CPSC 322: Learning Goals

1 What is AI?

- Identify real world examples that make use of deterministic, goal-driven agents
- Differentiate between single/static and sequential problems.

2 Search

- Assess the size of the search space of a given search problem.
- Implement the generic solution to a search problem.
- Evaluate the complexity of search space in terms of number of nodes, paths, and frontier end nodes.
- Define/read/write/trace/degub different search algorithms (with/without cost)(informed/uninformed)(pruning cycles and repeated states) (including dynamic programming)
- Determine basic properties of search algorithms: completeness, optimality, time and space complexity of search algorithms.
- Select the most appropriate search algorithms for specific problems. BFS vs DFS - A^* vs. B&B vs IDA^* vs MBA^*
- Construct admissible heuristics for appropriate problems. Verify heuristic dominance (the proposition that one heuristic is a weakly tighter bound than another). Combine admissible heuristics.
- Formally prove A^* optimality.
- Define optimal efficiency and formally prove that A^* is optimally efficient.

3 Constraint Satisfaction Problems (CSPs)

- Define possible worlds in term of variables and their domains. Compute number of possible worlds on real examples.
- Specify constraints to represent real-world problems differentiating between unary and k-ary constraints and between list vs. function format. Verify whether a possible world satisfies a set of constraints (i.e., whether it is a model/solution).
- Implement the Generate-and-Test Algorithm. Explain its disadvantages.
- Solve a CSP by search (specify neighbors, states, start state, goal state). Compare strategies for CSP search. Implement pruning for DFS search in a CSP.
- Build a constraint network for a set of constraints. Verify whether a network is arc consistent.
- Define/read/write/trace/degub the arc consistency algorithm. Compute its complexity and assess its possible outcomes.
- Define/read/write/trace/degub domain splitting and its integration with arc consistency.

4 Stochastic Local Search

- Implement local search for a CSP. Implement different ways to generate neighbors.
- Implement scoring functions to solve a CSP by local search through either greedy descent or hill-climbing.
- Implement SLS with random steps and random restart to address limitation of greedy descent / hill climbing.
- Compare different SLS algorithms using runtime distributions.
- Implement a tabu list.
- Implement the simulated annealing algorithm.
- Implement population based SLS algorithms: beam search and genetic algorithms. Discuss pros and cons.

5 Planning

- Represent a planning problem with the STRIPS representation. Explain the STRIPS assumption.
- Solve a planning problem by search (forward planning). Specify states, successor function, goal test and solution.
- Construct and justify a heuristic function for forward planning.
- Translate a planning problem represented in STRIPS into a corresponding CSP problem (and vice versa). Solve the planning problem by extending the horizon until a solvable CSP is found.

6 Logic

- Verify whether an interpretation is a model of a knowledge base expressed in propositional definite clause logic.
- Verify when a conjunction of atoms is a logical consequence of a knowledge base.
- Define/read/write/trace/debug the bottom-up proof procedure.
- Prove that bottom-up procedure is sound and complete.
- Model a relatively simple domain with propositional definite clause logic.
- Define/read/write/trace/debug the top-down proof procedure as a search problem.
- Express knowledge in complex domains using objects and relations. List the advantages of this approach.
- Define the syntax and semantics of Datalog.

7 Reasoning Under Uncertainty

- Define and give examples of random variables, their domains and probability distributions.
- Calculate the probability of a proposition f given $\mu(\omega)$ for the set of possible worlds.
- Define a joint probability distribution.
- Given a joint, compute distributions over any subset of the variables.
- Prove the formula to compute $P(h|e)$.
- Derive the Chain Rule and Bayes' Rule.
- Define and use Marginal Independence.
- Define and use Conditional Independence.
- Build a Belief Network for a simple domain.
- Classify the types of inference.
- Compute the representational saving in terms on number of probabilities required.
- Given a simple belief network, determine whether one variable is conditionally independent of another variable, given a third variable.
- Define factors. Derive new factors from existing factors. Apply operations to factors, including assigning, summing out and multiplying factors.
- Carry out variable elimination by using factor representation and using the factor operations. Use techniques to simplify variable elimination. Prune the belief network based on conditional independence.
- Specify a Markov Chain and compute the probability of a sequence of states.

8 Decision Theory

- Compare and contrast stochastic single-stage (one-off) decisions vs. multistage decisions.
- Define a utility function on possible worlds.
- Define and compute optimal one-off decision (max expected utility).
- Represent one-off decisions as single stage decision networks and compute optimal decisions by variable elimination.
- Represent sequential decision problems as decision networks. Explain the non-forgetting property for sequential decision problems.
- Verify whether a possible world satisfies a policy and define the expected value of a policy.
- Compute the number of policies for a decision problem.
- Compute the optimal policy by variable elimination.
- Effectively represent indefinite/infinite decision processes.
- Compute the probability of a sequence of actions in a Markov Decision Process (MDP).
- Compute number of policies for an MDP and define the computation of the expected total reward of a policy for an MDP.
- Explain influence of rewards on optimal policy.