stop them.



Assume, for our purposes that we are playing a version of Tetris that instead works as follows. Each turn, when a new piece appears to be placed, the player must select the location and orientation before it falls. Once the piece begins to fall, no other adjustments are possible.



The objective of the game is to configure the pieces to fill a board completely without any surrounded gaps. A gap is defined as an empty cell on the board. A row (or column) is complete if there are no gaps in that row (or column respectively). A gap is called a blocked gap if for the corresponding column where the gap belongs to, there exists an occupied cell somewhere above that gap.

There are 7 kinds of Tetriminos (refer to the image above). Assume that we start with a fixed number of Tetriminos, ***N*** (comprising some of each kind), and all are required to be used to fill the board (i.e. there exists a way to place all these tetriminos such that the board is filled).

Thus, in this problem, the **states** are different partially filled Tetris fields with a Tetrimino that is about to be placed in the field next (but not placed yet); the **initial state** is an empty field with a starting Tetrimino; an **action** is the choice of orientation and column for the give piece (assume the player selects an intended configuration before descent); the **transition model** takes in a state, applies the given action on the Tetrimino that enters the field, and outputs a state where the Tetrimino of that specified configuration descended onto the field; the **goal state** is a completely filled board where there are no gaps (and every Tetrimino fits perfectly); the **transition cost** is 1. You may assume there exists such a goal state.

(a) Select all the heuristics that are admissible from. If you feel that none **are admissible**, select only the option "*None of these options are admissible*". For each option, briefly, but clearly, explain why it is admissible/inadmissible.

- $h_1(n)$ = number of unfielded Tetriminos
- $h_2(n)$ = number of gaps
- $h_3(n)$ = number of incomplete rows
- $h_4(n)$ = number of blocked gaps
- None of these options are admissible.

Solution: Only $h_1(n)$ and $h_4(n)$ are admissible.

- h_1: **Admissible**. The number of unfielded Tetriminos is the minimum number of steps required to get to the goal, so it is admissible.
- h_2: **Inadmissible**. Consider a field where a there is a horizontal gap line of size 4. Then, the cyan Tetrimino rotated can fill it in 1 move. The cost to the goal is 1, but h_2 returns 4 – i.e., it overestimates.
- h_3: **Inadmissible**. Consider a field where there is a vertical gap line of size 4. Then, the cyan Tetrimino can fill it in 1 move. The cost to the goal is 1, but h_3 returns 4 – i.e., it overestimates.
- h_4: **Admissible** (under assumption that blocked gaps may not be filled). The actual cost is infinity as long as there is 1 blocked gap, and the number of gaps on the field must be finite, so it underestimates. On the optimal path, this heuristic returns 0, which always underestimates. However, if one assumes that blocked gaps may still be filled, then h_4 is inadmissible since a single piece may fill more than 1 blocked gap.

(b) With reference to the heuristics defined in Part **(a)**, select all the following statements that **are True**. If you feel that none of the statements are True, select the option "*None of these options are True*". For each option, briefly, but clearly, explain why it is True/False.

- ***max***(h_1, h_2) is admissible.
- ***min***(h_2, h_3) is admissible.
- ***max***(h_3, h_4) is inadmissible.
- ***min***(h_1, h_4) is admissible.
- None of these options are True.

Solution: Only "***max***(h_3, h_4) is inadmissible" and "***min***(h_1, h_4) is admissible" are ***True***.

- ***max***(h_1, h_2) is admissible: **False**. The maximum of an admissible heuristic and an inadmissible heuristic is inadmissible since we would pick the higher inadmissible value for some states.
- ***min***(h_2, h_3) is admissible: **False**. While is it possible that the minimum of two inadmissible heuristics may still be admissible, this is not generally true since is it trivial to construct a counterexample where both heuristics are inadmissible for the same state, which would cause the minimum of both heuristics at such a state to also be inadmissible.
- ***max***(h_3, h_4) is inadmissible: **True**. The maximum of an inadmissible heuristic and some another heuristic is inadmissible since we would pick the higher inadmissible value for some states.
- ***min***(h_1, h_4) is admissible: **True**. The minimum of two heuristics when either one of them is admissible, is admissible since we would always choose the smaller (admissible) value.

(c) With reference to the heuristics defined in Part **(a)**, select all the following statements that **are True**. If you feel that none of the statements are True, select the option "*None of these options are True*". For each option, briefly, but clearly, explain why it is True/False.

- h_1 dominates h_2.
- h_2 dominates h_4.
- h_3 does not dominate h_2.
- h_4 does not dominate $h_2/2$.
- None of these options are True.

Solution: Only "h_1 dominates h_2" and "None of these options are True" are ***False***.

- h_1 dominates h_2: **False**. Admissible heuristics cannot dominate inadmissible heuristics.
- h_2 dominates h_4: **True**. The number of blocked gaps has to be ≤ the number of gaps since blocked gaps is a subset of gaps.
- h_3 does not dominate h_2: **True**. Consider the state n, where a horizontal (cyan) Tetrimino piece is required to complete a row; we have: $h_3(n) = 1 < h_2(n) = 4$.
- h_4 does not dominate $h_2/2$: **True**. Consider the initial state where there are 0 blocked gaps but many gaps.

2. Pac-Man is a maze chase video game where the player controls the eponymous character through an enclosed maze. The objective of the game is to eat all the dots placed in the maze while avoiding the four ghosts that pursue him. (Source: Wikipedia)

   Consider a simplified model of the game, where we exclude the ghosts from the game. Also, the player wins when all the dots (i.e., pellets) are eaten. We model the Pac-Man game as such:

   - **State representation**: The position of Pac-Man within the grid at any point in time and the positions of the remaining (i.e., uneaten) pellets.
   - **Initial State**: A grid that is entirely filled with pellets except at Pac-Man's starting position.
   - **Goal State**: A grid with no (uneaten) pellets remaining.
   - **Action**: Moving up, down, left, or right.
   - **Transition Model**: Updating the position of Pac-Man and eating (i.e., removing) the pellet at this new position (if a pellet exists there).
   - **Cost function**: 1 for each action.

   Consider the following heuristics.

   - h_1: The number of pellets left on the grid.
   - h_2: The number of pellets left on the grid + the minimum among all the Manhattan distances between each remaining pellet and the current position of Pac-Man.
   - h_3: The Maximum among all Manhattan distances between each remaining pellet and the current position of Pac-Man.
   - h_4: The average over all Euclidean distances between each remaining pellet and the current position of Pac-Man.

   Determine the admissibility of each of the above heuristics. Provide justifications.

   Solution:

   - h_1: **Admissible**. At any point in time, the number of pellets will always be less than or equals to the number of moves that Pac-man has taken. The best case (i.e., lowest-cost) scenario occurs when at every move that Pac-man makes, he consumes a pellet. Hence, actual cost = total number of pellets.
   - h_2: **Inadmissible**. Consider a state, n, when there is only 1 pellet left and Pac-man is 1 step away from that final pellet. Here, we have $h_2(n) = 1 + 1 = 2$. However, the actual cost to the goal, $h^*(n) = 1$. Therefore, h_2 overestimates at state n.
   - h_3: **Admissible**. Let the Manhattan distance between the position of Pac-man and the position of a particular pellet be $x + y$ (where x corresponds the difference along the x-axis, and y corresponds to the difference along the y-axis). The maximum among these Manhattan distances ensures that there is no overestimation since the best case (i.e., lowest-cost) scenario occurs when all the other pallets are along the optimal path to the most distant pellet.

- h_4: **Admissible**. Euclidean distance is similar to Manhattan distance, but with a further relaxation on the constraint of Pac-man's movement. The straight-line distance represented by Euclidean distance is an underestimation of the $x + y$ movement that Pac-man actually uses (i.e., note the triangular inequality between Euclidean and Manhattan distances).

3. Extending from the previous question (i.e., Question **2**), compare the dominance of the admissible heuristics that you have selected from the previous question. Justify your answer.

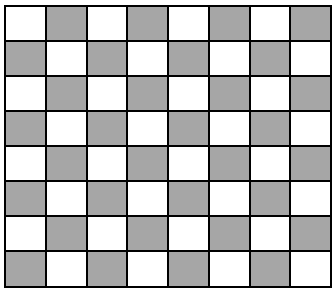
   For this question, assume that dominance does not assume admissibility.

   Solution:

   - h_3 dominates h_4. Euclidean distance is a more relaxed constraint as compared to Manhattan distance. At any state, n, $h_3(n) \geq h_4(n)$.
   - h_1 neither dominates or is dominated by either h_3 or h_4. Consider the following examples.
     - **Case 1**: Consider the state n, where there is one pellet remaining and Pac-man is 10 units away from it. Here, we have $h_1(n) = 1$, $h_3(n) = 10$, $h_4(n) = 10$.
     - **Case 2**: Consider the state n, where there are four pellets remaining, with each being 1 unit away from Pac-man – i.e., 1 unit above, 1 unit beside (left and right), and 1 unit below. Here, we have $h_1(n) = 4$, $h_3(n) = 1$, $h_4(n) = 4/4 = 1$.

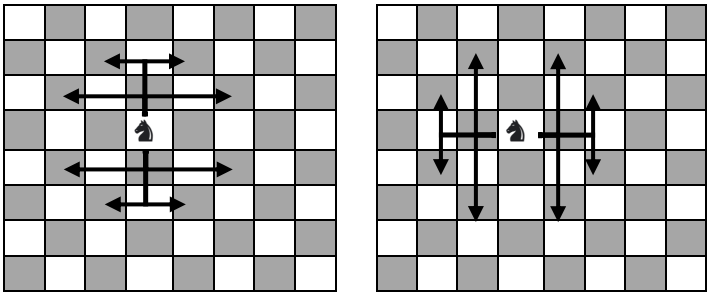
4. Consider the following puzzle that is played out on a Chess board.

   Chess is typically played on an 8-by-8 board, which is depicted below.



   However, assume that the current puzzle is played on a Chess board of infinite size.

   A Knight (♞) is a Chess piece that moves in an "L" shape. More precisely, the Knight piece can move two cells vertically followed by one square horizontally, or two cells horizontally followed by one cell vertically. The Knight piece's movement is summarised in the figures below.



In this puzzle, a Knight starts at a given Chess board cell, (x_s, y_s), and must find the ***shortest path*** to a specified *goal* cell at (0, 0) using the **A* Search** algorithm.

Do note that the Chess board in this puzzle will contain several obstacles. The Knight may **NOT** occupy any square that is occupied by an obstacle, although it may jump over the obstacles. For example, a Knight is allowed to move from cell (x, y) to cell $(x+2, y+1)$ even if square $(x+1, y)$ contains an obstacle.

Assume that the ***cost*** of any action taken by the Knight is equal to 1.

Further, assume that **two heuristics**, h_1 and h_2, have been defined.

The heuristic, h_1, is defined as follows.

- $h_1(n) = 0$ if $n = goal$, where n and $goal$ are coordinates on the Chess board
- $h_1(n) = 1$ if $(n \neq goal) \wedge (\textbf{\textit{ManhattanDistance}}(n, goal) \% 2 = 1)$
- $h_1(n) = 2$ if $(n \neq goal) \wedge (\textbf{\textit{ManhattanDistance}}(n, goal) \% 2 = 0)$

The heuristic, h_2, is defined as follows.

- $h_2(n) = \lceil \textbf{\textit{ManhattanDistance}}(n, goal) / 3 \rceil$

Note that the **Manhattan distance** (i.e., the output of the ***ManhattanDistance*** function) between two cells, (x_1, y_1) and (x_2, y_2), is given by $|x_2 - x_1| + |y_2 - y_1|$.

Consequently, it should be noted that h_1 is admissible since every Knight's move will change the Manhattan distance to the goal from odd to even, or from even to odd. The heuristic h_2 is also admissible since every Knight's move covers a Manhattan distance of 3, but not every cell that has a Manhattan distance of 3 away may be reached via a Knight's move.

(a) Prove or disprove the **admissibility** and **consistency** of the following heuristic.

$$h_3(n) = h_1(n) + h_2(n)$$

Solution: The heuristic h_3 is **not admissible**. Consider the following counterexample.

Let n correspond to a state where a Knight is at square (2, 1). Consequently, we have $h^*(n) = 1$. However, we have $h_3(n) = h_1(n) + h_2(n) = 1 + 1 = 2$. Since $h_3(n) > h^*(n)$, h_3 is not admissible.

Since h_3 is not admissible, h_3 is **not consistent**.

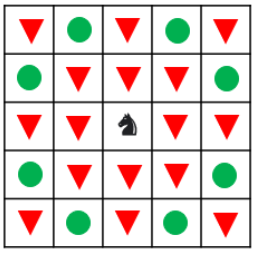
(b) Let the **Chebyshev distance** between two cells, (x_1, y_1) and (x_2, y_2), be defined by $\textbf{\textit{max}}(|x_2 - x_1|, |y_2 - y_1|)$. Assuming that the function ***ChebyshevDistance***(a, b) determines the Chebyshev distance between cells a and b, prove or disprove the **admissibility** and **consistency** of the following heuristic.

$$h_4(n) = \lceil \textbf{\textit{ChebyshevDistance}}(n, goal) / 2 \rceil$$

Solution:

Proof of admissibility

- Every Knight's move under h_4 covers a Chebyshev distance of 1 or 2, whereas every actual move that a Knight may make under the original rules only covers a subset of destination cells have a Chebyshev distance of 2. We loosen the Knight's move constraint so that every square a Chebyshev distance of 1 or 2 away can be reached in 1 move. Refer to the diagram on the right. The cells containing circles depict where the Knight may move to under the original rules. However, under the relaxed rules, the Knight may also move to the cells containing triangles.



- With this relaxed Knight move constraint, every cell of Chebyshev distance $2k$ away from the goal can be reached in k moves, where $k \subset \mathbb{Z}+$. Every cell of distance $2k+1$ away from the goal requires at least $k+1$ moves to reach.
- The above is a mathematically description of the h_4 heuristic.
- As this heuristic **represents a relaxation** of the Knight's move constraint, this would not overestimate the true number of moves required to reach the goal for any square, and hence the heuristic **is admissible**.

Proof of consistency

- Under the relaxation given by h_4, **every Knight's move covers a Chebyshev distance of 1 or 2**, with all moves made having a uniform cost of 1 – i.e., each move between a parent and its successor has a cost of 1. (Note the distinction between distance and cost.)
- Given a state n', which is a successor of state n. The following general cases are possible.
  - A. ***ChebyshevDistance***$(n, goal)$ – ***ChebyshevDistance***$(n', goal) = 2$.
    - Given the above, we have $h_4(n) - h_4(n') = \lceil a / 2 \rceil - \lceil (a - 2) / 2 \rceil = 1$, where $a =$ ***ChebyshevDistance***$(n, goal)$. (Assume the same about a below.)
    - Consequently, $cost(n, n') + h_4(n') = 1 + (h_4(n) - 1) = h_4(n)$.
  - B. ***ChebyshevDistance***$(n, goal)$ – ***ChebyshevDistance***$(n', goal) = 1$.
    - Here, we have $h_4(n) - h_4(n') = \lceil a / 2 \rceil - \lceil (a - 1) / 2 \rceil = 1$ when a is odd, or else when a is even, we have $h_4(n) - h_4(n') = \lceil a / 2 \rceil - \lceil (a - 1) / 2 \rceil = 0$.
    - Consequently, when a is odd, we have the same outcome as Case (A).
    - And, when a is even, $cost(n, n') + h_4(n') = 1 + h_4(n) > h_4(n)$.
  - C. ***ChebyshevDistance***$(n, goal)$ – ***ChebyshevDistance***$(n', goal) = 0$.
    - Here, we have the same outcome as Case (B) above, when a is even.
  - D. ***ChebyshevDistance***$(n, goal)$ – ***ChebyshevDistance***$(n', goal) = -1$ or -2.
    - When the result is -2, we have $h_4(n) - h_4(n') = \lceil a / 2 \rceil - \lceil (a + 2) / 2 \rceil$, which results in $cost(n, n') + h_4(n') = 1 + (h_4(n) + 1) > h_4(n)$. And when the result is -1, we similarly have $cost(n, n') + h_4(n') > h_4(n)$.

In **all the possible cases**, we see that the **consistency constraint** $h_4(n) \leq cost(n, n') + h_4(n')$ **holds**.

(c) Define an admissible heuristic h_5 that dominates h_4. The heuristic must be defined mathematically, and given an infinite board, must differ from h_4 in an infinite number of cells. Provide the justification that (1) h_5 is admissible, (2) h_5 is dominant over h_4, and (3) h_5 differs from h_4 in an infinite number of cells (i.e., you should show that h_5 does not, trivially, only improve on h_4 for a finite number of points). *Note that your heuristic must not be the (abstract) optimal heuristic,* h^* *or any function over it.*

Solution: $h_5 =$ ***max***(h_2, h_4)

Proof of admissibility

- As h_2 and h_4 are both admissible, for any state n, ***max***$(h_2(n), h_4(n)) \leq h^*(n)$.

Proof of dominance

- Given that $h_2 \neq h_4$, for any state n, ***max***$(h_2(n), h_4(n)) \geq h_2(n) \wedge$ ***max***$(h_2(n), h_4(n)) \geq h_4(n)$.
- Therefore, for any state p, where $h_2(p) > h_4(p)$, ***max***$(h_2(p), h_4(p)) > h_4(p)$, while for all other states, ***max***$(h_2(p), h_4(p)) = h_4(p)$. Thus, $h_5 =$ ***max***(h_2, h_4) dominates h_4.

Proof of h_4 and h_5 differing for infinitely many states

- Consider the cells in set S, which have coordinates in the form $(6k, 6k)$, where $k \subset \mathbb{Z}$.
- Here, for any $s \in S$, we have $h_4(s) = 3k$, while $h_5(s) =$ ***max***$(3k, 4k) = 4k$. As k ranges from $(-\infty, \infty)$, h_4 and h_5 differ in infinitely many cells.