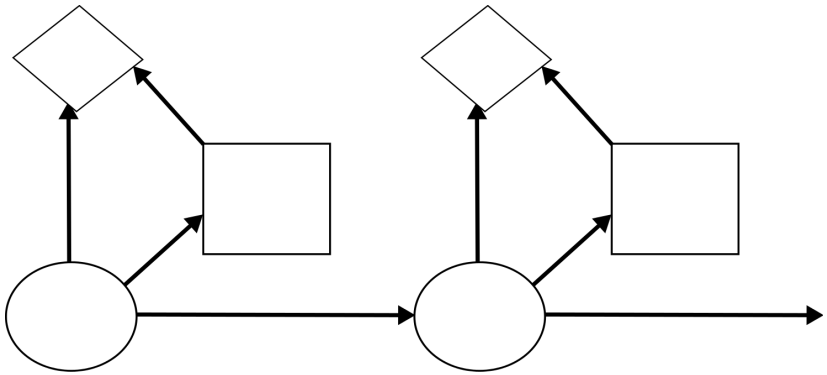
Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Probabilistic Graphical Models: Principles and Applications

Chapter 11: MARKOV DECISION PROCESSES

L. Enrique Sucar, INAOE



Outline

1. Introduction
2. Markov Decision Processes
   - Representation
3. Evaluation
   - Value Iteration
   - Policy Iteration
4. Factored MDPs
   - Abstraction
   - Decomposition
5. POMDPs
6. Applications
   - Power Plant Operation
   - Robot Task Coordination
7. References

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Introduction

- Sequential decision problems – those that involve a series of decisions over time
- Considering that there is uncertainty in the results of the agent's decisions, these type of problems can be modeled as Markov decision processes (MDPs)
- By solving the MDP model we obtain what is known as a *policy*, which indicates to the agent which action to select at each time step based on its current state; the optimal policy is the one that selects the actions so that the expected value is maximized
- Finally we will introduce partially observable MDPs (POMDPs), in which there is not only uncertainty in the results of the actions but also in the state

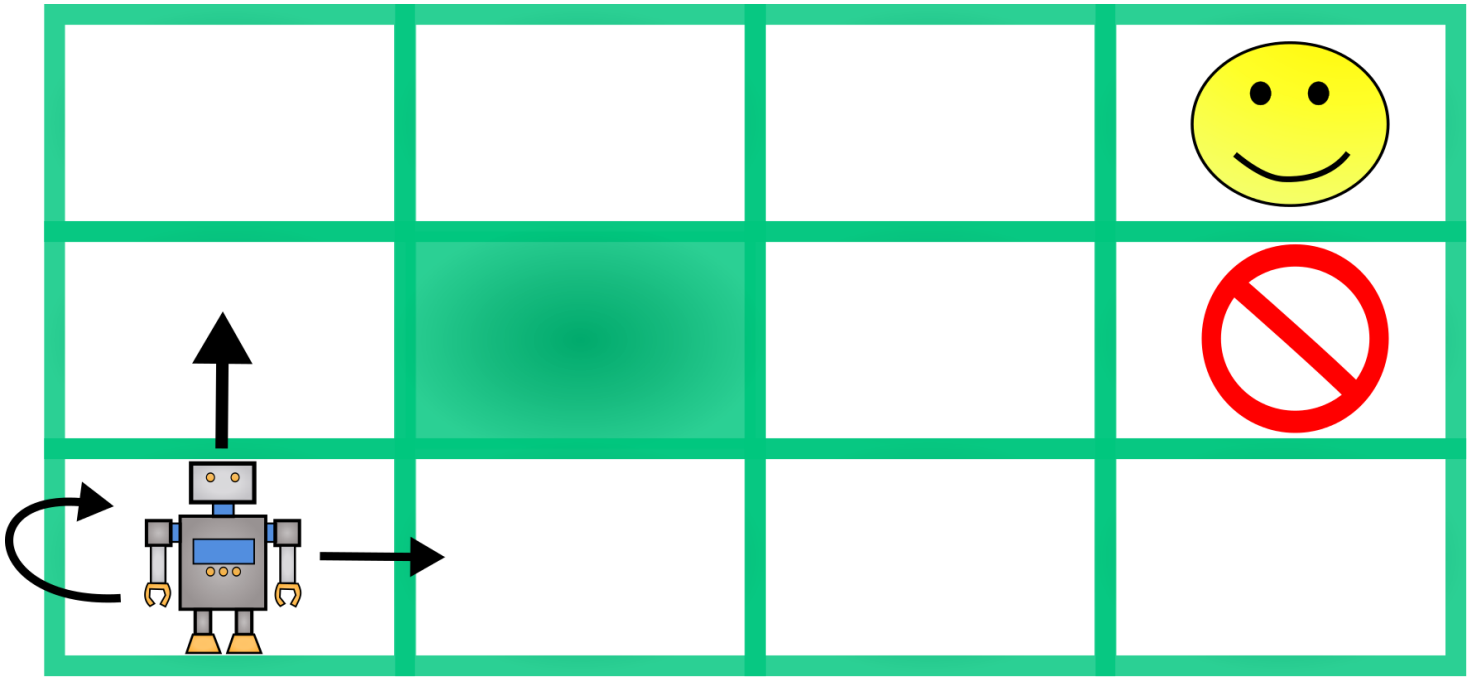
Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Markov Decision Processes

- A Markov decision process (MDP) models a sequential decision problem, in which a system evolves over time and is controlled by an agent
- The system dynamics are governed by a probabilistic transition function Φ that maps states **S** and actions **A** to new states **S'**
- At each time, an agent receives a reward R that depends on the current state s and the applied action a

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Example - robot in the grid world



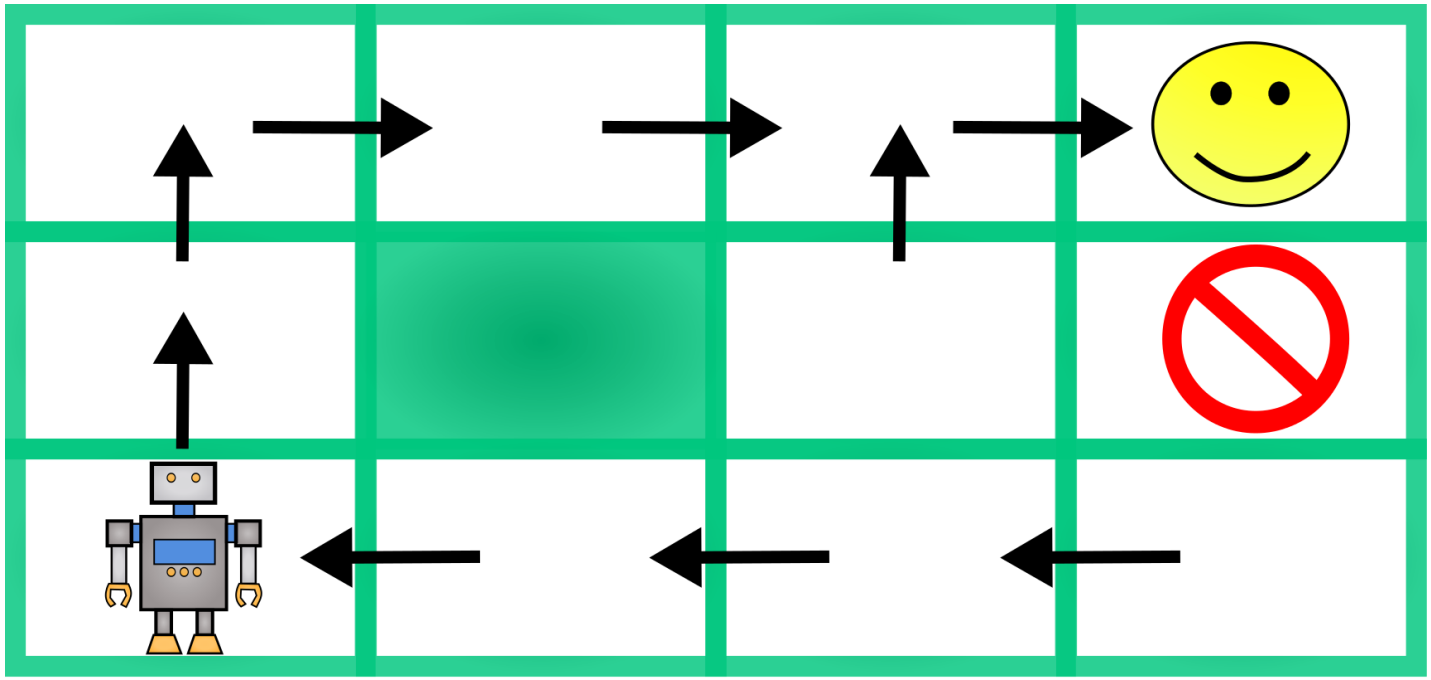
Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Grid World

- The robot's possible actions are to move to the neighboring cells (up, down, left, right)
- There is uncertainty in the result of each action taken by the robot. For example, if the selected action is *up* the robot goes to the upper cell with a probability of 0.8 and with probability of 0.2 to other cells
- The objective of the robot is to go to the goal cell as fast as possible and avoid the dangers. This will be achieved by solving the MDP that represents this problem, and maximizing the expected reward

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Example - policy



Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

MDP - formalization

- An MDP is a tuple $M =< S, A, \Phi, R >$, where S is a finite set of states $\{s_1, \ldots, s_n\}$. A is a finite set of actions $\{a_1, \ldots, s_m\}$. $\Phi : A \times S \times S \rightarrow [0, 1]$ is the state transition function specified as a probability distribution
- The probability of reaching state s' by performing action a in state s is written as $\Phi(a, s, s')$
- $R(s, a)$ is the reward that the agent receives if it takes action a in state s

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Horizon

- Depending on how much into the future (horizon) we consider there are two main types of MDPs: (i) finite horizon and (ii) infinite horizon
- Finite horizon problems consider that there exists a fixed, predetermined number of time steps for which we want to maximize the expected reward
- Infinite horizon problems do not have a fixed, predetermined number of time steps, these could vary and in principle could be infinite – we will focus on this case

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Value

- A *policy*, π, for an MDP is a function $\pi : S \rightarrow A$ that specifies for each state, s_i, the action to be executed, a_i
- Given a certain policy, the expected accumulated reward for a certain state, s, is known as the *value* for that state according to the policy, $V^{\pi}(s)$:

$$V^{\pi}(s) = R(s,a) + \sum_{s \in \mathbf{S}} \Phi(a,s,s') V^{\pi}(s') \qquad (1)$$

- $R(s,a)$ represents the immediate reward given action a, and $\sum_{s \in \mathbf{S}} \Phi(a,s,s') V^{\pi}(s')$ is the expected value of the next states according to the chosen policy

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Discount factor

- For the infinite horizon case, a parameter known as the *discount factor*, $0 \leq \gamma < 1$, is included so that the sum converges
- This parameter can be interpreted as giving more value to the rewards obtained at the present time than those obtained in the future
- Including the discount factor, the value function is written as:

$$V^{\pi}(s) = R(s,a) + \gamma \sum_{s \in \mathbf{S}} \Phi(a,s,s') V^{\pi}(s') \qquad (2)$$

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Bellman Equation

- What is desired is to find the policy that maximizes the expected reward; that is, the policy that gives the highest value for all states
- For the discounted infinite-horizon case with any given discount factor γ, there is a policy π^* that is optimal regardless of the starting state and that satisfies what is known as the *Bellman* equation

$$V^{\pi}(s) = max_a\{R(s,a) + \gamma \sum_{s \in \mathbf{S}} \Phi(a,s,s')V^{\pi}(s')\} \quad (3)$$

- The policy that maximizes the previous equation is then the optimal policy, π^*:

$$\pi^*(s) = argmax_a\{R(s,a) + \gamma \sum_{s \in \mathbf{S}} \Phi(a,s,s')V^{\pi}(s')\} \quad (4)$$

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Evaluation

- There are three basic methods for solving an MDP and finding an optimal policy: (a) value iteration, (b) policy iteration, and (c) linear programming
- The first two techniques solve the problem iteratively, improving an initial value function or policy, respectively
- The third one transforms the problem to a linear program which can then be solved using standard optimization techniques such as the simplex method
- In case the model is unknown an alternative is to *learn* the policy by trail and error, which is known as *Reinforcement Learning* (outside the scope of this book)

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Value Iteration

- Value iteration starts by assigning an initial value to each state; usually this value is the immediate reward for that state
- Then these estimates of the values are improved in each iteration by maximizing the *Bellman* equation
- The process is terminated when the value for all states *converges*
- The actions selected in the last iteration correspond to the optimal policy

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Algorithm

- (Initialization)
- $\forall_s V_0(s) = R(s,a)$
- $t = 1$
- REPEAT (Iterative improvement)
  $\forall_s V_t(s) = max_a\{R(s,a) + \gamma \sum_{s \in \mathbf{S}} \Phi(a,s,s') V_{t-1}(s')\}$
- UNTIL $\forall_s \mid V_t(s) - V_{t-1}(s) \mid < \epsilon$
- (Obtain optimal policy)
- $\pi^*(s) = argmax_a\{R(s,a) + \gamma \sum_{s \in \mathbf{S}} \Phi(a,s,s') V_t(s')\}$

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Analysis

- The time complexity of the algorithm is quadratic in terms of the number of state–actions
- Usually the policy converges before the values converge; this means that there is no change in the policy even if the value has not yet converged. This gives rise to the second approach, policy iteration

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Policy Iteration

- Policy iteration starts by selecting a random, initial policy
- Then the policy is iteratively improved by selecting the action for each state that increases the most the expected value
- The algorithm terminates when the policy converges, that is, the policy does not change from the previous iteration

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Algorithm

- $\pi_0 : \forall_s a_0(s) = a_k$ (Initialize the policy)
- $t = 1$
- REPEAT
- Calculate values for the current policy:
- $\forall_s V_t^{\pi_{t-1}}(s) = \{R(s,a) + \gamma \sum_{s \in \mathbf{s}} \Phi(a,s,s') V_{t-1}(s')\}$
- Iterative improvement:
- $\forall_s \pi_t(s) = argmax_a \{R(s,a) + \gamma \sum_{s \in \mathbf{s}} \Phi(a,s,s') V_t(s')\}$
- UNTIL $\pi_t = \pi_{t-1}$

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Analysis

- Policy iteration tends to converge in fewer iterations than value iteration, however the computational cost of each iteration is higher, as the values have to be updated
- Solving *small* MDPs with the previous algorithms is very efficient; however it becomes difficult when the state–actions space is very large
- An alternative is to decompose the state space and take advantage of the independence relations to reduce the memory and computation requirements, using a graphical model-based representation of MDPs known as *Factored* MDPs

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Factored MDPs

- In a factored MDP, the set of states is described via a set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$
- The transition model and reward function can become exponentially large if they are explicitly represented as matrices, however, the frameworks of dynamic Bayesian networks and decision trees give us the tools to describe the transition model and the reward function concisely

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Transition function - DBN

- The transition function for each action, a, is represented as a two–stage dynamic Bayesian network, that is a two–layer directed acyclic graph G_T whose nodes are $\{X_1, \ldots, X_n, X'_1, \ldots, X'_n\}$
- Each node X'_i is associated with a *conditional probability distribution* $P_\Phi(X'_i \mid Parents(X'_i))$
- The transition probability $\Phi(a, s_i, s'_i)$ is then defined to be $\Pi_i P_\Phi(x'_i \mid \mathbf{u_i})$ where $\mathbf{u_i}$ represents the values of the variables in $Parents(X'_i)$

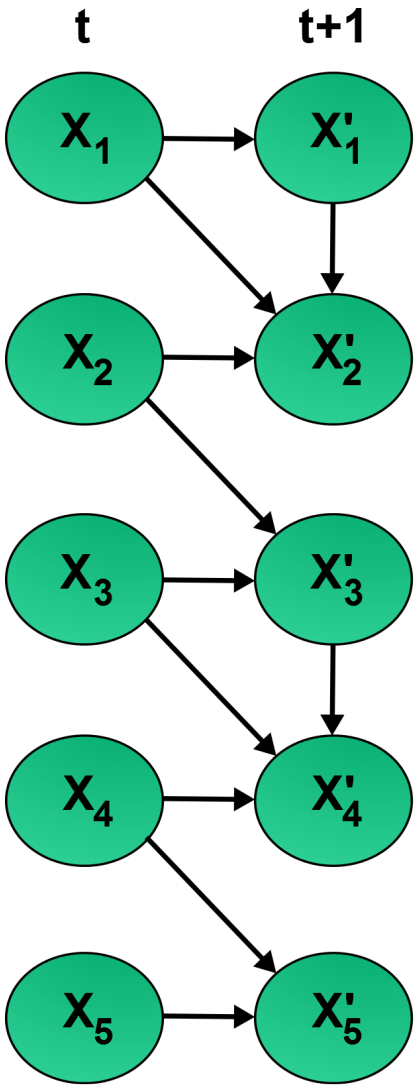
Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

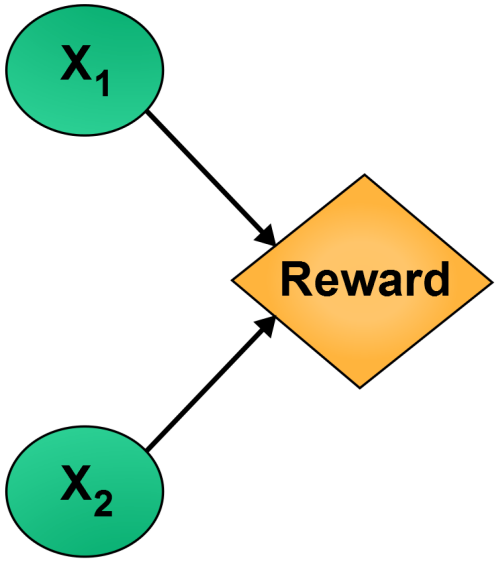
Reward function - DT

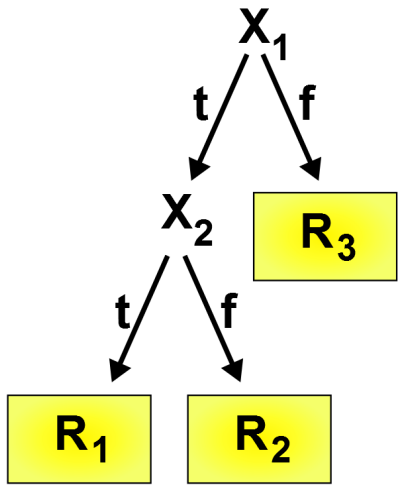
- The reward associated with a state often depends only on the values of certain features of the state
- The relationship between rewards and state variables can be represented with value nodes in an influence diagrams
- Although in the worst case the conditional reward table (CRT) will take exponential space to store the reward function, in many cases the reward function exhibits structure allowing it to be represented compactly using decision trees or graphs

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Example - factored MDP







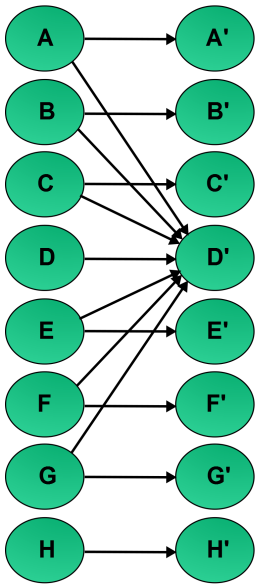
Algebraic Decision Diagrams

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

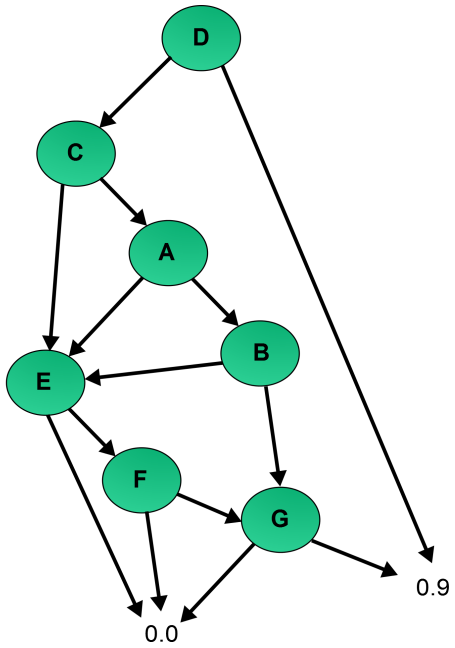
- The representation can be compacted even more by representing a CPT as a tree or a graph, such that repeated probability values appear only once in the leaves of these graphs
- A particular presentation that is very efficient is an *algebraic decision diagram* or ADD
- Based on this compact representation, very efficient versions of the value and policy iteration algorithms have been developed that also reduce the computational time required to solve complex MDP models. An example of this is the SPUDD algorithm

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

ADD



D	C	A	B	E	F	G	D'
T	T/F	T/F	T/F	T/F	T/F	T/F	0.9
F	T	T	T	T/F	T/F	T	0.9
F	T	T	T	T/F	T/F	F	0.0
F	T	T	F	T/F	T/F	T/F	0.0
F	T	F	T/F	T	T	T	0.9
F	T	F	T/F	T	T	F	0.0
F	T	F	T/F	T	F	T/F	0.0
F	T	F	T/F	F	T/F	T/F	0.0
F	F	T/F	T/F	T	T	T	0.9
F	F	T/F	T/F	T	T	F	0.0
F	F	T/F	T/F	T	F	T/F	0.0
F	F	T/F	T/F	F	T/F	T/F	0.0



Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Abstraction

- The idea of abstraction is to reduce the state space by creating an abstract model where states with similar features are grouped together
- *Equivalent states* are those that have the same transition and reward functions; these can be grouped together without altering the original model
- Further reductions can be achieved by grouping *similar* states; this results in approximate models and creates a trade-off between the precision of the model (and the resulting policy) and its complexity
- Different alternatives: partition the state space into a set of blocks such that each block is stable; partition the state space into *qualitative* states that have similar reward functions

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Decomposition

- Decomposition consists in dividing the global problem into smaller subproblems that are solved independently and their solutions combined
- There are two main types of decomposition: (i) serial or hierarchical, and (ii) parallel or concurrent

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Hierarchical MDPs

- Hierarchical MDPs provide a sequential decomposition, in which different subgoals are solved in sequence to reach the final goal
- Hierarchical MDPs accelerate the solution of complex problems by defining different subtasks that correspond to intermediate goals, solving for each subgoal, and then combining these subprocesses to solve the overall problem
- Examples: HAM and MAXQ

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Concurrent MDPs

- In concurrent or parallel MDPs, the subtasks are executed in parallel to solve the global task
- They consider that the task can be divided in several relatively independent subtasks that can be solved independently and then the solutions combined to solve the global problem
- An alternative is to consider conflicts or restrictions between task, initially solving each subtask independently, and when the solutions are combined, take into account potential conflicts between the partial policies, and solve these conflicts to obtain a global, approximately optimal policy

Partially observable MDPs

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

- In some domains, the state can not be observed completely, there is only partial information about the state of the system – POMDP
- In this case, there are certain observations from which the state can be estimated probabilistically
- Consider the previous example of the robot in the grid world. It could be that the robot can not determine precisely the cell where it is (its state), but can estimate the probability of being in each cell by observing the surrounding environment

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Formalization

- A POMDP is a tuple $M =< S, A, \Phi, R, O, \Omega, \Pi >$. The first four elements are the same as in an MDP
- O is a finite set of observations $\{o_1, \ldots, o_l\}$
- $\Omega : S \times O \rightarrow [0, 1]$ is the observation function specified as a probability distribution, which gives the probability of an observation o given that the process is in state s, $P(o \mid s)$
- Π is the initial state distribution that specifies the probability of being in state s at $t = 0$

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Solving a POMDP

- In a POMDP the current state is not known with certainty, only the probability distribution of the state, which is known as the *belief state*
- So solving a POMDP requires finding a mapping form the belief space to the action space - equivalent to a continuous state space MDP
- Solving a POMDP is much more difficult than solving an MDP, as the belief space is in principle infinite

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Approximate Solution Techniques

Policy tree and DDN techniques: assuming a finite horizon, a POMDP can be represented as a *policy tree* (similar to a decision tree) or as a dynamic decision network; then, algorithms for solving these types of models can be applied.

Sampling techniques: the value function is computed for a set of points in the belief space, and interpolation is used to determine the optimal action to take for other belief states which are not in the set of sampling points.

Value function approximation techniques: given that the value function is convex and piece-wise linear, it can be described as a set of vectors (called α vectors). Thus, it can be approximated by a set of vectors that *dominate* the others, and in this way find an approximate solution.

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Applications

- Assisting power plant operators in the operation of a power plant under difficult situations
- Coordinating a set of modules to solve a complex task for service robots

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Power Plant Operation Assistant

- The steam generation system of a combined-cycle power plant provides superheated steam to a steam turbine
- During normal operation, a three-element feedwater control system commands the feed-water control valve (*fwv*) to regulate the level (*dl*) and pressure (*pd*) in the drum
- However, this traditional controller does not consider the possibility of failures in the control loop and ignores whether the outcomes of executing a decision will help to increase the steam drum lifetime, security, and productivity
- This problem was modeled as an MDP – for training and assistance for power plant operators

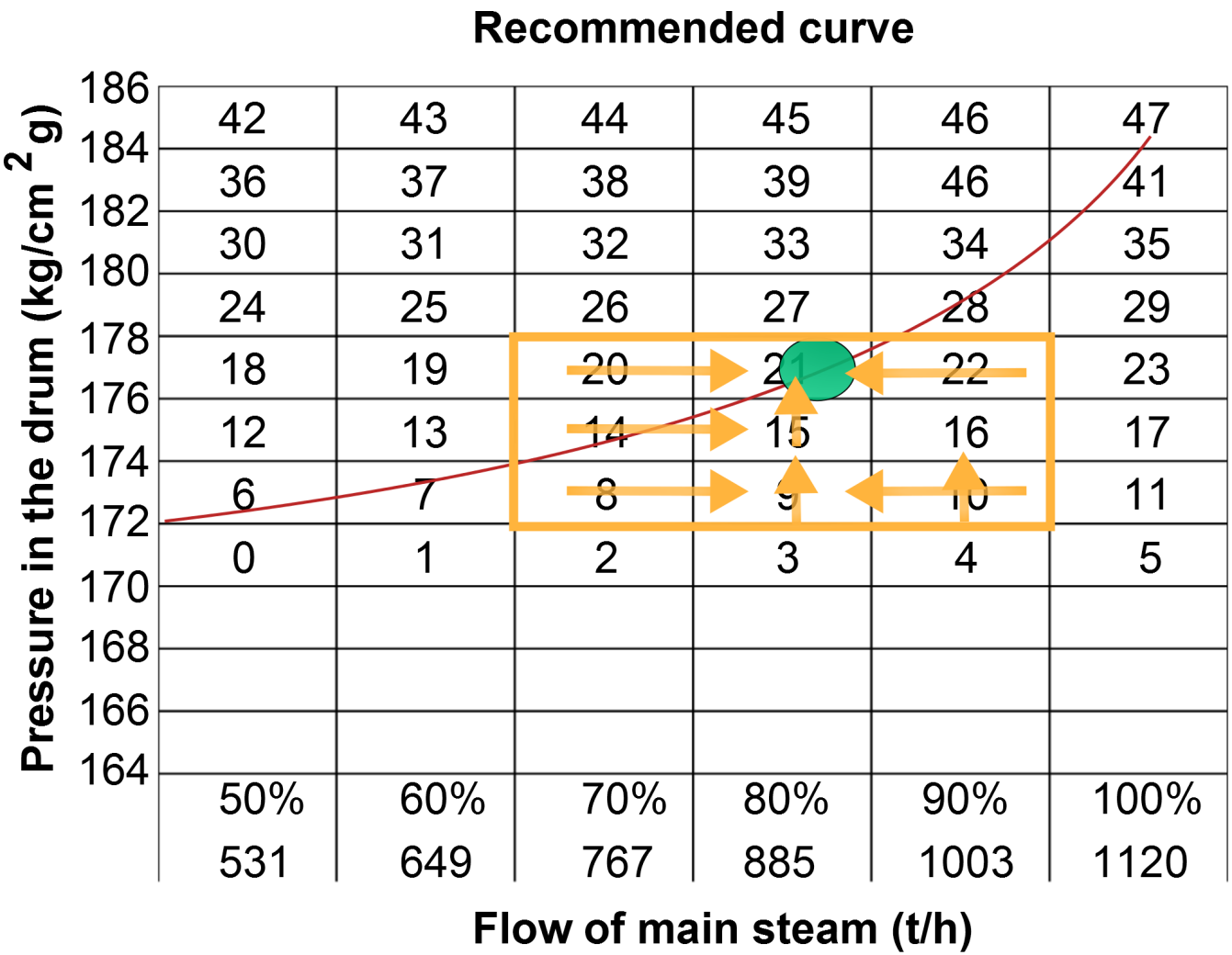
Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

AsistO

- AsistO is based on a decision-theoretic model -MDP- that represents the main elements of the steam generation system of a combined-cycle power plant
- The main variables in the steam generation system represent the state in a factored form
- The actions correspond to the control of the main valves in this subsystem of the power plant: feed-water valve (*fwv*) and main steam valve (*msv*)
- The reward function is defined in terms of a recommended operation curve for the relation between the drum pressure and steam flow, the control actions should try to maintain the plant within this recommended operation curve
- The transition function is learned by using the power plant simulator and sampling the state and action spaces

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Recommended Operation Curve



Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Experiments

- Model: five state variables: *Fms*, *Ffw*, *Pd*, *g*, *d*; and four actions: open/close the feed-water (*fwv*) and main steam (*msv*) valves.
- The reward function was defined based on the recommended operation curve
- The memory requirements for a *flat* MDP representation and a factored representation were compared. The flat MDP required $589,824$ parameters (probability values) while the factored MDP only 758
- The optimal solution for the factored MDP was obtained in less than two minutes on a standard personal computer

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Robot Task Coordination

- A service robot should combine several capabilities, such as localization and navigation, obstacle avoidance, people detection and recognition, object recognition and manipulation, etc.
- These different capabilities can be implemented as independent software modules, which can then be *combined* for solving a particular task
- It is necessary to coordinate the different modules to perform a task, ideally in an optimal way

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Task coordination with MDPs

- Markov decision processes provide an appropriate framework for task coordination for service robots
- Once a task is modeled as an MDP, the MDP can be solved to obtain a policy to perform the task, which is robust with respect to the uncertainty in the results of the different actions
- Under this framework, based on general software modules and an MDP–based coordinator, it is in principle relatively easy for a service robot to solve different tasks. We just need to modify the MDP reward function according to the new task objectives
- Additionally, it is desirable for the robot to perform several actions *simultaneously*, such as navigation to a certain location, avoiding obstacles and looking for people; all at the same time – this implies an explosion in the action—state space

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Concurrent MDPs

- Based on functional decomposition, a complex task is partitioned into several subtasks
- Each subtask is represented as an MDP and solved independently, and the policies are executed in parallel
- However, conflicts may arise between the subtasks: (i) *resource conflicts*, and (ii) *behavior conflicts*.

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Resource onflicts

- Resource conflicts occur when two actions require the same physical resource (e.g., to control the wheels of a robot) and cannot be executed concurrently
- This type of conflict is solved off–line by a two-phase process
- In the first phase we obtained an optimal policy for each subtask (MDP) – if there is a conflict between the actions selected by each MDP for a certain state, the one with maximum value is considered, and the state is marked as a *conflict* state
- This initial solution is improved in a second phase using policy iteration

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Behaivor

- Behavior conflicts arise in situations in which it is possible to execute two (or more) actions at the same time but it is not desirable given the application
- Behavior conflicts are solved on–line based on a set of restrictions specified by the user
- If there are no restrictions, all the actions are executed concurrently; otherwise, a constraint satisfaction module selects the set of actions with the highest expected utility

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Experiments

- An experiment was done with *Markovito*, a service robot, which performed a delivery task

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Delivery Task

- The goal is for the robot to receive and deliver a message, an object or both, under a user's request
- Subtasks:
  1. *navigation*, the robot navigates safely in different scenarios;
  2. *vision*, for looking and recognizing people and objects;
  3. *interaction*, for listening and talking with a user;
  4. *manipulation*, to receive and deliver an object safely; and
  5. *expression*, to show *emotions* using an animated face.
- Each subtask is represented as a factored MDP

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Representation

- If we represent this task as a single, flat MDP, there are 1,179,648 states (considering all the non-duplicated state variables) and 1,536 action combinations, giving a total of nearly two billion state-actions
- The MDP model for each subtask was defined using a structured representation
- The transition and reward functions were specified by the user based on task knowledge and intuition
- In this task, conflicts might arise between the different subtasks, so we need to include conflict resolution strategies

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Behavior Conflicts

action(s)	restriction	action(s)
get message	**not_during**	turn OR advance
ask_user_name	**not_before**	recognize_user
recognize_user	**not_start**	avoid_obstacle
get_object OR deliver_object	**not_during**	directed towards OR turn OR moving

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Evaluation

- For comparison, the delivery task was solved under two conditions: (i) without restrictions and (ii) with restrictions
- In the case without restrictions, the robot performs some undesirable behaviors. For example, in a typical experiment, the robot is not able to identify the person who wants to send a message for a long time
- In the case where restrictions were used, these allowed a more fluid and efficient solution
- On average, the version with restrictions takes about 50% of the time steps required by the version without restrictions to complete the task

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Book

Sucar, L. E, *Probabilistic Graphical Models*, Springer 2015 – Chapter 11

Additional Reading - Fundamentals

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References


Bellman, R.: Dynamic Programming. Princeton University Press, Princeton, New Jersey (1957)


Poupart, P.: An Introduction to Fully and Partially Observable Markov Decision Processes. In: L. E. Sucar, J. Hoey, E. Morales (eds.) Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions, IGI Global, Hershey (2011)


Puterman, M. L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (1994)

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Additional Reading - Abstraction and Decomposition I

 Dean, T., Givan, R.: Model Minimization in Markov Decision Processes. In: Proceedings of the 14th National Conference on Artificial Intelligence (AAAI), pp. 106–111 (1997)

 Dietterich, T.: Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. Journal of Artificial Intelligence Research, 13, 227–303 (2000)

 Hoey, J., St-Aubin, R., Hu, A., Boutilier, C.: SPUDD: Stochastic Planning Using Decision Diagrams. In: Proceedings UAI, pp. 279–288 (1999)

 Parr, R., Russell, S. J.: Reinforcement Learning with Hierarchies of Machines. In: Proceeding NIPS, (1997)

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Additional Reading - Abstraction and Decomposition II

 Li, L., Walsh, T. J., Littman, M. L.: Towards a Unified Theory of State Abstraction for MDPs. In: Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics, pp. 21–30 (2006)

 Meuleau, N., Hauskrecht, M., Kim, K.E., Peshkin, L., Kaelbling, L.P., Dean, T., Boutilier, C.: Solving Very Large Weakly Coupled Markov Decision Processes. In: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI), pp. 165–172 (1998)

Introduction
Markov Decision Processes
Representation
Evaluation
Value Iteration
Policy Iteration
Factored MDPs
Abstraction
Decomposition
POMDPs
Applications
Power Plant Operation
Robot Task Coordination
References

Additional Reading - Applications

 Corona, E., Morales E.F, Sucar, L.E.: Solving Policy Conflicts in Concurrent Markov Decision Processes. ICAPS Workshop on Planning and Scheduling Under Uncertainty. (2010)

 Corona, E., Sucar, L.E.: Task Coordination for Service Robots Based on Multiple Markov Decision Processes. In: L. E. Sucar, J. Hoey, E. Morales (eds.) Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions, IGI Global, Hershey (2011)

 Reyes, A., Sucar, L.E., Morales, E.F.: AsistO: A Qualitative MDP-Based Recommender System for Power Plant Operation. Computacion y Sistemas, 13 (1), 5–20 (2009)