Hilfszettel für Intro to AI
von Philip G., Seite 1 von 2

1 symbolic

uninformed search

properties
- time and space complexity: worst case
- semi-complete: guaranteed to find a solution
- complete: terminates if no solution exists

bfs
- Fifo, shallowest layer first, linked list, dequeue
- complete if finite, else semi
- optimal if all actions same cost
- $time(b^d)$, space is same

dfs
- Lifo, deepest node first, stack, not optimal
- complete if acyclic, finite else not semi/complete
- $time(b^{maxsearchd}), space(b \cdot maxsearchd)$

ida
- comlete if acyclic, finite else semi
- not optimal, $time(b^d)$, $space(bd)$

informed search

heuristic
$h: S \to \mathbb{R}_0^+ \cup \{\infty\}$
can be arbitrary function

perfect heuristic
h^*: perfect heuristic
- maps to cost of optimal solution
- maps to ∞ if no solution exists for s

heuristic properties
safe: $h^*(s) = \infty$ f.a $s \in S$ with $h(s) = \infty$
goal-aware: $h(s) = 0$ f.a goal states s
admissible: $h(s) \leq h^*(s)$ f.a states $s \in S$
consistent: $h(s) \leq cost(a) + h(s')$ f.a $s \xrightarrow{a} s'$

heuristic Theorem 1 + 2
T1: admissible $\implies$ safe + goal-aware
T2: goal-aware + consistent $\implies$ admissible
consistent alone not sufficient

Best-first Search
$f(n) := g(n)$: uniform-cost search
$f(n) := h(n)$: greedy best-first search
$f(n) := g(n) + h(n)$: A*
$f(n) := g(n) + w \cdot h(n)$: weighted A*

Best-first Search Properties
in general
complete for finite set of states
(otherwise semi-complete)
(h must be safe if states with $h(s) = \infty$ are pruned)(?)

greedy best first
- suboptimal: solutions can be arbitrarily bad
- often very fast
- monotonic transform. of h do not affect behaviour

astar and weighted astar
- with reopen: optimal with admissible h
- no-reopen: optimal with admis. and consist. h

ida*
- inherits important properties of A and DFS:
- semi-complete
- (must be safe if states with h(s) = ∞ are pruned)
- optimal if h admissible
- space complexity O(length * branchingfactor)

CSP
$\langle V, dom, C \rangle$
V: nonempty, finite set of Vars
dom: function of nonempty domain to each $v \in V$
C: set of constrains $c = \langle scope, rel \rangle$
scope = vars and $rel(c) \subseteq dom(v_1) \times dom(v_2)...$

Constraints
- unary has a single var in scope
- binary has scope(c) = (u,v) for $u, v \in V, u \neq v$
- trivial if $rel(c) = dom(u) \times dom(v)$
- scope(c) = (u, v) is equiv. to scope(c') = (v,u)

Node/Arc/Path Consistency
- Node: no domains in confilict with unary
- Arc: if one domain is fully covered
- Path: no assingment breaks other vars

Logic

Reasoning
reasoning can be reduced to testing validity
Does $\Phi \models \psi$?
1. test if $(\wedge_{\varphi \in \Phi} \varphi) \to \psi$ is tautology
2. if yes, then $\phi \models \psi$ else not

Resolution
join regel, sentence ϕ valid iff. $\neg\phi$ unsatisfiable

DPLL
simplify: assigning value and simplify
unit clause heuristic: single variables are assigned
pure symbol heuristic: same sign assigned
- sound and complete (time(exponential))
- computes a model where ϕ is true
- unassigned vars can be chosen arbitrarely

Planning
- plan / proof for plan
- optimal planning and suboptimal planning

PDDL
$\langle V, I, G, A \rangle$
V: finite set of binary state vars
I: ,initial state, not mentioned are false
G: goal (set), not mentioned vars are irrelevant
A: actions $a = \langle$ pre, eff, cost $\rangle$
everything is conj. over V also pre(a), eff(a)

Strips
- Statespace $\mathcal{S}(\Pi) = < S, A, cost, T, s_0, S_* >$
- like PDDL
- but just true vals (actions) are important
- instead eff: add and delete action
- set notation for actions instead of conjunction

SAS+
$\langle V, dom, I, G, A \rangle$
- dom: finite nonempty domain for $v \in V$
- no delete (like Stripes)
- with assignments to domain
- same statespace as strips

Abstraction
- drop distinctions between certain states
- preserve the state space behavior as well as possible
- $\alpha: S \to S'$ be surjective
- $\mathcal{S}^\alpha = \langle S', A, cost, T', s'_0, S'_* \rangle$

Pattern Databases
$\pi_{\{p, t_A\}}$, e.g. $\{p \to L, t_A \to L\} = 2$

Delete Relaxation
- Estimate solution costs by considering a simplified planning task
- ignore delete action, $del(a^+) = \emptyset$
- admissible and consistent heuristic

Relaxed planning graph

Maximum and Additive Heuristic
- cost of var 0 in layer 0, otherwise min of pre
- cost of action/goal h^{max} max, h^{add} sum of pre + cost(a)
- termination criterion: stability if $V^i = V^{i-1}$
- heuristic value: value of goal vertex in last layer - h^{max} optimistic assumption, max of cost
- h^{add} pessimistic assumption, all precond must be reached independently
- both safe and goal aware
- h^{max} is admissible and consistent, h^{add} neither
- h^{add} not for optimal planning

FF Heuristic
- like h^{add} but with extra step

Games
$\langle S, A, T, s_0, S_*, u, player \rangle$
transition model $T: S \times A \to S \cup \{\perp\}$
set of terminal states: S_*
utility function u : $S_* \to \mathbb{R}$
player function player : S $S_* \to$ {max, min}

Minimax Discuss
- yields optimal strategy (Max at least root val)
- assumption that the opponent plays perfectly

Evaluation Function
- problem: game tree too big
- search only up to predefined depth
- depth reached: estimate the utility with h (eval-func)

Linear (weighted) Eval Functions
$w1f1 + w2f2 + +wnfn$
- w_i are weights, and f_i are features.
assumes that feature contributions are mut. indep.

Move Ordering
- domain-specific ordering function (e.g. chess)
- dyn. move-ordering (good in past, ids, prev iter.)

Monte-Carlo Tree Search
1. selection (Tree policy): to terminal node
2. expansion: add missing state to the tree
3. simulation (default policy): find leaf value
4. back propagation: average to starting to node

2 probabilistic

Decisions under uncertainty

Decision Theory
Decision Theory = probability + utility

Unconditional Probability
- a world ω, is one possible state
- described as assignments to variables, e.g $w_{2,3}$
- sample space Ω: finite set of all possible worlds
- $\Omega = \{w_{x,y} | condition\}$
- Probability model: for each ω a $P(\omega)$
- $\sum_{\omega \in \Omega} P(\omega) = 1$
- event φ is a subset of Ω
- example event: "has onions on it"(pizza)
- $P(\varphi)$ is the sum of probabilities
- random variable X is a property of the world
- random variable: $X: \Omega \to dom(X)$
- random variable example: dom(Onion) = T, F, Onion$(\omega_1) = F$
- joint probability distributions: $P(Salami \wedge Broccoli)$ or $P(Salami, Broccoli)$
- full joint distribution: over all random variables
- marginalization: summing out irrelevant variables $Y_1, ..., Y_m$
- margin: $P(X_1, .., X_n) = \sum_{y_1,..,y_m} P(X_1, .., X_n, y_1, .., y_m)$

Conditional Probability
- conditional probability $P(x|e)$ when e
- joint prob.: $P(x|e) = \frac{P(x,e)}{P(e)}$
- Product rule: $P(x, e) = P(x|e) \cdot P(e)$
- full joint distribution sufficient for all queries

Query F.J. Distribution Cookbook (conditional)
space: $O(k^n)$ for k values
time: $O(k^m)$ for m hidden values
given: f.j.dist. $P(O, M, S, B)$ and query $P(S, B|m)$, steps:
1. part. vars in query, evidence and hidden vars
2. select entries consistent with evidence (m)
3. sum out hidden vars (probs indep. of hidden)
4. normalize (add probs and divide each prob with it)

Chain Rule
- goal: expression of joint probabilities
- $P(x_1, ..., x_n) = P(x_n|x_1, ..., x_{n-1}) \cdot P(x_1, ..., x_{n-1})$
- $P(x_1, ..., x_n) = P(x_n|x_1, ..., x_{n-1}) \cdot P(x_{n-1}|x_1, ..., x_{n-1}) \cdot P/x_1, ..., x_{n-2})$
- $P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|x_1, ..., x_{i-1})$

Bayes' Rule
- allows to compute cond. probab. from its reverse
- $P(x|y) = \frac{P(y|x) \cdot P(x)}{P(y)}$

Independence
- $P(x, y) = P(x) \cdot P(y)$ f.a $x \in dom(X)$ and $y \in dom(Y)$
- $P(x|y) = P(x)$ and $P(y|x) = P(y)$

Markov decision process

Expectimax
decision node: max of util. of children
chance node: prob-weight-sum of util. of children
- similar to Minimax
- computes (correct) expected utilities
- also the action: yields the max. expect. utility
- but it can obtain a much higher or lower utility

Hilfszettel für Intro to AI
von Philip G., Seite 2 von 2

Stochastic Shortest Path Problem
$\langle S, A, c, T, s_0, S_* \rangle$
$T : S \times A \times S \rightarrow [0,1]$, $\sum_{s' \in S} T(s,a,s') = 1$

Disc. Reward Infinite-horizon MDPs
- MDPs acts forever, infinite horizon, can be cyclic
- every action yields a pos./neg. reward
- (not only utilities in terminal states)
- aim: max overall reward (not goal state)
- disc. factor: converge (despite properties)

MPD definition
(discounted reward) infinite-horizon MPD
$T = \langle S, A, R, T, s_0, \gamma \rangle$ with
S, A = finite sets; $\gamma \in (0,1)$ = disc factor
R = reward function $R : S \times A \times S \rightarrow \mathbb{R}$
T = transition function $T : S \times A \times S \rightarrow [0,1]$
we require $\sum_{s \in S} T(s,a,s') = 1.$

Policy Definition
$\pi : S \rightarrow A \cup \{\perp\}$ (s.t $\pi(s) \in A(s) \cup \{\perp\}$ f.a $s \in S$.)
$S_\pi(s)$ = set of reachable states from s under π

Bellman Equation
The state-value (expected reward):
$V_*(s) := \max_{a \in A(s)} Q_*(s,a)$
The action-value:
$Q_*(s,a) := \sum_{s' \in succ(s,a)} T(s,a,s') \cdot (R(s,a,s') + \gamma \cdot V_*(s'))$

Optimal Policy
$V_*(s)$ of Bellman (max expected reward from s)
optimal policy if:
- $\pi(s) \in \arg\max_{a \in A(s)} Q_*(s,a)$ f.a $s \in S_\pi(s_0)$
(gives best action for highest expected reward)
- expected reward of π in T is $V_*(s_0)$
(policy gives the optimal solution)

Value Functions
The state-value (expected reward):
$V_\pi(s) := Q_\pi(s, \pi(s))$ (s is in the reachable set S_π)
The action-value:
$Q_\pi(s,a) :=$
$\sum_{s' \in succ(s,a)} T(s,a,s') \cdot (R(s,a,s') + \gamma \cdot V_\pi(s'))$

Policy Evaluation (iterative)
evaluation goal: computes V_π for a π
$\hat{V}^0_\pi(s) \in \mathbb{R}$ arbitrary f.a. $s \in S$ update rule:
$V^i_\pi(s) = \sum_{s' \in succ(s,\pi(s))} T(s,\pi(s),s')(R(s,\pi(s),s') + \gamma \cdot V^{i-1}_\pi(s'))$
converges to the true state-values (if ϵ is small)
$\lim_{i \rightarrow \infty} \hat{V}^i_\pi(s) = V_\pi(s) \quad$ f.a $s \in S$

Policy Improvement (Greedy Action)
V be a state-value function for T
the set of greedy actions in s with respect to V is:
$A_v(s) = \arg\max_{a \in A(s)} \sum_{s' \in succ(s,a)} T(s,a,s') \cdot (R(s,a,s') + \gamma \cdot V(s'))$
policy π_V with $\pi_V(s) \in A_V(s)$ is a greedy policy

Policy Iteration
- starts with arbitrary policy π_0
- alternates π evaluation and π improvement
- as long as policy changes

Value Iteration
- searches over state-values (instead π)
- $(\hat{V}^0, \hat{V}^1, \hat{V}^2 ... \hat{V}_*)$
- $\hat{V}^{i+1}(s) :=$
$\max_{a \in A(s)} \sum_{s' \in succ(s,a)} T(s,a,s') \cdot (R(s,a,s') + \gamma \hat{V}^i(s'))$
- starts with arbitrary $\hat{V}^0$
- converges to optimal policy
- terminates when max change is smaller than ϵ
- max expect. value of the best action

3 Machine Learning

Components of a Learning Agent
- performance element: decision making comp.
- learning element: improves the agent's performance through experience
- critic: feedback component that evaluates the agent's behavior
- problem generator: explorative component that seeks new experience

Supervised Learning
- learn unknown target function f
- input: trainingset of labeled examples $(x, f(x))$
- output: hypothesis/model h that is similar to f
- classification: learn f with discrete output
- regression: learn f with real valued output

Entropy
goal: degree of uncertainty of a random variable
$H(x) := -(P(x) \cdot \log_2 P(x) + P(\neg x) \cdot \log_2 P(\neg x))$
higher entropy $\rightarrow$ higher remaining difficulty

Decision Trees
- iterativly build decision tree
- perform greedy search over possible extensions
- extend based on remaining difficulty

Decision Trees Cookbook
1. comp. entropy H of problem (or initial bool set)
2. compute H of features (weight-avrg of H)
3. choose feature
4. repeat

Decision Trees Step-by-Step (?)
1. compute current difficulty
2. compute H given we extend d-tree with a feature
3. substract prob-weight Hs from prev. difficulty to obtain info-gain for selected feature
4. extend d-tree with feature with highest info-gain

Linear Regression
- training set $\{(x_i, y_i) | 1 \leq i \leq N\}$
- goal: $h_w = w_1 \cdot x + w_0$
- Loss: $h_w = \sum_{i=1}^{N} (y_i - h_w(x_i))^2$
- solution is h_w that minimizes loss
- method: partial derivatives to 0, unique solution
- $w_1 = \frac{N(\sum x_i y_1) - (\sum x_i)(\sum y_i)}{N(\sum x_i^2) - (\sum x_i)^2}$
- $w_0 = \frac{\sum y_i - w_1 \cdot \sum x_i}{N}$

Gradient Descent
- start: arbitrary w_0 and w_1, minimize
- move in direction of steepest descent in each step
- use learning rate (step size) α to adjust change
- converges to global optimum since loss is convex
- update rules:
$w_0 \leftarrow w_0 + \alpha \sum_{i=1}^{N} (y_i - h_w(x_i))$
$w_1 \leftarrow w_1 + \alpha \sum_{i=1}^{N} (x_i \cdot (y_i - h_w(x_i))$

Passive Reinforcement Learning

Difference Planning and Rl-learning
Planning:
has model of env.;deliberates to improve policy;(no interaction with env.)
RL agent:
has no or partial model;interacts with env.;improves policy indirectly by planning on learned model(model-based RL); improves policy directly by learning Q-values or policy (model-free RL)

Direct Utility Estimation Cookbook
- each path occurs proportional to Prob.
- converges to true state-values
$\hat{V}_\pi(s) \rightarrow V_\pi(s)$; converges to true state-values
1. trail: from s_0 to s_n terminal
2. observe: visited states and obtained rewards
3. increase: visitcounter $N(s_i)$ of all s_i
4. update: $\hat{V}_\pi(s_i)$
of all s_i to average of obtained rewards
$\hat{V}_\pi(s_i) \leftarrow \hat{V}_\pi(s_i) + \frac{\sum_{j=i}^{n} \gamma^{j-i} \cdot r_j - \hat{V}_\pi(s_i)}{N(s_i)}$
multiply V with previous N add new reward and divide through new N

Adaptive Dynamic Programming
TODO

Passive Temporal Difference
- converges: with a lot of steps and small α
- update: propportional to probability
- converges to true state-values under π
- update immediate and no terminal states (but slow convergence)
1. execute action $\pi(s)$ in s
2. observe successor s' and reward r
3. update $\hat{V}_\pi(s)$ s.t. difference between $\hat{V}_\pi(s)$ and $\hat{V}_\pi(s')$ better matches expectation:
$\hat{V}_\pi(s) \leftarrow \hat{V}_\pi(s) + \alpha(r + \gamma \hat{V}_\pi(s') - \hat{V}_\pi(s))$
($\hat{V}_\pi(s) = 0$ for terminal states s)
4. repeat

Active Reinforcement Learning

SARSA
- difference to temporal learning is that Q-values are estimated instead of States
- if fixed π converges to true s-values under π
- else converges to true state-values (GLIE)
1. select: action a' in s'
2. update: if s' was reached with a from s:
$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha \cdot (r + \gamma \hat{Q}(s',a') - \hat{Q}(s,a))$
($\hat{Q}(s,\cdot) = 0$ for terminal states s)
3. execute: a' and repeat

Exploration

Fixed Exploration
- exploration with constant N_e
- selects an action that has been executed $< N_e$
- if there no action, the greedy action is selected
- not GLIE: greedy in the limit + not explore forever

ϵ-greedy
- selects greedy action with (1 - ϵ)
- otherwise uniform distributed
- weakness: non greedy moves are treated equal
- not GLIE: explores forever + not greedy in the limit
- GLIE variant: decaying ϵ

Softmax
- constant parameter $\tau > 0$
- select actions freq. proportionaly to estimate
- Boltzmann exploration: $P(n) \propto e^{\frac{\hat{u}(n)}{\tau}}$ (for min $-\hat{u}$)
- Boltzmann weakness: doesn't account for variance
- softmax can perform arbitrary bad - GLIE variant: decaying τ

UCB1
- upper confidence bound: coptimal in the limit
- select successor n' of n that maximizes $\hat{u}(n') + B(n')$
- select $B(n')$ s.t $u_*(n') \leq \hat{u}(n') + B(n')$
- Chernof-hoeffding: $\sqrt{\frac{2 \cdot \ln N(n)}{N(n')}}$

Q-Learning
1. select action a in s
2. execute a and observe r and s
3. update $\hat{Q}(s,a)$:
$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha \cdot (r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a))$
($\hat{Q}(s,\cdot) = 0$ for terminal states s)
4. repeat
- updates by using best action ion s' rather one that is exec.
- off policy: performs update on best possible subseq. decision
- GLIE: infin. exploration for convergence + greedy in the limit for low regret