Artificial Intelligence, Master degree - Exam 1 July 2019

Instructions:
Time: 70 min. Use of literature, notes and electronic devices is not allowed. Please state your answers short and clear, answering the questions directly to the point.

Oral exam: Friday, 5 July 2019 at 13 pm

1.
(a) For each of the algorithms A*, IDA*, RBFS and RTA*, state briefly their main advantages and disadvantages in terms of time and space complexity, and guarantees regarding optimality (quality) of solutions.
(b) What does the abbreviation RBFS mean? How would you briefly describe the relation between RBFS and A*?
(c) State the admissibility theorem for A*. State the variants of this theorem applied to the RBFS and IDA* algorithms? For which algorithms does the monotonicity of evaluation function **f** matter? Why does it matter?
(d) What does it mean »inheritance of backed up values F« in RBFS? Define the rules of inheritance (how are inherited values determined?).
(e) What is achieved by inheritance of F-values? How would the difference between RBFS and RBFS without inheritance be manifested during the execution of these algorithms? What would be the important difference between them?
(f) Could it happen that RBFS would return a solution with a lower cost than RBFS without inheritance; or the other way round? If the answer is YES, illustrate the answer with an example.

2. Consider the planning domain with four robots a, b, c and d moving in the 2x3 grid. The start state is shown below:

a 4	c 5	6
b 1	d 2	3

The rules of the domain are as usual: a robot can move into an unoccupied adjacent cell either vertically or horizontally. Any cell may contain one robot at the most. The relations used to specify the states of this domain are of the forms: at(Robot,Cell) and c(Cell) (Cell is empty).
The start state is specified by the following literals: at(a,4), at(b,1), at(c,5), at(d,2), c(3), c(6).
Actions are of the form: m(Robot,CellA,CellB). The goals of the plan are: at(a,1), at(b,4). That is, the robots a and b want to swap places.

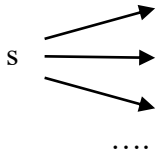
(a) How many time-optimal partially ordered plans (POPs) are there for this task? Specify one of these plans in the representation used by the POP planning method (set of actions and time constraints).

(b) The POP algorithm begins the search for a plan with the initial POP plan that only includes two virtual actions: { Start, Finish}. What are the open preconditions of this plan? What are all the immediate successor POP plans of this plan? For each of these successor plans, state the actions (in addition to Start and Finish).

(c) Assume that during execution the POP algorithm has constructed a partial POP plan that includes the two virtual actions, and one additional action m(a,2,1): that is{ Start, Finish, m(a,2,1)}. What are the causal links in this plan? What are the open preconditions of this plan? For each of these open preconditions, state at least one action that achieves the precondition. How many such relevant actions are there to be considered altogether?

(d) Give all time-optimal plans for this planning problem represented in the form used by the GRAPHPLAN method.

3. A problem of sequential decision making is specified by a state-transition function **delta**, and a reward function **r(s,s')** that determines the reward for each possible transition from a state **s** to next state **s'** (illustrated below). Let cumulative rewards be discounted by a factor **Gamma**. Assume that the system uses policy **Pi** which maps states to actions (action to be executed in a state **s** is **Pi(s)**). Let **U(s)** denote the utility of a state **s** when the system executes this policy **Pi**.



Possible transitions from **s** to successor states when **a** is executed

(a) Assume the system is deterministic, so that the function **delta(s,a)** specifies the transition from a state **s** to state **s'** when **a** is executed in **s**. Give the equation for computing the utilities $\mathbf{U^{Pi}(s)}$ for any **s**, in terms of the utility of the successor state of state **s**, and functions **delta** and **r** (that is: recursive relation for U-values). The form of the equation should be:

$U^{Pi}(s) = ... U^{Pi}(...) ...$

(b) Give the equation for the computation of the Q-values **Q(s,a)** for any **s** and action **a**, in terms of the Q-values of the successor state of state **s**, and functions **delta** and **r** (that is: recursive relation for Q-values). The form of the equation should be:

$Q(s,a) = ... Q(...) ...$

(c) Now let the system be **non-deterministic**, so that the function **delta(s,a)** specifies the probability distribution over all possible transitions from a state **s** to state **s'** when action **a** is executed. Assume **Pi** is an optimal policy, and that the optimal utilities of states are known. Give the equation for computing the values **Q(s,a)** in terms of the utilities $\mathbf{U^{Pi}}$ of the successor states of **s**

after **a**, and functions **delta** and **r** for this (non-deterministic) case. The form of the equation should be:

$Q(s,a) = \Sigma \ ... \ U^{Pi}(...) ...$

4. Consider a qualitative model of QSIM type. There are 4 variables in the system: X, Y, VX, VY. The landmarks for these variables are:

X: minf, x0, 0, inf
Y: minf, 0, y0, inf
VX, VY: minf, 0, inf

The constraints in the model are:

deriv(X, VX)
deriv(Y, VY)
plus(X, Y, zero:std) % Y + X = 0
VX = M_0^+(Y) % Monotonically inc. function with corresponding values (zero,zero)
VY = M_0^+(X) % Monotonically inc. function with corresponding values (zero,zero)

Initial values of X and Y in the start state at time t0 are: X(t0) = x0, Y(t0) = y0

(a) Determine all possible qualitative states of the system at time t0; that is qualitative magnitudes and directions of change of all the variables: X, Y, VX, VY.
(b) Determine all possible states of the system at time interval t0..t1.
(c) Determine all possible qualitative states of the system at time t1.
(d) Try to explain intuitively the mechanics of how this system works.

ANSWERS

1.
(a) Comparative advantages/disadvantages of algortihms can be summarised by the table:

	Time	Space	Optimality, cost of solutions
A*	+	-	+
IDA*	+/-	+	+/-
RBFS	(+)	+	+
RTA*	++	+	-

(b) RBFS = Recursive Best-First Search
RBFS tries to simulate A*, but uses (much) less space at the expense of time (repeated searches)

(c) Admissibility theorem: If for all nodes n, $h(n) \leq h^*(n)$, then A* is guaranteed to find a least-cost solution. The same holds for RBFS. Both A* and RBFS respect best-first order of expanding nodes during search. IDA* is affected by monotonicity of function f. If f is monotonic then IDA* also respects best-first order, otherwise best-first order is not guaranteed, so the solution returned by IDA* may differ from that by A*. However, for optimistic h, a least-cost solution is still guaranteed (so theorem still applies).

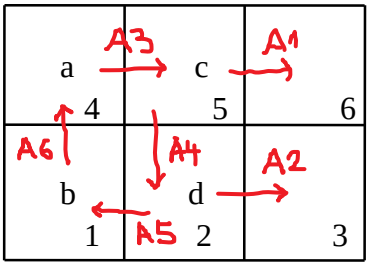
(d) Inheritance of F-values means that under some conditions a node's F-value may be "inherited" from the node's parent. The rule for determining F-values is: Let N be a node that has hust been expanded, and Ni is one of N's children. If f(N) < F(N) then F(Ni) = max(F(N),f(Ni)) else F(Ni) = f(Ni). The first branch of this if-then-else expression may thus result in inheritance.

(e) Inheritance sometimes enables more realistic (larger) F-values and thus speeds up search in RBFS by avoiding unnecessary repetition of searching parts of search space. Thus RBFS is typically faster than RBFS without inheritance.

(f) No, both versions return the same solution.

2.
(a) There are 4 time-optimal POP plans. One of them is: robots c and d move to the right, then robot a moves from 4 to 5 then to 2 then to 1, and in the meantime b moves up. A variation of this is that simultaneously with final move m(a,2,1) we have m(c,6,5) (of course the latter move is completely redundant, but it does not affect the execution time). This makes 2 plans. Another 2 plans are obtained as a variation of these, when b moves round from 1 to 2 then to 5 then to 4.

Consider the actions represented by red arrows:



One of POP plans is: Actions = {A1, A2, …, A6},
Order constraints = { A1<A3, A2<A4, A3<A6, A6<A5}

(b) Open preconditions: at(a,1) and at(b,4) both for action Finish.
There are 4 successor POP plans of plan {Start, Finish}. Each of these 4 plans contains one of the following actions in addition to Start and Finish:
m(a,2,1), m(a,4,1), m(b,1,4), m(b,5,4)

(c) Plan { Start, Finish, m(a,2,1)} contains causal link: causes(m(a,2,1), at(a,1), Finish)
Open preconditions in this plan are:
at(b,4) : Finish (i.e. at(b,1) is an open precondition for Finish)
This can be achieved by m(b,1,4) or m(b,5,4)
at(a,2) : m(a,2,1), which can be achieved by m(a,5,2) or 2 other actions
c(1) : m(a,2,1), which can be achieved e.g. by m(d,1,4) or 7 other actions
The number of all relevant actions for these preconditions is thus 2 + 3 + 8 = 13

(d) There are 8 time-optimal GRAPHPLAN plans. They are obtained by combining the following variations: (1) which robot (a or b) takes the longer route to its goal, (2) robots c and d move simultaneously or not, and (3) there is an extra redundant action by c or d at the end of the plan.
Three of these 8 plans are:
[[m(d, 2, 3), m(c, 5, 6)], [m(a, 4, 5)], [m(b, 1, 4), m(a, 5, 2)], [m(a, 2, 1)]]
[[m(c, 5, 6)], [m(d, 2, 3), m(a, 4, 5)], [m(b, 1, 4), m(a, 5, 2)], [m(a, 2, 1)]]
[[m(d, 2, 3), m(c, 5, 6)], [m(a, 4, 5)], [m(b, 1, 4), m(a, 5, 2)], [m(c, 6, 5), m(a, 2, 1)]]
…

3.
(a) $U^{Pi}(s) = r(s,s') + gamma * U^{Pi}(s')$ where s'= delta(s,a) and a = Pi(s)

(b) $Q(s,a) = r(s,s') + gamma * max[a'] Q(s', a')$ where s' = delta(s,a)

(c) $Q(s,a) = \Sigma_{s'} P(s'|s,a) * [r(s,s') + gamma * U^{Pi}(s')]$

4.	Time	X	Y	VX	VY
(a)	t0	x0/inc	y0/dec	pos/dec	neg/inc

(b)	t0..t1	x0..0/inc	0..y0/dec	pos/dec	neg/inc
(c)	t1	0/std	0/std	0/std	0/std

(d) Initially X < 0 and Y > 0. X “tries” to move towards Y, and Y towards X. They meet at X = Y = 0, and stay steady afterwards. All this can be illustrated as: