COMP417 Artificial Intelligence
Exercise Set 1

Pantourakis Michail AM 2015030185
School of Electrical and Computer Engineering
Technical University of Crete

19 March 2018

Exercise 1

Part A

PEAS descriptions for a robot basketball player, industrial orange-apple sorter and a stock investor are given in Table 1.

Agent Type	Robot Basketball Player	Industrial Orange - Apple Sorter	Stock Investor
Performance Measure	• win % • goals made / goals attempted % • average points per game • performance index rating	• accuracy % • fruits per second	• average ROI • total earnings (or losses) • earnings per month/year
Environment	• basketball court • other agents (teammates and opponents)	• conveyor belt with fruits and surrounding parts of engine • reception of sorted fruits (holes, bins)	• online stock markets • other agents (investors)
Actuators	• motor • mechanical arms and hands for ball control	• mechanical arm and hand sorter	• display forms, stocks, statistical data • display negotiating offers/deals
Sensors	• cameras • gyroscope • distance sensor • wireless agent communication	• cameras • infrared • weight sensor • chemical sensor (electrical nose)	• list of available stocks • stock values and bidding/offers • credit ratings • natural-language processing (for negotiations)

Table 1: PEAS descriptions of three examples of agent types.

Part B

Robot basketball player: *Simple reflex agents* cannot perform well in this partially observable, dynamic environment. An *internal-state model* could be used, but we would need many states that are difficult to define in such a continuous-state environment. In addition, simply storing the current state of the system would not be enough for the player to decide its next course of action. Furthermore, a *goal-based agent* would be inefficient since scoring is not the only goal in basketball. Therefore, *a utility-based model* is deemed necessary to provide a more efficient performance measure in this multiagent environment. Online *learning* could also be applied during the match, allowing adaptation against the opposing team. Its application however would be difficult without sacrificing (at least their initial) performance.

Industrial orange-apple sorter: An *internal-state model* would be enough for fruit classification based on the partially observable fruits and stochastic background (e.g. varying light in the room or existing branches/leaves). *Learning* would be useful if we wanted our agent to classify more types of fruits in the future.

Stock investor: Similar to a robot basketball player, this agent needs to consider not only its basic goal of maximizing its net profit (namely buying cheaply and selling expensively), but also other features such as credibility, financial forecasts and cooperation with other investors. Therefore, a *utility-based agent* would again be more appropriate. *Learning* could also be proven useful in adaptation against opposing agents and new financial, cultural and political conditions.

Exercise 2

Part A

The solution is given in Figure 1.

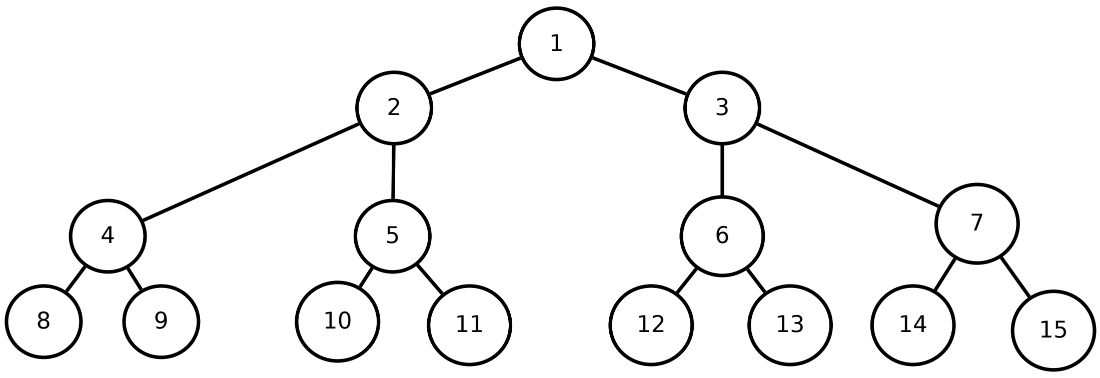

Figure 1: State space for states 1 to 15, starting from 1, while the successor function returning two states, $2n$ and $2n + 1$.

Part B

The order in which the nodes will be visited for each type of search is given below:

- Breadth-First Search: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
- Depth-Limited Search (depth limit 3): 1, 2, 4, 8, 9, 5, 10, 11
- Iterative Deepening Search: (depth limit 0) 1 - (depth limit 1) 1, 2, 3 - (depth limit 2) 1, 2, 4, 5, 3, 6, 7 - (depth limit 3) 1, 2, 4, 8, 9, 5, 10, 11

Part C

Bidirectional search would be quite appropriate for this problem, since we could easily employ the inverse of the successor function for searching in the opposite direction (from goal to start state): $\lfloor n/2 \rfloor$. For example, the bidirectional search algorithm would work as follows:

- Search from start: 1, 2, 3
- Search from goal: 11, 5, (2)

The algorithm would terminate when the search from goal reaches node 2, which would have already been reached by the search from start. The final solution would therefore be 1, 2, 5, 11.

Bidirectional search would actually allow us to avoid searching altogether because each node has only a single predecessor. Hence, the solution could simply be found in $\log_2 n$ steps.

Exercise 3

The solution is given in Figure 2.

Exercise 4

Part A

Since only one state ($k = 1$) is stored in memory after each iteration, it is iteratively succeeded by its best neighboring state like in hill climbing. Hence, the local beam search algorithm with $k = 1$ is a simple hill climbing algorithm.

Part B

With temperature $T = 0$, we can say that the probability $e^{\Delta E/T} = 0$ when $\Delta E \leq 0$, namely only better neighboring solutions are accepted. Therefore, simulated annealing with $T = 0$ is a first choice hill climbing algorithm.

Part C

If $N = 1$, the population will consist of a single individual. Crossover will thus happen between (two copies of) that individual, resulting in the exact same solution. The random mutation mechanism will introduce a small number of point changes during each iteration, consequently turning genetic algorithm into a random walk.

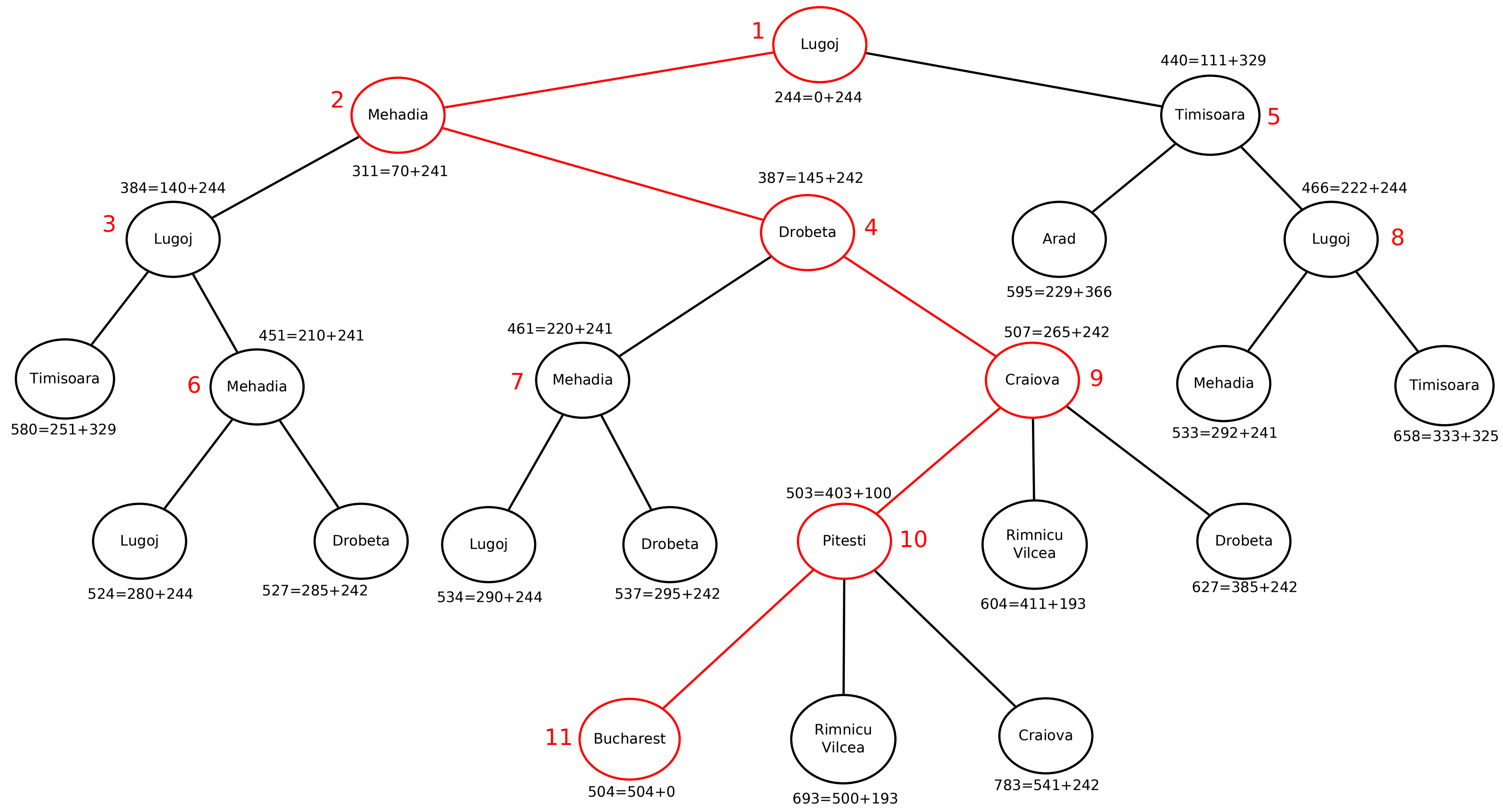

Figure 2: Stages in an A* search from Lugoj to Bucharest. Nodes are labeled with $f = g + h$ (black font), and with the selected order of expansion (red font). Having a sequence of states during each iteration, the algorithm expands the node with the lowest value of f, replacing it in the sequence with all its successors (to be considered for the following iteration). The optimal solution is depicted with red nodes and edges.