Selecting Computations in Sequential Decision Problems

David Tolpin

Ben-Gurion University of the Negev
Beer Sheva, Israel

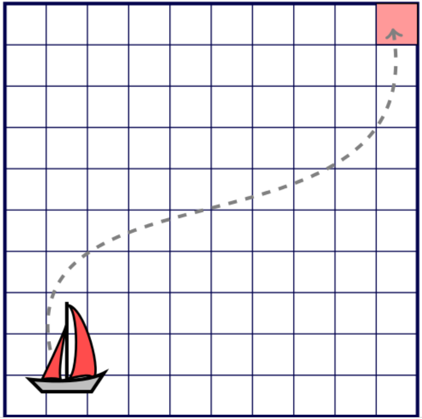
March 12, 2014

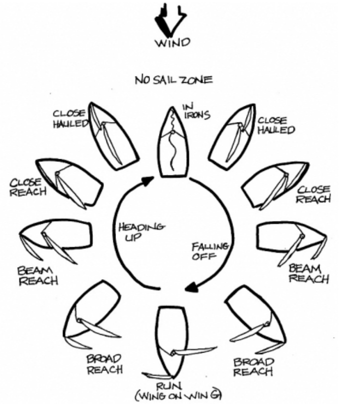
Outline

Sequential Decision Problems

- The agent selects actions based on outcomes of earlier actions.
- A solution is a *contingency plan*: a function from the history of actions and their outcomes to the next action.
- The optimal solution maximizes the *expected reward.* The reward is a known function of the history of actions and outcomes.

Sailing Domain





Multi-Armed Bandit

Board Games

- Backgammon
- Checkers
- Chess
- Computer Go

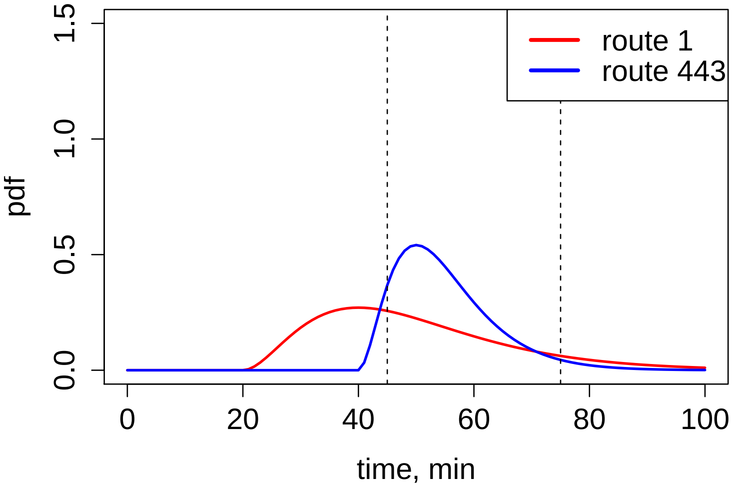
From Tel Aviv to Jerusalem

- the agent plans a journey from Tel Aviv to Jerusalem.
- there are two main routes - 1 and 443.
- the agent has a prior belief about travel time distribution.
- the agent can make a phone call to inquire about each of the routes.

From Tel Aviv to Jerusalem: Objective Function

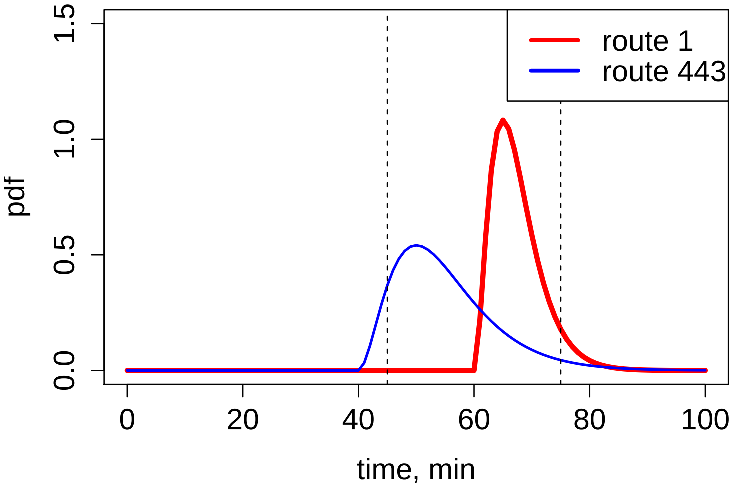
1. the agent wants to minimize the time on the road:
   $$u(t) = -t$$
2. the agent has to arrive by a particular time T:
   $$u(t) = 1 \text{ if } t \leq T, -1 \text{ otherwise.}$$
3. the agent wants to minimize the time on the road, but has to arrive by a particular time:
   $$u(t) = 1 - \frac{t}{T} \text{ if } t \leq T, -1 \text{ otherwise.}$$

From Tel Aviv to Jerusalem: prior belief



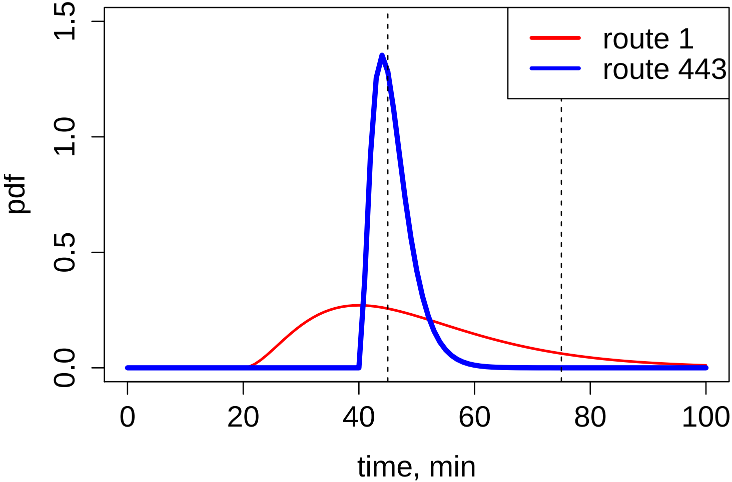
Prior belief.

From Tel Aviv to Jerusalem: updated 1



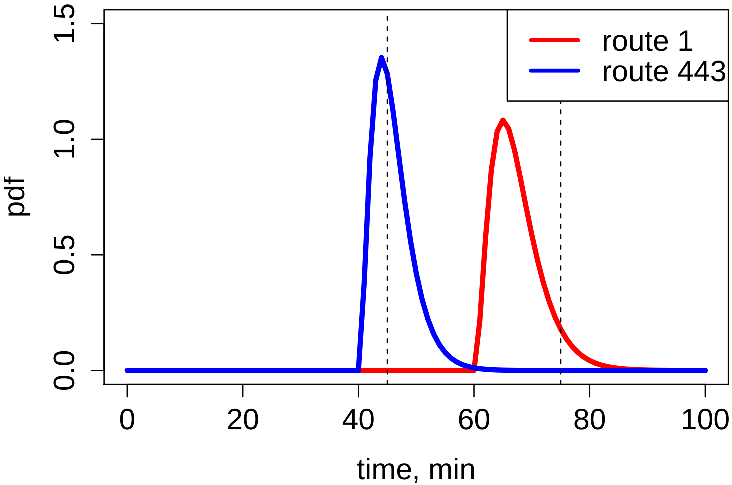
1 updated.

From Tel Aviv to Jerusalem: updated 443



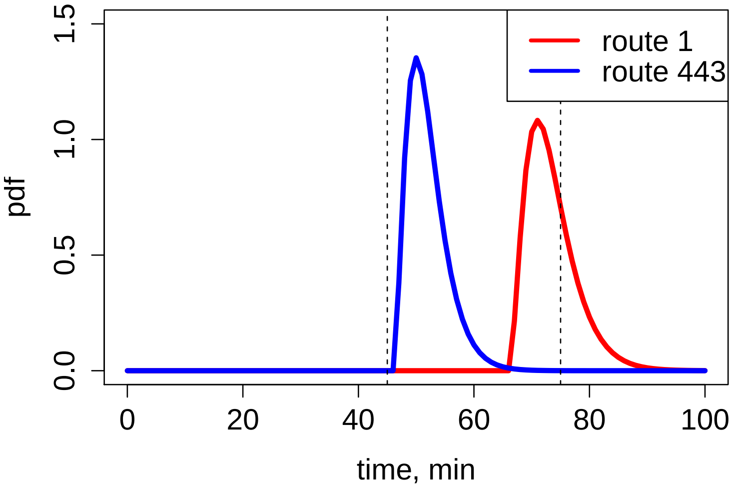
443 updated.

From Tel Aviv to Jerusalem: updated 1 and 443



Both 1 and 443 updated.

From Tel Aviv to Jerusalem: updated 1 and 443 with cost



Both 1 and 443 updated, phone call duration 3 minutes.

Outline

Rational Metareasoning

- A problem-solving agent can perform *base-level* actions from a known set $\{A_i\}$.
- Before committing to an action, the agent may perform a sequence of *meta-level* deliberation actions from a set $\{S_j\}$.
- At any given time there is a base-level action A_α that maximizes the agent's *expected utility.*

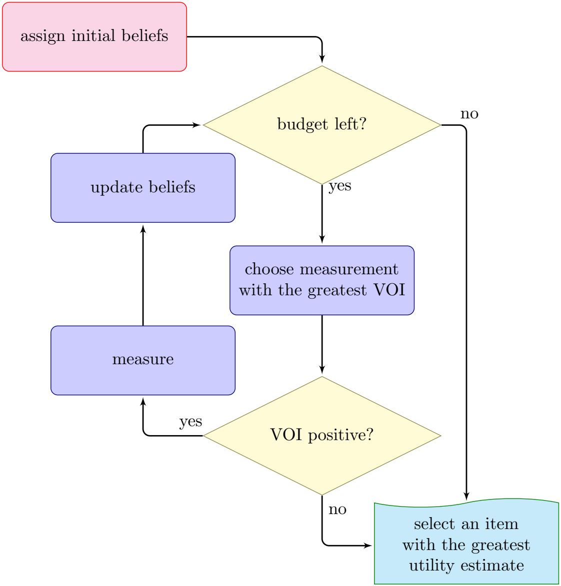
The **net VOI** $V(S_j)$ of action S_j is the **intrinsic VOI** Λ_j less the cost of S_j:

$$V(S_j) = \Lambda(S_j) - C(S_j)$$

$$\Lambda(S_j) = \mathbb{E}\left(\mathbb{E}(U(A^j_\alpha)) - \mathbb{E}(U(A_\alpha))\right)$$

- $S_{j_{\max}}$ that maximizes the net VOI is performed: $j_{\max} = \arg\max_j V(S_j)$, if $V(S_{j_{\max}}) > 0$.
- Otherwise, A_α is performed.

Greedy Algorithm



Simplifying assumptions

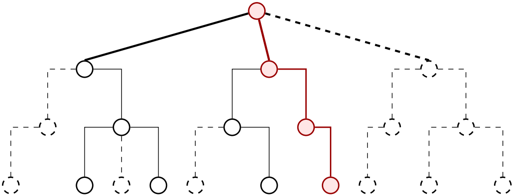
- **Myopicity:**
  - Meta-greedy: consider only the effect of single computatitonal actions.
  - Single-step: Assume all computations are complete, act as though a base-level action immediately follows the computation.
- **Subtree independence:** any computation updates the expected utility of a single action.

Outline

MCTS

Monte **C**arlo **T**ree **S**earch helps in large search spaces. At each node:

- Repeats:
  1. **Selection:** select an action to explore.
  2. **Simulation:** simulates a rollout until a goal is reached.
  3. **Backpropagation:** updates the action value.
- Selects the best action.



Adaptive Generally, MCTS samples 'good' moves more frequently, but sometimes **explores** new directions.

Multi-armed Bandit Problem

Multi-armed Bandit Problem:

- We are given a set A of K arms, $A = \{1..K\}$.
- Each arm can be pulled multiple times.
- The reward X^j for the jth arm is drawn from an **unknown** (but normally *stationary* and *bounded*) distribution with mean μ_j.
- The **total reward** must be maximized over the budget of N samples.

The **expected total regret** r_{total} of a sampling policy π for MAB is the difference between the expected reward from always pulling the best arm and the expected total reward of the policy.

$$r_{total} = N \max_{j \in A} \mu_j - \mathbb{E}(\sum_{i=1}^{N} x_i^j) \quad (1)$$

UCB

UCB is near-optimal for MAB — solves *exploration/exploitation* tradeoff.

- pulls an arm that maximizes **U**pper **C**onfidence **B**ound: $b_i = \overline{X}_i + \sqrt{\frac{c \log(n)}{n_i}}$
- the expected total regret is $O(\log n)$.

UCT

UCT (**U**pper **C**onfidence Bounds applied to **T**rees) is based on UCB.

- Adaptive MCTS.
- Applies the UCB selection scheme at each step of the rollout.
- Demonstrated good performance in Computer Go (MoGo, CrazyStone, Fuego, Pachi, ...) as well as in other domains.

However, the first step of a rollout is different:

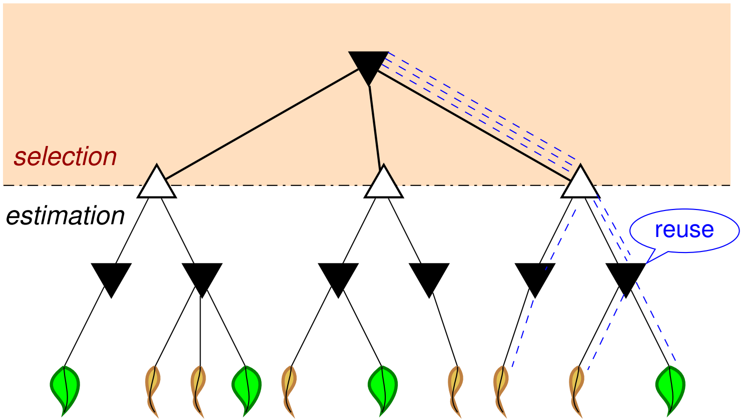
- The purpose of MCTS is to choose an action with the greatest utility.
- Therefore, the **simple regret** must be minimized.

Simple Regret

The **simple regret** of a sampling policy for MAB is the expected difference between the best expected reward $\max_{j \in A} \mu_j$ and the expected reward μ_j of the empirically best arm $\max_i \overline{X}_i$:

$$\mathbb{E} r_{simple} = \max_{j \in A} \mu_j - \mathbb{E}(\max_i \overline{X}_i) \tag{2}$$

MCTS Tree: Selection & Estimation



Upper Bounds on Value of Information

Assuming that:

1. Samples are i.i.d. given the value of the arm.
2. The expectation of a selection in a belief state is equal to the sample mean.

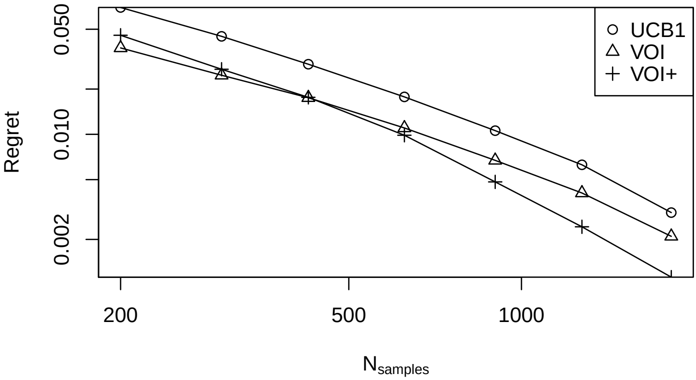
Upper bounds on intrinsic VOI Λ_i^b of testing the ith arm N times are (based on Hoeffding inequality):

$$\Lambda_\alpha^b < \frac{N\overline{X}_\beta^{n_\beta}}{n_\alpha + 1} \cdot 2\exp\left(-1.37(\overline{X}_\alpha^{n_\alpha} - \overline{X}_\beta^{n_\beta})^2 n_\alpha\right)$$
$$\Lambda_{i|i\neq\alpha}^b < \frac{N(1 - \overline{X}_\alpha^{n_\alpha})}{n_i + 1} \cdot 2\exp\left(-1.37(\overline{X}_\alpha^{n_\alpha} - \overline{X}_i^{n_i})^2 n_i\right)$$

Tighter bounds can be obtained (see the paper).

VOI-based Sampling in Bernoulli Selection Problem

25 arms, 10000 trials:



UCB1 is always worse than VOI-aware policies (VOI, VOI+).

Sampling in Trees

- Hybrid sampling scheme:
  1. At the *root node*: sample based on the VOI estimate.
  2. At *non-root nodes*: sample using UCT.
- Stopping criterion: Assuming sample cost c is known, stop sampling when intrinsic VOI is less than $C = cN$:

$$\frac{1}{N}\Lambda_\alpha^b \le \frac{\overline{X}_\beta^{n_\beta}}{n_\alpha+1}\Pr(\overline{X}_\alpha^{n_\alpha+N} \le \overline{X}_\beta^{n_\alpha}) \le c$$

$$\frac{1}{N}\max_i \Lambda_i^b \le \max_i \frac{(1-\overline{X}_\alpha^{n_\alpha})}{n_i+1}\Pr(\overline{X}_i^{n_i+N} \ge \overline{X}_\alpha^{n_\alpha}) \le c$$

$$\forall i : i \ne \alpha$$

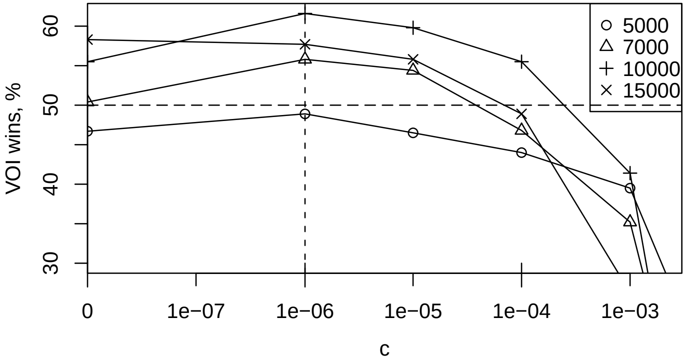
Sample Redistribution

- The VOI estimate assumes that the information is **discarded** between states.
- MCTS **re-uses rollouts** generated at earlier search states.

- Either incorporate 'future' influence into the VOI estimate (*non-trivial!*).
- Or behave myopically w.r.t. search tree depth:
  1. Estimate VOI as though the information is discarded.
  2. Stop early if the VOI is below a certain threshold.
  3. Save the unused sample budget for search in future states.
- The cost c of a sample is

  the VOI of increasing a future budget by one sample.

Playing Go Against UCT: Tuning the Sample Cost

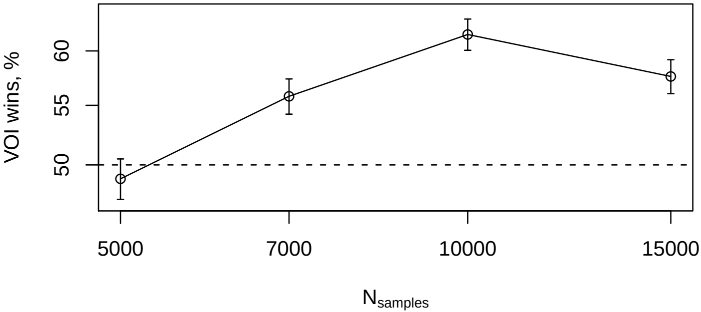


Best results for sample cost $c \approx 10^{-6}$:

winning rate of **64%** for 10000 samples per ply.

Playing Go Against UCT: Winning Rate vs. Number of Samples per Ply

Sample cost c fixed at 10^{-6}:



Best results for *intermediate* $N_{samples}$:

- When $N_{samples}$ is too low, poor moves are selected.
- When $N_{samples}$ is too high, the VOI of further sampling is low.

Outline

A^*

Apply all heuristics to initial state s_0
Insert s_0 into OPEN
while OPEN *not empty* **do**
  $n \leftarrow$ best node from OPEN
  if *Goal(n)* **then**
    return trace(n)
  foreach *child c of n* **do**
    Apply h_1 to c
    insert c into OPEN
  Insert n into CLOSED
return FAILURE

Lazy A^*

```
Apply all heuristics to initial state s_0
Insert s_0 into OPEN
while OPEN not empty do
    n ← best node from OPEN
    if Goal(n) then
        return trace(n)
    if h_2 was not applied to n                                   then
        Apply h_2 to n
        re-insert n into OPEN
        continue        //next node in OPEN
    foreach child c of n do
        Apply h_1 to c
        insert c into OPEN
    Insert n into CLOSED
return FAILURE
```

Rational Lazy A^*

```
Apply all heuristics to initial state s_0
Insert s_0 into OPEN
while OPEN not empty do
    n ← best node from OPEN
    if Goal(n) then
        return trace(n)
    if h_2 was not applied to n  and h_2 is likely to pay off then
        Apply h_2 to n
        re-insert n into OPEN
        continue        //next node in OPEN
    foreach child c of n do
        Apply h_1 to c
        insert c into OPEN
    Insert n into CLOSED
return FAILURE
```

Rational Decision

- When does computing h_2 pay off?
- Suppose h_2 was computed for state s. Then either:
  1. s will be expanded later on anyway
  2. an optimal goal is found before s is expanded
- Computing h_2 pays off only in outcome 2 — call this "h_2 is helpful"

"It is difficult to make predictions, especially about the future"

— Yogi Berra / Neils Bohr

Towards a Rational Decision

- Myopic assumption: this is the *last* meta-level decision to be made, and henceforth the algorithm will act like lazy A^*.
- When a node re-emerges from the open list, compare the regret of computing h_2 as in lazy A^*, vs. just expanding the node.
- Note: if rational lazy A^* is indeed better than lazy A^*, the myopic assumption results in an upper bound on the regret.

	Compute h_2	Bypass h_2
h_2 helpful	0	$\sim b(s)t_1 + (b(s) - 1)t_2$
h_2 not helpful	$\sim t_2$	0

$b(s)$ denotes the number of successors of s

Disclaimer: for the exact analysis, see the paper

From Regret to Rational Decision

	Compute h_2	Bypass h_2
h_2 helpful	0	$\sim b(s)t_1 + (b(s)-1)t_2$
h_2 not helpful	$\sim t_2$	0

- Suppose that the probability of h_2 being helpful is p_h
- Then the rational decision is to compute h_2 iff:

$$\frac{t_2}{t_1} < \frac{p_h b(s)}{1 - p_h b(s)}$$

Approximating p_h

$$\frac{t_2}{t_1} < \frac{p_h b(s)}{1 - p_h b(s)}$$

- We can directly measure t_1, t_2 and $b(s)$, but need to approximate p_h
- If s is a state at which h_2 was helpful, then we computed h_2 for s, but did not expand s. Denote the number of such states by B.
- Denote by A the number of states for which we computed h_2.
- We can use $\frac{A}{B}$ as an estimate for p_h

Empirical Evaluation: Weighted 15 Puzzle

- h_1 — weighted manhattan distance
- h_2 — lookahead to depth l with h_1

	Generated			Time		
l	A^*	LA^*	RLA^*	A^*	LA^*	RLA^*
2	1,206,535	1,206,535	1,309,574	0.707	0.820	0.842
4	1,066,851	1,066,851	1,169,020	0.634	0.667	0.650
6	889,847	889,847	944,750	**0.588**	0.533	0.464
8	740,464	740,464	793,126	0.648	**0.527**	0.377
10	611,975	611,975	889,220	0.843	0.671	**0.371**
12	**454,130**	**454,130**	807,846	0.927	0.769	0.429

Empirical Evaluation: Planning Domains

- h_{LA} — admissible landmarks
- $h_{\text{LM-CUT}}$ — landmark cut

Alg	Solved	623 Commonly Solved Time (GM)	Expanded	Generated
h_{LA}	698	1.18	183,320,267	1,184,443,684
$h_{\text{LM-CUT}}$	697	0.98	23,797,219	114,315,382
max	722	0.98	**22,774,804**	**108,132,460**
selmax	747	0.89	54,557,689	193,980,693
LA^*	747	0.79	**22,790,804**	**108,201,244**
RLA^*	**750**	**0.77**	25,742,262	110,935,698

- RLA^* solves the most problems, and is fastest on average

Outline

Insights into the Methodology

- Rational metareasoning works best when:
  1. Ubiquitous heuristic evaluation of the search space *decreases* the total search time.
  2. The heuristic computation time constitutes *a significant part* of the total search time.
- It is important to identify the right metareasoning decision.
- Simple utility and information model serve well.
- Tunable parameters should reflect algorithm implementation rather than problem set.

Bibliography

1. Nicholas Hay, Stuart J. Russell, David Tolpin, and Solomon Eyal Shimony. Selecting computations: Theory and applications. In UAI, pages 346–355, 2012.
2. Stuart Russell and Eric Wefald. Do the right thing: studies in limited rationality. MIT Press, Cambridge, MA, USA, 1991.
3. Eric J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In Proceedings of the 1987 Workshop on Uncertainty in Artificial Intelligence, pages 429–444, 1987.
4. Levente Kocsis and Csaba Szepesvḿakeari. Bandit-based Monte-Carlo planning. In ECML, pages 282–293, 2006.
5. Peter Auer, Nicolá Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the Multiarmed-bandit problem. Mach. Learn., 47:235–256, May 2002.

Thank You