Intelligent Systems (Sheet #2)

Marius Gavrilescu

1. $w = 0$ performs UCS. $w = 1$ performs A*. $w = 2$ performs best-first search. Optimal for the first two, not necessarily optimal for the third.

   For $w \neq 2$, we have $f(n) = (2-w)(g(n) + \frac{w}{2-w}h(n))$, which is equivalent to an A* search with heuristic $\frac{w}{2-w}h(n)$. This heuristic is guaranteed to be admissible only if $\frac{w}{2-w} \leq 1 \iff w \leq 2 - w \iff w \leq 1$ and $\frac{w}{2-w} > 0$. So the search is guaranteed to get the right result when $w \in [0, 1]$.

2. Local beam search with $k = 1$ starts from the root node and then at every step goes to the neighbour with the best heuristic value, until it finds a goal state. Its frontier is always one element.

   In the genetic algorithm the `REPRODUCE` function is a no-op, because the parents are always identical. Hence at every step we either stay in the same state, or we mutate the state with the `MUTATE` function. We therefore return the output of $\texttt{MUTATE}^k(x)$ where x is the only member of the input population.

3. Computing H can be done quickly by starting a breadth-first graph search from the goal node, and updating H whenever we add a node into the frontier.

   Then we start from the starting square (a,b), and at every step we move to (a', b') such that $H(a', b') + 1 = H(a, b)$. This will give us the optimal solution. Alternatively, we can run an A* search from the starting square using H as a heuristic.

4. (a) The only predicate is move.

$$\begin{aligned} & move(x_1, y_1, x_2, y_2) \\ PRECONDITIONS : & peg(x_1, y_1), peg((x_1 + x_2)/2, (y_1 + y_2)/2) \neg peg(x_2, y_2), \\ & (x_1 = x_2 \wedge |y_1 - y_2| = 2 \vee y_1 = y_2 \wedge |x_1 - x_2| = 2), valid(x_2, y_2) \\ EFFECT : & \neg peg(x_1, y_1), \neg peg((x_1 + x_2)/2, (y_1 + y_2)/2), peg(x_2, y_2) \end{aligned}$$

   Initial state: $peg(x, y) \forall x, y : valid(x, y) \wedge \neg(x = y = 4), valid(x, y) \forall x, y :$
   $(x \in \{3, 4, 5\} \vee y \in \{3, 4, 5\}) \wedge 1 \leq x, y \leq 8$
   Goal state: $peg(4, 4), valid(x, y) \forall x, y : (x \in \{3, 4, 5\} \vee y \in \{3, 4, 5\}) \wedge 1 \leq x, y \leq 8$

(b) $\varphi_T^{initial} = \left(\bigwedge_{\substack{1 \le x,y \le 8 \\ x \in \{3,4,5\} \vee y \in \{3,4,5\} \\ \neg(x=y=4)}} peg^0(x,y) \wedge valid^0(x,y) \right) \wedge valid^0(4,4) \wedge \neg peg^0(4,4).$

$\varphi_T^{goal} = \left(\bigwedge_{\substack{1 \le x,y \le 8 \\ x \in \{3,4,5\} \vee y \in \{3,4,5\}}} valid^T(x,y) \right) \wedge peg^T(4,4)$

$\varphi_T^{succ} = \bigwedge_{\substack{1 \le x,y \le 8 \\ x \in \{3,4,5\} \vee y \in \{3,4,5\}}} peg^{T+1}(x,y)_{t+1} \leftrightarrow$

$ActionCausesPeg^T(x,y) \vee (peg^T(x,y) \wedge \neg ActionCausesNotPeg^T(x,y))$

where $ActionCausesPeg^T(x,y) = \bigvee move(a,b,x,y)$ and $ActionCausesNotPeg^T(x,y) = \bigvee move(x,y,a,b) \vee move(x-1,y,x+1,y) \vee move(x+1,y,x-1,y) \vee move(x,y-1,x,y+1) \vee move(x,y+1,x,y-1)$.

$\varphi_T^{prec} = \bigwedge move(x,y,z,t)^T \rightarrow peg(x,y) \wedge peg((x+z)/2,(y+t)/2) \wedge \neg peg(z,t) \wedge (x = z \wedge |y-t| = 2 \vee y = t \wedge |x-z| = 2) \wedge valid(y,t)$

$\varphi_T^{excl} = \bigvee_{(a,b,c,d) \neq (x,y,z,t)} \neg(move(a,b,c,d) \wedge move(x,y,z,t))$

(c) The solver is not intelligent, it applies a mechanical algorithm to some inputs, without the inputs having any meaning for the solver. Intelligence comes into play only when transforming the problem into STRIPS/SAT form.

5. (a) If we have a cycle, then we have 0 total orders. Otherwise, the smallest number of total orders is 1, attained when the partial order is already a total order.

The most orders we can have are $n!$, attained when there are no constraints whatsoever.

(b) `ABCD ABDC ADBC DABC ACBD ACDB ADCB DACB`

With the extra constraint:
`ABCD ABDC ACBD`

6. (a) A configuration space is a set of valid configuration values of the robot.

A is the set of configuration values where the robot is overlapping with Y. C is the set of configuration values where the robot is overlapping with X. B is the set of configuration values where the robot is overlapping with Z and not with Y.

P' is the configuration where the robot is at S, Q' is the configuration where the robot is at G.

(b) A path is desired from P' to Q'. We could decompose the configuration space into around 10 vertical stripes, and use the centers of these stripes as the points of interest for region decomposition. Then a planner would connect P' amd Q' to the points of interest it is able to, and finally run a search on the graph whose nodes are the centers of the regions plus P' and Q', with edges between any two points you can travel between in a straight line.

(c) The base of the robot is at $(4,0)$.

The joint of the robot is at $(a,b) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 5 \end{bmatrix} + \begin{bmatrix} 4 \\ 0 \end{bmatrix}$.

The head of the robot is at $\begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} 4-a \\ -b \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix}$

The C-space can be computed numerically by taking many values of θ and φ and seeing if the head of the robot (according to the formula above) is inside an obstacle or not.