Introduction to Artificial Intelligence (IAI) BSc and MSc Exam

IT University of Copenhagen

17 May 2021

This exam consists of 13 numbered pages with 5 problems. Each problem is marked with the weight in percent it is given in the evaluation. **Notice that the first version of problem 4 only is to be solved by Bachelor (BSc) students, while the second version only is to be solved by Master (MSc) students**.

- Ensure that you have a quiet, comfortable, and undisturbed work environment.
- Use notations and methods that have been discussed in the course.
- Make appropriate assumptions when a problem has been incompletely specified.

Good luck!

Problem 1: Heuristic Search (20%)

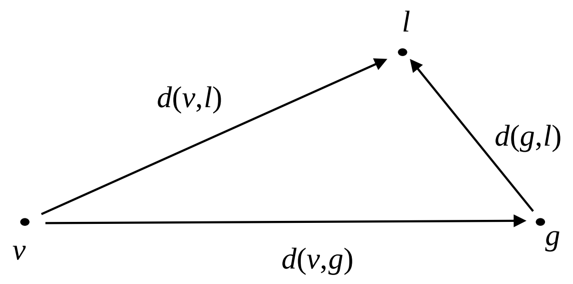
In route planning it is possible to compute strong heuristics based on landmarks. Consider a road network represented by a directed graph (V, E). The set of vertices V is the intersections in the road network. The set of edges E is the road sections. Each edge $e \in E$ is represented by a triple $e = (v, v', c)$, where $v, v' \in V$ are the two intersections the road section connects and $c > 0$ is its length.

The states of the route planning problem is the set of vertices V. The actions is the set of edges E. An action of an edge (v, v', c) is to drive from v to v' and it has cost c. The initial state is some intersection $v_0 \in V$ and the goal is to find a route to some intersection $g \in V$.

The landmarks L is a subset of intersections $L \subset V$ that are geographically evenly distributed in the road network. An initial one-time effort is spent on computing the shortest distance in the road network $d(v, l)$ from any intersection $v \in V$ to any landmark $l \in L$. The result is used to compute heuristics for general route planning problems.

Consider the shortest distance in the road network $d(v, g)$ from some intersection v to the goal intersection g. Notice that unless g is a landmark, this distance has not been precomputed.

a) Use the triangle below to show that $d(v, g) \geq d(v, l) - d(g, l)$ for any landmark $l \in L$.



Let a heuristic function $h(v)$ be defined by $h(v) = max_{l \in L}\{d(v, l) - d(g, l)\}$.

b) A valid heuristic function requires $h(g) = 0$. Show that this requirement is fulfilled.

c) A valid heuristic function also requires $h(v) \geq 0$. Argue that even for a large set of evenly distributed landmarks it is possible for this requirement to be broken.

In the remainder we assume that $h(v) \geq 0$.

d) Why is $h(v)$ an admissible heuristic?

e) Do you expect $h(v)$ to be a stronger heuristic than the straight line heuristic $h_{SLD}(v)$ (why/why not)?

Problem 2: Local Search (25%) [1]

Assume that a student during a work week has seven lectures A, B, C, D, E, F, and G. Each lecture requires a single preparation activity that the student must plan. The name of a preparation activity is the lowercase letter of the lecture. Your task is to make a timetable of the preparation activities.

A timetable is feasible if the following constraints are fulfilled:

1. There are five days in a work week: Mon, Tue, Wed, Thu, and Fri.
2. Each day has four consecutive time slots.
3. The duration of each activity is one time slot.
4. The student can at most work on one activity in a time slot.
5. The time slots of the lectures are given and fixed.
6. A preparation activity must take place before the lecture and in the same week.

Assume that the student has found the feasible timetable FT shown below. Notice that the slots that can hold preparation activities are numbered 1 to 13.

Mon	Tue	Wed	Thu	Fri
a 1	*B*	2	*e* 3	*g* 4
A	5	*C*	6	*F*
c 7	*d* 8	*D*	*E*	*G*
b 9	10	11	*f* 12	13

A feasible timetable has a cost that reflects the student's preferences. The cost is defined as the sum of the following terms:

1. Work early: there is a cost of two for each activity scheduled in the last time slot of a day (e.g., b causes an additional cost of two in FT).
2. Consolidate activities: a day has a cost of one if it has a period of inactivity between activities (e.g., Tue causes an additional cost of one in FT).
3. Prepare close to lectures: there is a cost of one for each day a preparation activity is before its lecture (e.g., c causes an additional cost of two in FT).

[1]This problem is inspired by a BSc thesis project by M.R. Almind and R.K. Petersen, 2021.

a) What is the cost of *FT* ?

We will improve *FT* with local search using a swap neighborhood. A swap can happen between any pair of the 13 time slots that can hold preparation activities. The time slots must be different and at least one of them must currently hold a preparation activity. The result of a swap is that the time slots exchange assigned activities. A swap is only possible if the resulting timetable is feasible. Let $x \leftrightarrow y$ denote a swap between time slot x and y. The result of $2 \leftrightarrow 3$ in *FT* is that time slot 2 is assigned to e and time slot 3 becomes unoccupied. Swap $1 \leftrightarrow 7$ is impossible since a must happen before A. The swap neighborhood of a timetable is defined by all the possible swaps that can be done on it.

b) Write the neighborhood of *FT* as a list of pairs ($i \leftrightarrow j$, *cost*), where *cost* is the cost of the resulting timetable.

Recall the pseudocode of the simulated annealing algorithm for a maximization problem shown below.

```
1. function SIMULATED-ANNEALING(problem,schedule) returns a solution
2.     current ← MAKE-NODE(problem.INITIAL-STATE)
3.     for $t = 1$ to $\infty$ do
4.         $T$ ← schedule($t$)
5.         if $T = 0$ then return current
6.         next ← a random selected neighbor of current
7.         $\Delta E$ ← next.VALUE − current.VALUE
8.         if $\Delta E \geq 0$ then current ← next
9.             else current ← next only with probability $e^{\Delta E/T}$
```

c) Re-write the pseudocode of the simulated annealing algorithm to change it to solve a minimization problem. You only have to re-write the lines that change.

Assume that your corrected algorithm uses a temperature schedule defined by *schedule*(1) = 3, *schedule*(2) = 2, *schedule*(3) = 2, and *schedule*(4) = 0. Further, assume that the random selected neighbor of *current* in line 6 is the result of swap $10 \leftrightarrow 12$, $6 \leftrightarrow 12$, $2 \leftrightarrow 3$ in iteration 1, 2, and 3 of the algorithm, respectively. Finally, assume that only neighbors with $e^{\Delta E/T}$ larger than 0.55 are accepted.

d) Write the value of T, *next*, and $\Delta E/T$ for each iteration of your corrected simulated annealing algorithm. Write also the value of $e^{\Delta E/T}$ and *current* when relevant. Assume that *problem*.INITIAL-STATE is the feasible timetable FT found by the student.

Prof. Smart wants to show that the swap neighborhood is complete. Assume that FT_1 and FT_2 are two arbitrarily chosen feasible timetables for the same set of lectures with fixed time slots. Prof Smart must show that there is a series of possible swaps that transforms FT_1 to FT_2.

e) Assume that preparation activity p is at time slot s_1 in FT_1 and at time slot s_2 in FT_2, where s_1 happens chronologically before s_2. Explain why the swap $s_1 \leftrightarrow s_2$ always is possible in FT_1.

The question is what to do if preparation activity p is at time slot s_1 in FT_1 and at time slot s_2 in FT_2, where s_1 happens chronologically after s_2. Prof. Smart claims that since the swap $s_2 \leftrightarrow s_1$ is feasible in FT_2, the series of swaps that transforms FT_1 to FT_2 can be found as follows. Start by transforming FT_2 by iteratively swapping any preparation activity that has an earlier time slot than in FT_1 to the time slot it has in FT_1. Call the resulting timetable FT'_2. Every preparation activity in FT_1 must now be at the same or an earlier time slot than in FT'_2. Transform FT_1 to FT'_2 by iteratively swapping any preparation activity that has an earlier time slot than in FT'_2 to the time slot it has in FT'_2. The series of swaps transforming FT_1 to FT_2 is the ones just performed on FT_1 followed by the ones done on FT_2 in reverse order and with every swap reversed.

f) Is Prof. Smart's claim correct (why/why not)?

Problem 3: Logic and ROBDDs (30%)

Mastermind is a code-breaker game for two players. One player is the *codemaker*, the other is the *codebreaker*. The game is played using a decoding board with a shield at one end covering a code. In the version of the game that we will study, the code is made with black and white *code pegs*. The code consists of three of these placed in a row of holes behind the shield. The game starts by the codemaker making a code behind the shield. The codebreaker is not allowed to see it. The codebreaker tries to guess the code. Each guess is made by placing a row of code pegs on the decoding board. Once placed, the codemaker provides feedback by placing small black and white *key pegs* in the three holes to the left of the row with the guess. A black key peg is placed for each code peg from the guess which is correct in both color and position. A white key peg indicates the existence of a correct color of a code peg but placed in the wrong position. Consider the state of the game shown below after the codebreaker has made two guesses.

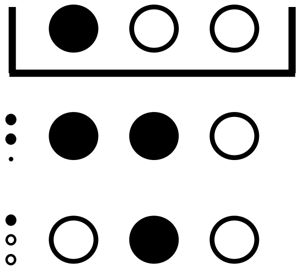


Figure 1: Example game state

The code is the top row behind the indicated shield. The first guess is the next row. It gets two black key pegs shown to the left. The reason is that the left and right code peg in the guess have the correct color. The middle black code peg, on the other hand, has a wrong color. The second guess gets one black key peg since the right white code peg in the guess have the correct color. It also gets two white key pegs. The reason is that the left and middle code peg by swapping position can get the right color without using a position currently associated with a black key peg. Notice that the ordering of the key pegs is irrelevant.

We will use propositional logic to reason about this game. Let the Boolean variables x_1, x_2, and x_3 represent the left, middle, and right code peg of the code made by the codemaker. Similarly, let the Boolean variables y_1, y_2, and y_3 represent the left, middle, and right code peg of a code guess made by the codebreaker. Assume that a peg variable is assigned to *True* if the peg is black and *False* if the peg is white.

a) Write a Boolean expression on x_1, x_2, and x_3 that is true exactly for the code made by the codemaker in the game state shown above.

Now, consider any code made by the codemaker and any guess made by the codebreaker.

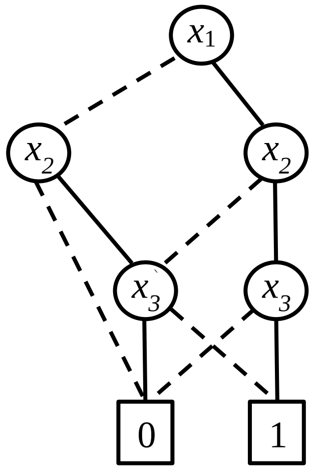
b) Write a Boolean expression on x_1, x_2, x_3, y_1, y_2, and y_3 that is true if and only if the codebreaker has guessed the code (use $p \Leftrightarrow q$ to express that two Boolean variables have the same truth value).

c) Write a Boolean expression E on x_1, x_2, x_3, y_1, y_2, and y_3 that is true if and only if the feedback from the codemaker is two black key pegs.

d) Formally prove that $(\mathit{True} \Leftrightarrow x) \equiv x$ and $(\mathit{False} \Leftrightarrow x) \equiv \neg x$.

e) Assume that the codebreaker's guess is the first guess made in the game state shown in Figure 1. Use logical equivalences to simplify the expression $E[\mathit{True}/y_1, \mathit{True}/y_2, \mathit{False}/y_3]$ representing this situation. What codes in addition to black-white-white are possible?

Prof. Smart claims that the expression $E[\mathit{True}/y_1, \mathit{True}/y_2, \mathit{False}/y_3]$ is represented by the ROBDD below.



f) Show that Prof. Smart's claim is correct. Write and explain the expression that the ROBDD represents and prove that it is logically equivalent to $E[\mathit{True}/y_1, \mathit{True}/y_2, \mathit{False}/y_3]$.

Consider the feedback of the second guess of the codebreaker shown in Figure 1.

g) Which codes of the codemaker can get this feedback?

h) Draw an ROBDD with variable order $x_1 \prec x_2 \prec x_3$ that represents the set of codes you found in question g).

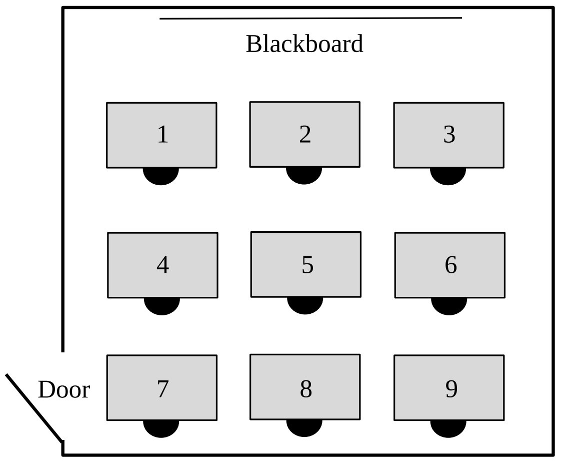
Assume that Prof. Smart's ROBDD and your ROBDD from question h) is represented by a multi-rooted ROBDD using variable order $x_1 \prec x_2 \prec x_3$.

i) Draw the multi-rooted ROBDD and write identifiers next to each internal node that is consistent with the multi-rooted ROBDD being made by a series of calls to Mk.

j) Use Apply to make the conjunction ($\wedge$) of the two ROBDDs using your multi-root representation. Draw the call tree of App as in the Apply example of Lecture 7. Indicate cache hits and draw the resulting multi-rooted ROBDD. How can your result be used by the codebreaker in the example game state shown in Figure 1?

Problem 4: FOR BSC STUDENTS ONLY (25%)

After the COVID-19 reopening, a class teacher faces the challenge of assigning pupils to tables in the classroom. There are five students Adam, Ben, Connie, David, and Eve. The classroom has nine numbered tables in three rows as shown below.



The rules are:

1. At most one pupil can sit at a table.
2. Connie must be close to the door either at table 4,5,7, or, 8.
3. Eve is near-sighted and must sit near the blackboard in the front row.
4. David and Adam must be separated by at least a complete row.
5. There can not be more than one girl or one boy per row.
6. Due to COVID-19, a table directly above, below, to the left, or to the right of an occupied table must be empty.

We will use constraint programming to solve this problem. Let the variables A, B, C, D, and E denote the table assigned to Adam, Ben, Connie, David, and Eve, respectively. The domain of each variable is $\{1, 2, \ldots, 9\}$. The constraints are defined by the six rules described above.

a) Which of the rules are unary constraints and why?

b) The remaining constraints can be formulated as binary constraints. Define each of these constraints formally and introduce notation as needed. As an example, a formalization of the last rule is:

$$(\alpha = i) \Rightarrow (\beta \neq j) \text{ forall } \alpha \in V, i \in D, \beta \in V \setminus \{\alpha\}, j \in Adj_i,$$

where $V = \{A, B, C, D, E\}$, $D = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and Adj_i is the set of any tables directly above, below, to the left, or to the right of table i.

We will use the following table to indicate possible values of the variables.

	1	2	3	4	5	6	7	8	9
A	✓	✓	✓	✓	✓	✓	✓	✓	✓
B	✓	✓	✓	✓	✓	✓	✓	✓	✓
C	✓	✓	✓	✓	✓	✓	✓	✓	✓
D	✓	✓	✓	✓	✓	✓	✓	✓	✓
E	✓	✓	✓	✓	✓	✓	✓	✓	✓

c) Remove check marks from the table corresponding to domain values that break node consistency.

d) Continue by also removing check marks of values that break arc consistency. Explain why the removed values break arc consistency. Be careful there might be less than you think.

We will use FORWARD-CHECKING-SEARCH and MAC-SEARCH to solve the CSP. Assume that the CSP has been made node and arc consistent before calling the algorithms.

e) Draw the call tree of RECURSIVE-FORWARD-CHECKING. Draw each internal call node of the tree as the state of the table of domain values after the inference step of the algorithm. Draw leaf nodes that returns *failure* as <failure>. Draw leaf nodes that find a result as <success>. Next to each arc in the tree, write the assignment of the variable selected in the call of the parent node associated with it. Assume that variables are selected in ascending lexicographical order and that domain values are assigned in ascending numerical order.

f) Draw the call tree of RECURSIVE-MAC using the same approach.

Problem 4: FOR MSC STUDENTS ONLY (25%)

After being expelled from the Garden of Eden, Adam and Eve played Rock-Paper-Scissors about the hard tasks they faced. Rock-Paper-Scissors is played by each person simultaneously showing a hand sign for either Rock (R), Paper (P), or Scissors (S). The outcome of the game is win, draw, or lose, with utility 1, 0, and -1, respectively. From Adam's point of view, the different combinations have the utilities shown in the table below.

		Adam		
		R	P	S
	R	0	1	-1
Eve	P	-1	0	1
	S	1	-1	0

Let x_1, x_2, and x_3 denote the probabilities that Adam plays rock, paper, and scissors, respectively. The expected utility U of the game for Adam depends on what Eve plays as shown in the table below.

Eve plays	Rock	Paper	Scissors
U	$x_2 - x_3$	$-x_1 + x_3$	$x_1 - x_2$

After some time, Adam learns that Eve plays Paper with probability $\frac{1}{2}$ and Rock and Scissors with probability $\frac{1}{4}$. He wonders which probabilities x_1, x_2, and x_3 he should choose to maximize his expected utility of the game in this situation. He realizes that he needs to find an optimal solution to the problem

$$\begin{array}{ll} \text{Maximize} & \frac{1}{4}(x_2 - x_3) + \frac{1}{2}(-x_1 + x_3) + \frac{1}{4}(x_1 - x_2) \\ \text{Subject to} & \qquad x_1 + x_2 + x_3 = 1 \\ & \\ & \qquad x_1, x_2, x_3 \geq 0. \end{array}$$

a) Explain why the problem is a linear program.

b) Write the problem in standard form.

c) Write the slack form of the linear program and explain why it is infeasible as an initial dictionary to the SIMPLEX algorithm.

d) Write the slack form of the auxiliary problem used by the two phase SIMPLEX algorithm.

e) Solve the first phase of the two phase SIMPLEX algorithm. Write all dictionaries produced by the algorithm.

f) Solve the second phase of the two phase SIMPLEX algorithm. Write all dictionaries produced by the algorithm.

g) What are the optimal probabilities of x_1, x_2, and x_3 for Adam and what is Adam's expected utilization of the game when using these probabilities?

Eve soon realizes that Adam has figured her out and changes her strategy. Adam reconsiders how to choose x_1, x_2, and x_3. No matter what choice he makes, Eve eventually will learn his probabilities. When she has done that, she will simply play the token that gives him least utilization. For that reason he must find an optimal solution to the problem

$$\begin{array}{ll} \text{Maximize} & min\{x_2 - x_3, -x_1 + x_3, x_1 - x_2\} \\ \text{Subject to} & \qquad x_1 + x_2 + x_3 = 1 \\ & \\ & \qquad\quad x_1, x_2, x_3 \geq 0. \end{array}$$

h) Show that this problem can formulated as a linear program. You do not have to write it in standard form (hint, you need to introduce an auxiliary variable).