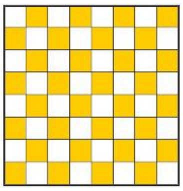
G Weiss, Dept. Data Science and Knowledge Engineering, Maastricht University

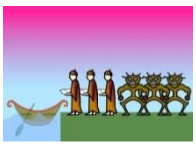
→ Uninformed Search

TASK UIS1 – "Uninformed Search – Queens, Missionaries and Cannibals"

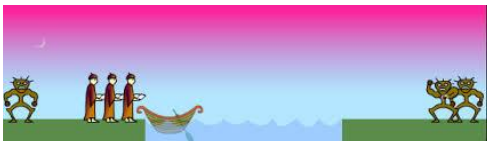
There are some very famous examples of search problems, one of which is the *8-queens problem*. Its goal is to place eight queens on a chessboard in such a way that no queen attacks any other queen.



Another very well-known one is the *missionaries and cannibals problem*:
"*Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to carry everybody to the other side, without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place*."
Thereby "place" means (i) left side of the river (including bank and – if on the left side – boat) and (ii) right side (bank and – if on the right side – boad).



…



… ?

a) Represent both search problems by specifying initial/goal state, goal test, successor function, and path costs.

b) Solve them with the depth-first algorithm and the breadth-first algorithm. Which one do you think is more adequate (given your search problem representation), and why?

<u>Background material:</u> Lecture slides.

→ Informed Search

TASK "InfS1 – "n-queens"

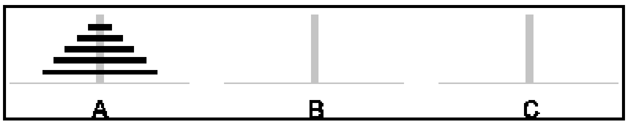
Consider the *n*-queens problem (i.e., the problem of placing *n* queens on a *n*x*n* chessboard so that no queen attacks any other). Assume we want to solve this with an informed tree search, that is, with a search algorithm that employs a heuristic that estimates the value of any possible board state.

(i) Propose such a heuristic and briefly explain the underlying idea.

(ii) Discuss your heuristic in terms of admissibility, if possible. (Recall: A heuristic is said to be admissible if it never overestimate the remaining number of necessary moves.)

(iii) Does your heuristic have any shortcomings? Justify your answer.

TASK "InfS2 – Towers of Hanoi"



Consider the Towers of Hanoi problem. The Tower of Hanoi problem is to move a set of *n* disks of different sizes from a start peg to a goal peg, using a third peg for temporary storage; disks are moved one at a time, and a larger disk cannot rest on a smaller one. (See http://en.wikipedia.org/wiki/File:Tower_of_Hanoi_4.gif for an animated solution for *n* = 4. The figure illustrates this problem for n = 5; "A" is the start peg and "B" and "C" are the temporary-storage peg and the goal peg.)

(i) To get familiar with the Towers of Hanoi problem, draw the complete search space for *n* = 3.

(ii) Propose a heuristic for this problem and briefly explain the underlying idea.

(iii) How good is your heuristics, what are its shortcomings (if any)?

<u>Background material:</u> Lecture slides.

→ Local Search

TASK "LOCS1 – Genetic Algorithm and Queens Problem"

Assume the Genetic Algorithm shall be used to solve the 8-queens problem. Specify a representation scheme, genetic operators and a fitness function. Discuss pros and cons of your scheme and operators.

TASK "LOCS2 – Hill-Climbing and TSP"

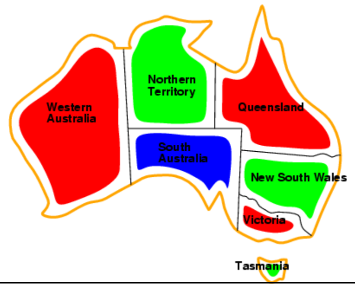
Devise a hill-climbing approach to solve the traveling salesperson problem (TSP): *Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once?* (There are several variants of the TSP; perhaps the most famous version of the TSP is that the route has to return to the origin city. TSP is a so-called NP-hard problem, that is, there is no fast solution known and its complexity grows superpolynomially with the number of cities.)

TASK "LOCS3 – Genetic Algorithm and TSP"

Think about solving the TSP with a Genetic Algorithm. How could a representation and the basic operators (mutation and recombination) look like? Compare the Genetic-Algorithm-based search with the Hill-Climbing-based search, what are main differences?

Background material: Lecture slides.

AI – Basic Exercises
G Weiss, Dept. Data Science and Knowledge Engineering, Maastricht University



→ Constraint Satisfaction

TASK "ConS1 – Cryptarithmetics"

Solve the cryptarithmetic problem you know from the lecture (i.e., "TWO + TWO = FOUR") by hand. First formulate all constraints and then apply backtracking, forward checking, and the "minimum remaining value" and "least constraining value" heuristics.

TASK "ConS2 – Constraints for ToH and n-Queens"

Consider *(i)* the Towers of Hanoi problem and *(ii)* the n-queens problem. Formulate the constraints for this problem as detailed as possible (so that they can be "directly implemented" in a software program).

TASK "ConS3 – Ternary Constraints"

Show how a single ternary constraint such as "A + B = C" can be turned into three binary constraints by using an auxiliary variable.

TASK "ConS4 – Einstein's Puzzle"

Consider the following puzzle, which is also known as Einstein's Puzzle: *There are five houses of different colors next to each other on the same road, in each house lives a man of a different nationality, and every man has his favorite drink, his favorite brand of cigarettes, and keeps pets of a particular kind. Some hints:*

1. *The Englishman lives in the red house.*
2. *The Swede keeps dogs.*
3. *The Dane drinks tea.*
4. *The green house is just to the left of the white one.*
5. *The owner of the green house drinks coffee.*
6. *The Pall Mall smoker keeps birds.*
7. *The owner of the yellow house smokes Dunhills.*
8. *The man in the center house drinks milk.*
9. *The Norwegian lives in the first house.*
10. *The Blend smoker has a neighbor who keeps cats.*
11. *The man who smokes Blue Masters drinks bier.*
12. *The man who keeps horses lives next to the Dunhill smoker.*
13. *The German smokes Prince.*
14. *The Norwegian lives next to the blue house.*
15. *The Blend smoker has a neighbor who drinks water.*

Question: Who keeps fish?

Formulate this puzzle as a constraint satisfaction problem and try to solve it.

<u>Background material:</u> Lecture slides.

G Weiss, Dept. Data Science and Knowledge Engineering, Maastricht University

→ Adversarial Search

TASK "AdS1 – Game Tree"

Consider the following game (taken from Russell&Norvig)

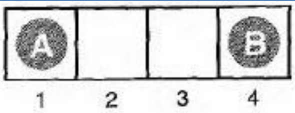


Figure 6.14 The starting position of a simple game. Player A moves first. The two players take turns moving, and each player must move his token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then a player may jump over the opponent to the next open space if any. (For example, if A is on 3 and B is on 2, then A may move back to 1.) The game ends when one player reaches the opposite end of the board. If player A reaches space 4 first, then the value of the game to A is +1; if player B reaches space 1 first, then the value of the game to A is −1.

and address these subtasks:

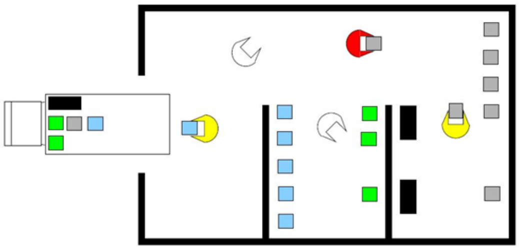
(i) Draw the complete game tree. Use these conventions:
- Put each terminal state in a box and put loop states (i.e., states that already appear on the path to the root) in double square boxes.
- For each terminal state write its game value in a circle. Since it is not clear how to assign values to loop states, declare its game value as "?".

(ii) Mark each node with its backed-up minimax value (also in a circle). Explain how you handled the "?" values and why.

(iii) Is the standard minimax procedure suited for this game?

Background material: Lecture slides.

→ Components of Coordination



TASK "CC1 – Analysis of an IS Application"

Consider the Docking Station application from an engineering perspective.

a) Identify and specify (in a precise/formal style)

- (i) main goal(s) to be achieved,
- (ii) main activities needed for achievement,
- (iii) actors (= agents), and
- (iv) interdependencies among the activities.

b) Reflect on the assignment of the identified activities to actors by addressing these questions

- (i) Why is this assignment in general non-trivial?
- (ii) How can this assignment be done in principle?

<u>Background material: </u>Lecture slides on „Coordination", esp. see slides 6f.

G Weiss, Dept. Data Science and Knowledge Engineering, Maastricht University

→ Voting

TASK "Voting1 – Iterated Borda"

Problem: Standard Borda violates Independence of Irrelevant Alternatives (IIA). An interesting question thus is whether Borda voting can be modified so that IIA is not longer violated by the modified variant.

Idea: Iterated version of Borda voting (i.e., run multiple rounds of standard Borda and remove the least popular option in each round).

Question: Does iterated Borda solve the IIA issue? Justify your answer.

TASK "Voting2 – Strategic (Tactical, insincere) Voting"

Strategic voting = A voter supports an option other than their sincere (true) preference in order to prevent an undesirable outcome.

Scenario: Assume that a "society" consisting of five agents (A1, …, A5) uses Borda voting to come to an agreement on four options (A, B, C and D), where the true preferences of the agents are as follows:

	A1	A2	A3	A4	A5
4	C	B	C	B	B
3	A	D	D	D	C
2	D	C	A	C	D
1	B	A	B	A	A

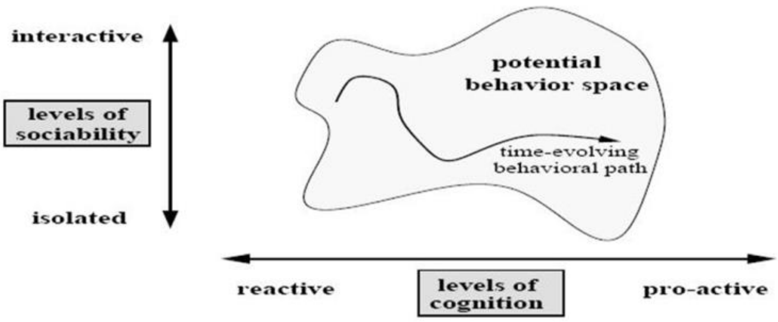
Question: Are there possibilities for strategic voting? (Hint: also think about "strategic voting in response to strategic voting".)

<u>Basic background material: </u>Lecture slides on „Coordination".

→ Agent Architectures 1a

TASK "AA1 – "Behavioral Space of Agent Architectures"

Qualitatively characterize the potential behavioral space of PRS, IRMA, GRATE*, INTERRAP in terms of the „levels-of-sociability – levels-of-cognition coordinate system".



TASK "AA2 – Application → Architecture"

Consider the three application domains

1) **Chess**,
2) **Mars exploration**, and
3) **Robot soccer**.

Are the architectures discussed in the course suited for any of these applications? Briefly indicate the main reason(s) for your answer.

<u>Basic background material: </u>Lecture slides.