

Representations and Algorithms for Large Scale Markov Decision Processes

Adrian Vladu

Dilyana Dimitrova

Teodor Moldovan



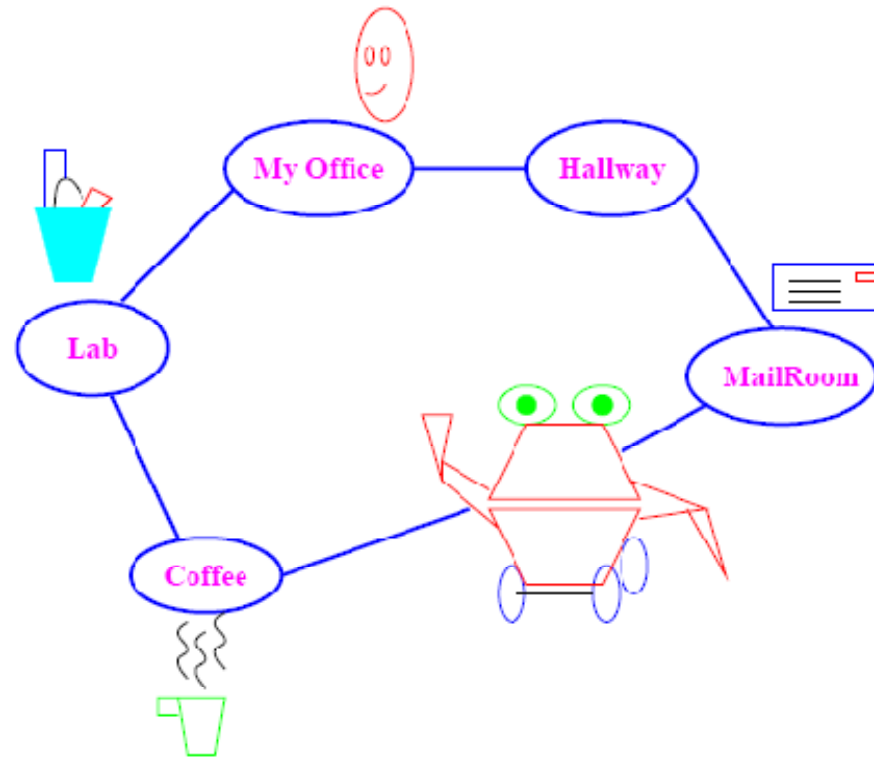
Introduction

- What are MDPs?
- Why are MDPs useful in AI planning?
 - AI planning problems can be modeled as MDPs.
- What is good about MDPs?
 - They provide a well-defined and structured way of representing AI problems.
- What is wrong with MDPs as a planning model?
 - Solution techniques for MDPs don't work with planning problems of reasonable size because of the “curse of dimensionality”.

MDP: Problem Formulation

- State space S
- Set of actions A – each action a_k is represented by a probability transition matrix P_k .
- Value function – defined by a reward function $R: S \rightarrow \mathbf{R}$ and a cost function $C: S \times A \rightarrow \mathbf{R}$
- Horizon T – the number of stages over which the value function should be evaluated.
- Optimality criterion – a criterion for evaluating potential solutions to planning problems.

Robot Example



Elements of the Robot Domain

Features	Denoted	Description
Location	$Loc(M)$, etc.	Location of robot. Five possible locations: mailroom (M), coffee room (C), user's office (O), hallway (H), laboratory (L)
Tidiness	$T(0)$, etc.	Degree of lab tidiness. Five possible values: from 0 (messiest) to 4 (tidiest)
Mail present	M, \overline{M}	Is there mail in user's mail box? True (M) or False (\overline{M})
Robot has mail	RHM, \overline{RHM}	Does the robot have mail in its possession?
Coffee request	CR, \overline{CR}	Is there an outstanding (unfulfilled) request for coffee by the user?
Robot has coffee	RHC, \overline{RHC}	Does the robot have coffee in its possession?
Actions	Denoted	Description
Move clockwise	Clk	Move to adjacent location (clockwise direction)
Counterclockwise	$C\overline{Clk}$	Move to adjacent location (counterclockwise direction)
Tidy lab	$Tidy$	If the robot is in the lab, the degree of tidiness is increased by 1
Pickup mail	PUM	If the robot is in the mailroom and there is mail present, the robot takes the mail (RHM becomes true and M becomes false)
Get coffee	$GetC$	If the robot is in the coffee room, it gets coffee (RHC becomes true)
Deliver mail	$DelM$	If the robot is in the office and has mail, it hands the mail to the user (RHM becomes false)
Deliver coffee	$DelC$	If the robot is in the office and has coffee, it hands the coffee to the user (RHC and CR both become false)
Events	Denoted	Description
Mail arrival	$ArrM$	Mail arrives causing M to become true
Request coffee	$ReqC$	User issues coffee request causing CR to become true
Untidy the lab	$Mess$	The lab becomes messier (one degree less tidy)

Solving MDPs: Value Iteration

- Computes the value function of each state for all actions.

$$V_t^*(s) = R(s) + \max_{a \in \mathcal{A}} \left\{ C(a) + \sum_{s' \in \mathcal{S}} \Pr(s' | a, s) V_{t-1}^*(s') \right\}$$

- Sets each state's value to the greatest value achieved among all courses of action.
- The actions that yield the optimal state values can be extracted as the optimal policy.

Solving MDPs: Policy Iteration

- Policy evaluation – computes the state values for a current fixed policy.
- Policy improvement – improves upon the current policy in a greedy fashion. Finds the action a^* that maximizes:

$$Q_{i+1}(a, s) = R(s) + C(a) + \gamma \sum_{s' \in S} \Pr(s'|a, s) \cdot V^{\pi_i}(s')$$

- Slower than value iteration per iteration, however takes far fewer iterations to converge.

Problem: “Curse of Dimensionality”

- Both algorithms require explicit enumeration of the state space at each iteration. They rely on an *extentional* representation for the set of states.
- Problem: The state space grows exponentially in the number of features required to describe a system:
 - In the robot example: six features to describe a state.
 - The state space comprises all possible combinations of feature values, with $|S| = 400$.
 - State transition matrix with 160000 parameters.

Intensional vs. Extensional Representation of States

- Extensional representation:
 - States are explicitly enumerated.
- Intensional representation:
 - Structured states - defined using a finite set of state variables whose values change over time.
 - Describes sets of states based on certain features rather than enumerating each state explicitly.
 - Conclusion: It is more natural and compact.

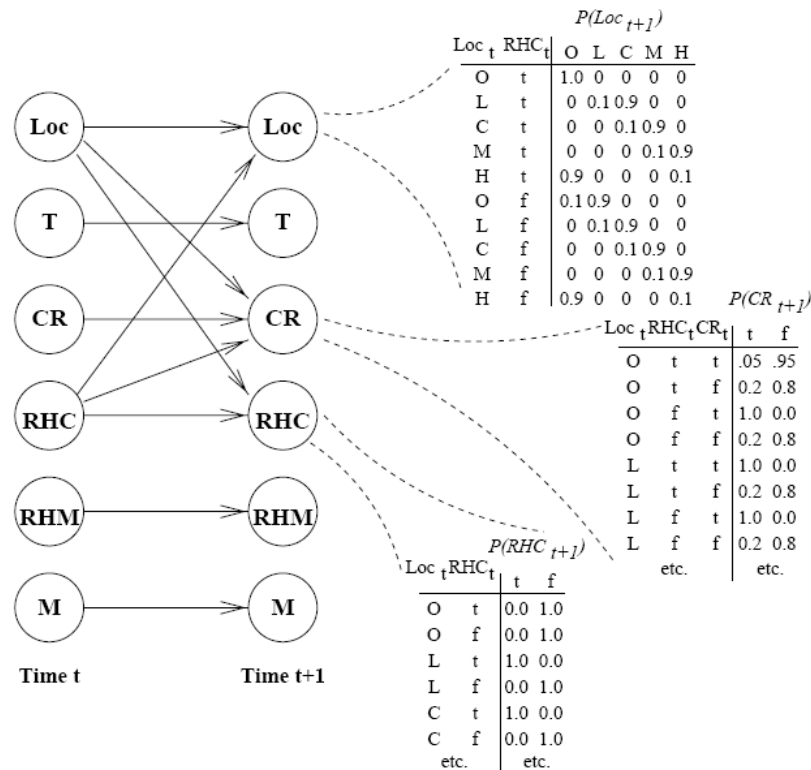
Factored Representations for Markov Random Chains

- Factored state space – specified using more than one state variable (otherwise it is called flat).
- Intensional representation of states.
- State space is the cross product of the value spaces for the individual state variables.
- Compact representation of state transition matrices.
- Used to represent actions, rewards, policy and value functions.
- Regularities can be exploited to speed up algorithms.

Bayesian network representation: 2TBN

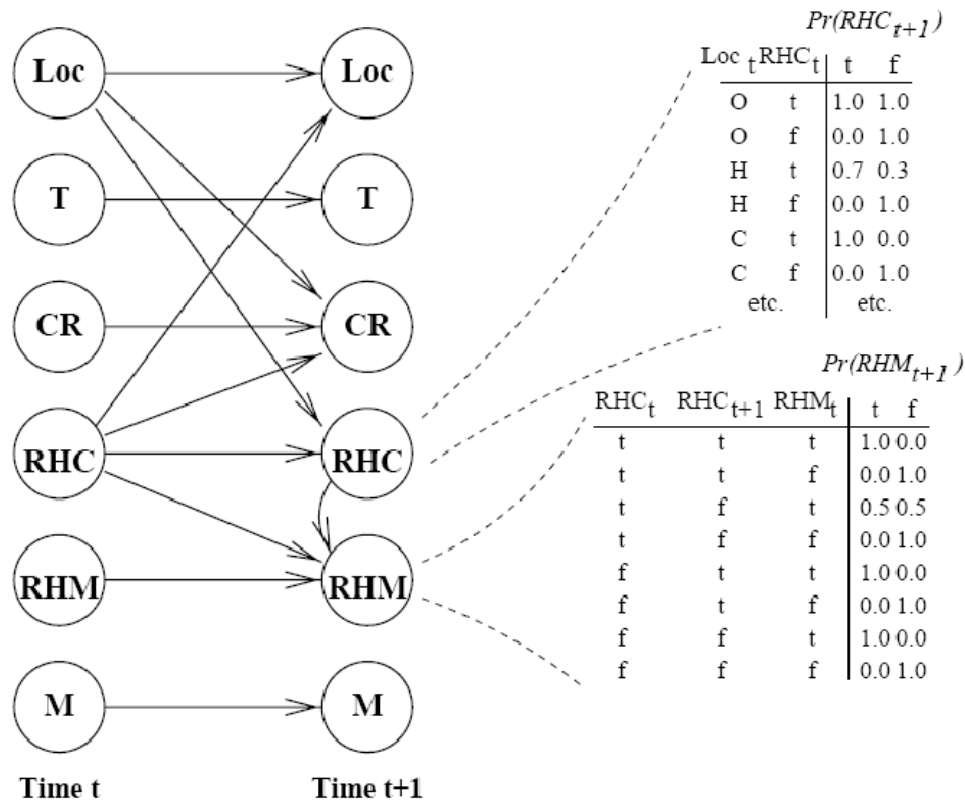
- Graph representing dependencies between state variables at consecutive time steps.
 - Diachronic arcs
 - Synchronic arcs
- Take advantage of independences among variables.
- Transition probabilities for every variable represented in conditional probability tables (CPTs).
- In our robot example:
 - Full transition matrix: 160000 parameters
 - 2TBN representation: 66 parameters

2TBN Graph and CPT Example (1)



A 2TBN for the Markov chain induced by moving counterclockwise and delivering coffee.

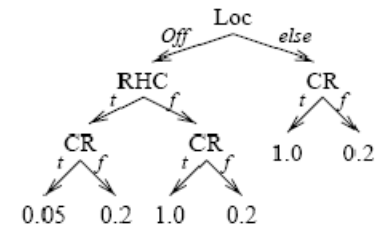
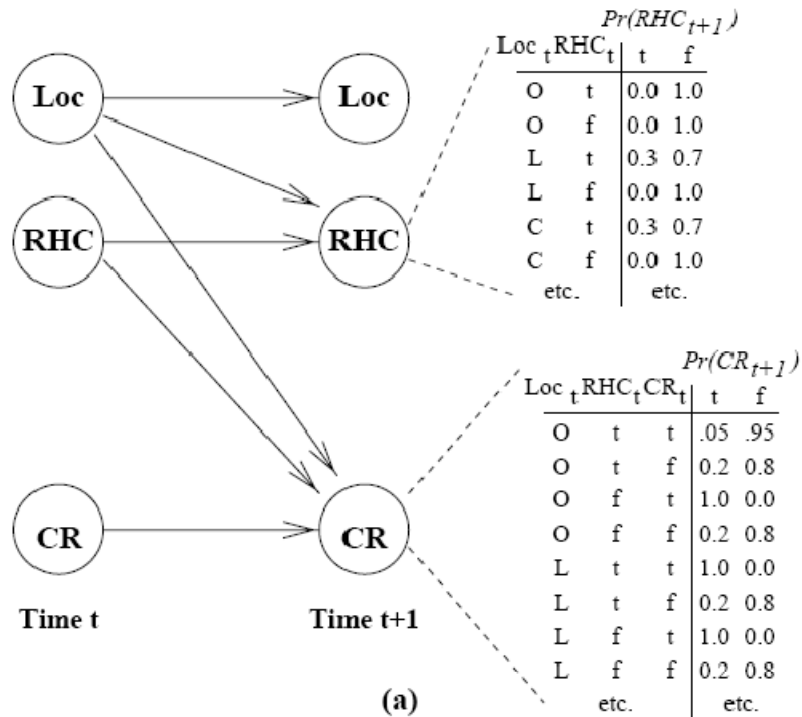
2TBN Graph and CPT Example (2)



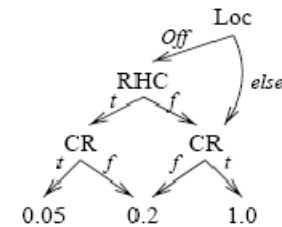
A 2TBN for the Markov chain induced by moving counterclockwise and delivering coffee with correlated effects.

Structured CPTs

- CPTs can be represented more compactly by using Decision Trees.



(b)

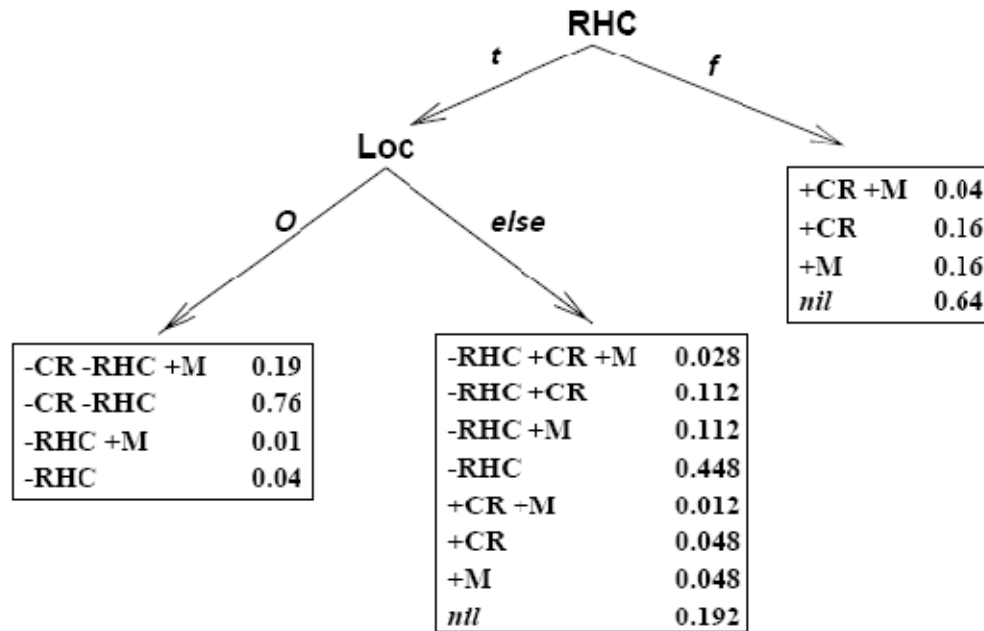


(c)

Factored Action Representation: Probabilistic STRIPS Operator (PSO)

- Outcome-oriented representation – describes the possible outcomes of an action or the possible joint effects over all variables.
- Consists of a *precondition* (a set of states in which the action can be executed) and a set of *effects*.
- Difference from 2TBN – all variables that are unaffected by an action must be given CPTs in the 2TBN model, while such variables are not mentioned in the PSO model (solves the so called *frame problem* of 2TBN).

PSO Representation Example



A PSO representation for the *DelC* action.

Techniques Based on Factored Representations

- Problem: Standard MDP algorithms such as value iteration and policy iteration no longer work on factored representations.
- Need new algorithms to take advantage of regularity exposed by factored representation.

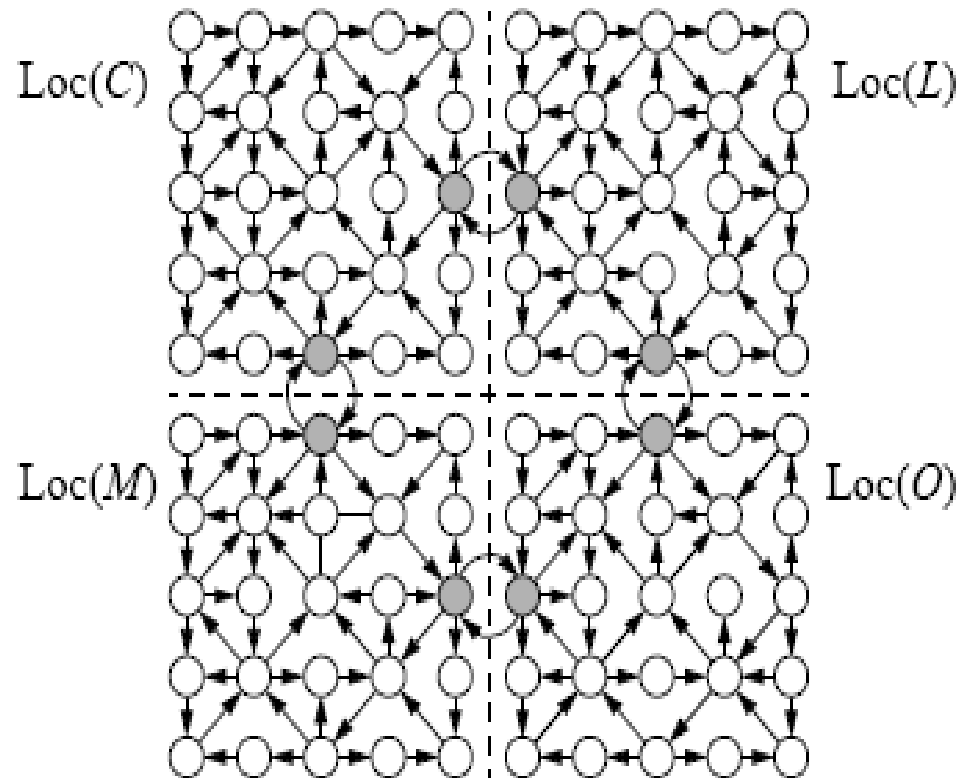
Techniques Based on Factored Representations

- Aggregation by Abstraction techniques:
 - Goal Regression
 - Stochastic Dynamical Programming with Structured Representations
 - Model Minimization and Reduction Methods
- Problem decomposition techniques:
 - Serial Problem decomposition
 - Parallel Problem decomposition

Problem Decomposition Techniques

- MDP split into pieces. Each solved independently.
- Cluster based on communicating structure.
- Treat sub-processes that interact minimally as independent. Solve them independently. Piece together solutions.
- Serial decomposition
 - Splits the state space into clusters between which there is little communication and solve them independently.
- Parallel decomposition
 - Finds a set of sub-MDPs that are run in parallel; the state space of the initial MDP is the product of the state spaces of the sub-MDPs.

Serial Decomposition

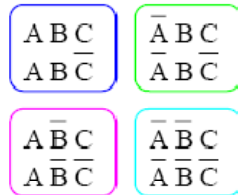


Aggregation and Abstraction

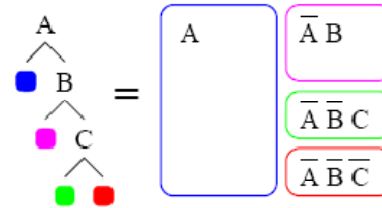
- **Aggregation** – grouping states that are indistinguishable with respect to certain characteristics (such as expected reward)
- Each abstract state can be treated as a single state.
- **Abstraction** – particular form of aggregation. States are aggregated by ignoring certain problem features.

Different Forms of State Space Abstraction

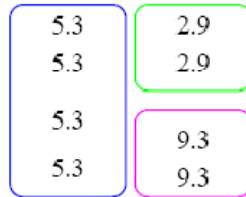
Uniform



Nonuniform



Exact



Approximate



Adaptive

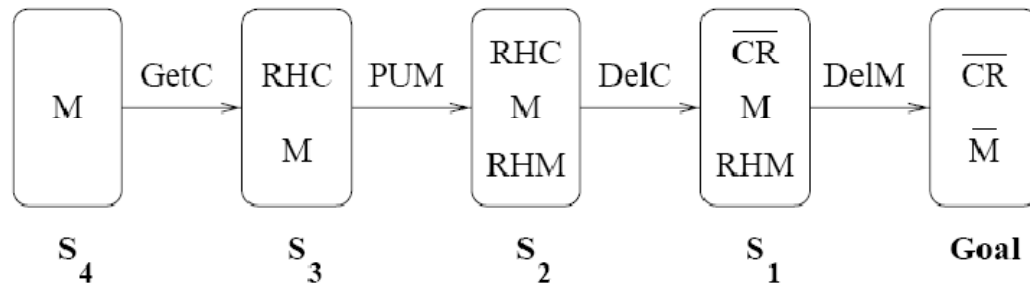


Fixed



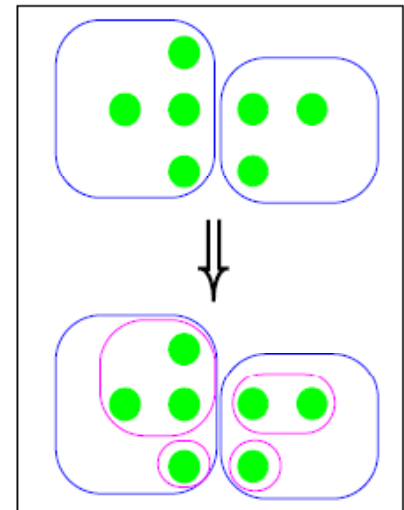
Goal Regression: Robot Example

- Set of four actions: PUM, GetC, DelC and DelM
- Initial state: $\langle CR, M, \overline{RIIC}, \overline{RIIM} \rangle$
- Goal set G: $\{\overline{CR}, \overline{M}\}$
- At each iteration an action is selected that achieves the current subgoals.
- The process is repeated until one of two conditions holds:
 - the current subgoal set is satisfied by the initial state, or
 - no action can be applied.



Stochastic Dynamic Programming with Structured Representations

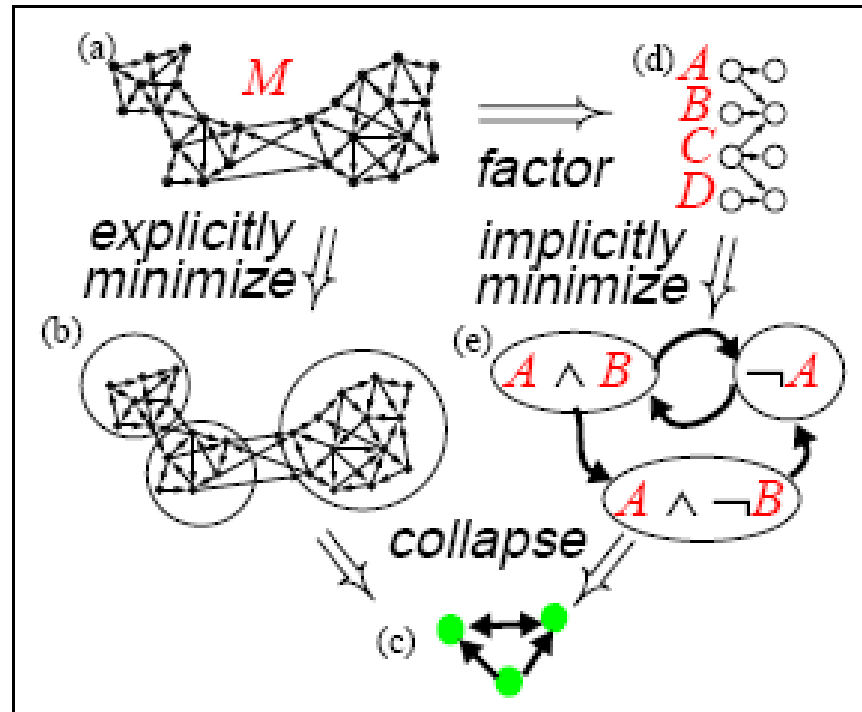
- Idea: Grouping states according to their value with respect to a fixed policy.
- Initially group states according to their immediate rewards.
- Using regression construct new partition in which states are grouped according to the one-stage-to-go value function.
- On the k th iteration – states are grouped according to the k -stage-to-go value function.



Model Minimization

- Idea:
 - Have a large problem in intensive form.
 - Want a small problem in extensive form.
 - Solve small problem with our previous techniques.
- Inspired by automata minimization.
- Want to cluster states based on expected gain. Have to look at value function and transition probabilities.
- Problem: a small minimal model may exist but be hard to find.

Model Minimization



Utility of Exact Aggregation Methods

- Proven experimentally to be useful for a limited set of abstract process-planning problems.
- Not tested on any real problems. Probably not effective.
- Approximation based methods might be more appropriate.
- It is often difficult to separate the essential assumptions from the accidental

Model Minimization and Reduction Methods

- We present a reduction method based on automata minimization.
- We're looking forward to obtain a *quotient model* of the MDP.
- Need to partition states somehow such that any element in the partition can be aggregated into a single state of the new MDP.

Minimization via Partitioning

- A partition of the states $P = \{B_1, B_2, \dots, B_n\}$ is homogenous if for any two B_i, B_j the following holds:

$$\forall \alpha \in A \forall p, q \in B_i \sum_{r \in B_j} f_{pr}(\alpha) = \sum_{r \in B_j} f_{qr}(\alpha)$$

- In the quotient model the transition probabilities are defined by $f'_{ij}(\alpha) = \sum_{r \in B_j} f_{pr}(\alpha)$
- This doesn't distinguish between states that differ on the basis of reward. Hence we start with an initial partition P_0 , then we split each block in P_0 into a homogenous partition.

Partitioning the States

- Call a partition P' a refinement of P iff every block in P' is a subset of some block in P . P is called to be coarser than P' .
- Nice approach for partitioning: find the coarsest homogenous refinement of P_0

Stability

- C is stable in respect to B and α iff every state in C has the same probability of being carried into block B by action α .
- A block is stable if it's stable in respect to every block in P and action in A, therefore homogeneity \Leftrightarrow all the blocks are stable

$$\exists c \in [0, 1], \forall p \in C, \Pr(X_{t+1} \in B | X_t = p, U_t = \alpha) = c$$

where

$$\Pr(X_{t+1} \in B | X_t = p, U_t = \alpha) = \sum_{q \in B} \Pr(X_{t+1} = q | X_t = p, U_t = \alpha)$$

Partitioning the States

- Theorem: given a partition P , blocks B and C , and states p and q in C such that

$$Pr(X_{t+1} \in B | x_t = p) \neq Pr(X_{t+1} \in B | X_t = q)$$

then p and q do not fall in the same block of the coarsest homogenous refinement of P .

- Algorithm: find unstable block C in respect to some B and α . Split it into maximal sub-blocks s.t. each of them is stable in respect to B and α (call it $SPLIT(B, C, P, \alpha)$).

Partitioning the States

- Theorem: the above algorithm produces the coarsest homogenous refinement of an initial partition.
- To get the reduced MDP, we define the reward function $R'(\alpha, i) = R(\alpha, p)$, for any p in B_i
- Theorem: the exact solution of the reduced MDP induces an exact solution of the original MDP.

Factored Representations

- Use Bayesian networks
- A state is a product of features

$$X_t = \langle X_{1,t}, \dots, X_{m,t} \rangle$$

- State transition probabilities are factored

$$\Pr(X_{t+1}|X_t, U_t) = \prod_{i=1}^m \Pr(X_{i,t+1}|\text{Parents}(X_{i,t+1}), U_t)$$

- Each of the conditional probability distributions can be represented as a CPT or decision tree

Partitioning Factored Representations

- Partitions are represented in DNF
- Finding the coarsest homogenous refinement of a partition represented in DNF is NP-hard 😞
- We consider factor-wise split operations a.k.a. the resulting partition $\text{FSPLIT}(B, C, P, \alpha)$ is factor-wise representable.
- The coarsest homogenous factor-wise representable refinement is computable in polynomial time.

Ideas for Future Research

- Find an approximation scheme for this algorithm.
- We don't know how well this minimization technique works on real problems.
Experiment!
- MDP minimization not based on state partitioning. Exact minimization. Approximate minimization. Any ideas?