1. This is an open question. Any reasonable answer is OK.

2. No. The intractability results in question are worst-case. In other words, for practical problems of reasonable size, solutions may still be tractable or efficiently approximable. Further, intelligence does not imply optimality under all conditions, so these results do not necessarily apply. An intelligent system needs to be rational under resource constraints, i.e. it might choose to take (suboptimal) actions given the cost of further computation and the cost of not taking an action. For example, if an agent is being chased by a lion, computing the very best direction to run in might be the optimal thing to do, but it might be intractable, resulting in the state of “getting eaten,” which has low utility for the agent. So an intelligent system should be willing to settle for “any direction away from the lion,” even though it is a suboptimal solution.

3. The dynamic execution of a program is very different from the program itself, just as the same genetic sequence can result in different individuals (e.g. twins). Programs can modify themselves as well during execution, just as the brain modifies itself as a result of experience. If we were to follow the question’s reasoning, then since at some level, everything obeys physical laws, we might conclude intelligent behavior cannot exist at all---but clearly that is not so.

4. (i) Ping-pong agent
   Performance measure: winning a game. Could also get intermediate bonuses for hitting the ball, fooling the opponent etc.

   Environment: Ping-Pong ball, Ping-Pong table, Ping-Pong Racket. Static, sequential, continuous. Can be fully observable if opponent is fixed, else not. Can be deterministic if suitable physical abstractions are employed (e.g. elastic collisions), else not.

   Actions: direction to move racket

   Sensors: interface that allow user to move the racket to hit the ball

(ii) Theorem-proving assistant

   Performance measure: minimize the steps it needed to for the proving

   Environment: the set of axioms, statements, rules, hypothesis, conjectures and so on; the computer. Static, deterministic, fully observable, episodic, discrete.

   Actions: apply the proof derivation rules; print results.

   Sensors: interface to read statements.
5. An agent function $f$ maps $|S|$ states to $|A|$ actions, where $|S|$ and $|A|$ are the cardinality of $S$ and $A$. Consider some state $S_1$. Clearly there are $|A|$ possible outcomes when applying some agent function to $S_1$. For a second state $S_2$, there are $|A|$ possible outcomes for each possible outcome of $S_1$, and so forth. Therefore, the total number of distinct functions are $|A| \times |A| \times |A| \ldots (|S| \text{ times}) = |A|^{|S|}$. In general, if a function $f : X \rightarrow Y$ is defined on two discrete domains $X$ and $Y$, the cardinality of the set of functions is $|Y|^{|X|}$.

6. The following is an example where iterative deepening does not find an optimal solution. The initial state is A and the goal state is C. This can be easily extended arbitrarily to be made as bad as we please.

![Diagram](image)

Iterative deepening will end at level 1 and it finds a solution “A→C” with cost 1000; however, the optimal solution is “A→B→C” with cost 2.

7. The proof for this is analogous to the proof for A* with a consistent heuristic: the first time a node is expanded, the optimal path to it has been found, since uniform cost search expands nodes with lowest $g(n)$ (i.e. $h(n) = 0$, which is admissible and consistent). As a result when it finds the goal, it has found the lowest cost solution to it.

8. Assume that for a node $n_0$, an optimal path $n_0 \rightarrow n_1 \rightarrow \ldots \rightarrow n_g$ leads to the goal node $n_g$. (Observe that if there is no path to the goal from some node, the path cost can be treated as infinity, so any $h$ value would be admissible. So only nodes from which there are paths to the goal are interesting.)

We use $c(i, j)$ to denote the cost of the search operator from node $i$ to $j$ in the following proof:

$$h(n_0 ) \leq h(n_1 )+c(n_0 ,n_1 ) \text{ (by consistency)}$$

$$\leq h(n_2 )+c(n_1 ,n_2 )+c(n_0 ,n_1 )$$

$$\leq h(n_3 )+c(n_2 ,n_3 )+c(n_1 ,n_2 )+c(n_0 ,n_1 )$$

$$\ldots$$

$$\leq h(n_g )+c(n_{(g-1)},n_g )+\ldots+c(n_1,n_2 )+c(n_0,n_1 ) \ldots (1)$$

Since $h(n_g )=0$, the sum of $(1)$ is the optimal solution $C^*$; so we have
h(n₀) ≤ C* (admissibility)

9. **Adapt simulated annealing to gradient ascent search**

For gradient ascent we have \( x_{\text{new}} \leftarrow x_{\text{old}} + \alpha \nabla f(x) \). We can adjust simulated annealing as follows for minimizing a function:

Evaluate \( f(x_{\text{new}}) \). Move to \( x_{\text{new}} \) with probability \( \min(\exp[f(x_{\text{old}}) - f(x_{\text{new}})]/T, 1) \), where \( T \) is a temperature parameter varying as in simulated annealing. Thus, if \( f(x_{\text{new}}) \leq f(x_{\text{old}}) \), we will always move to \( x_{\text{new}} \). If \( f(x_{\text{new}}) > f(x_{\text{old}}) \), we may still move to \( x_{\text{new}} \) if \( T \) is relatively large compared to the difference in function values, else we will stay at \( x_{\text{old}} \). We could also add a term for random moves instead of staying at \( x_{\text{old}} \).

10. **Newton-Raphson algorithm**

In order for to make Newton-Raphson method improve at each step, we should guarantee that (for minimization)

\[
\begin{align*}
& f(x_{\text{new}}) < f(x_{\text{old}}), \\
& f(x_{\text{old}}) + u^T \nabla f(x_{\text{old}}) + 1/2 u^T \nabla^2 f(x_{\text{old}}) u < f(x_{\text{old}}), \\
& u^T \nabla f(x_{\text{old}}) + 1/2 u^T \nabla^2 f(x_{\text{old}}) u < 0
\end{align*}
\]

Substituting the Newton step for \( u \), we get the condition: \( \nabla f(x)^T (\nabla^2 f(x))^{-1} \nabla f(x) > 0 \).