Stochastic Scheduling and MDPs

Pietro Sala

Data Mining 24/25 - Exercise 3

1 Stochastic Scheduling

A stochastic scheduling problem is defined as a tuple $Sch = (\mathcal{T}, \mathcal{C}, \Delta, \pi)$ where:

- $T = \{T_1, \ldots, T_n\}$ is a finite set of tasks
- $\mathcal{C}$ is a set of temporal constraints of the form:

$$T_i^{*_1} \leq_{[x,y]} T_j^{*_2} \text{ where } *_1, *_2 \in \{b, e\}$$

- $\Delta : T \to \mathcal{P}(\mathbb{N}^+)$ is the duration function that maps tasks to intervals over positive natural numbers:

$$\Delta : T \to \mathbb{I}(\mathbb{N}^+)$$

  Where $\mathbb{I}(\mathbb{N}^+) = \{[x, y] \mid x \leq y, x, y \geq 1, x, y \in \mathbb{N}\}$ For example: $\Delta(T_i) = [5, 10]$

- $\pi : T \times \mathbb{N}^+ \to [0, 1]$ is a probability distribution function over task durations satisfying:

$$\forall T \in T : \sum_{n \in \Delta(T)} \pi(T, n) = 1$$

This formalization captures:

1. The set of tasks to be scheduled
2. Temporal constraints between task start and end times
3. Possible duration intervals for each task
4. Probability distribution over task durations within their intervals

For a task T_i, a typical probability distribution might be:

$$\begin{aligned}\pi(T_i, 5) &= 0.1\\ \pi(T_i, 6) &= 0.2\\ \pi(T_i, 7) &= 0.15\\ \pi(T_i, 8) &= 0.15\\ \pi(T_i, 9) &= 0.3\\ \pi(T_i, 10) &= 0.1\end{aligned}$$

where these values represent the probability of the task taking each possible duration within its interval $\Delta(T_i) = [5, 10]$.

2 Plan

A plan P for Sch is defined as a function:

$$P : T \to \mathbb{I}(\mathbb{N})$$

where $\mathbb{I}(\mathbb{N}) = \{[x, y] \mid x \leq y \wedge x, y \in \mathbb{N}\}$ such that:

- For all tasks $T \neq T' \in \mathcal{T}$ with $P(T) = [x, y]$ and $P(T') = [x', y']$ we have $x \neq x'$ (i.e., we can start at most one task for each unit of time)
- For all tasks $T \in \mathcal{T}$ with $P(T) = [x, y]$, the duration $y - x$ must belong to $\Delta(T)$ (i.e., the duration of each task must be consistent with its allowed duration interval)

Given an interval $[x, y]$, we define two accessor functions:

- $b([x, y]) = x$ returns the beginning of the interval
- $e([x, y]) = y$ returns the end of the interval

Given $T_i^{*_1} \leq_{[x,y]} T_j^{*_2}$ in $\mathcal{C}$, a plan P satisfies it,written $P \models T_i^{*_1} \leq_{[x,y]} T_j^{*_2}$, if and only if:

$$(*_2(P(T_j)) - *_1(P(T_i))) \in [x, y]$$

where $*_1$ and $*_2$ can be either the b or e accessor functions as specified in the constraint.

A plan P is *winning* if and only if for all constraints of the form $T_i^{*_1} \leq_{[x,y]} T_j^{*_2}$ in the set of inequalities, $P \models T_i^{*_1} \leq_{[x,y]} T_j^{*_2}$.

3 Stochastic System

A stochastic system is defined as a stochastic scheduling tuple $Sch = (\mathcal{T}, \mathcal{C}, \Delta, \pi)$ plus the structure for maintaining the current state of the system consisting of:

- A partial plan $P : \mathcal{T} \to \mathbb{I}(\mathbb{N})$ initialized to $\{\}$
- A clock time (or current time) $ct \in \mathbb{N}$ initialized to 0

3.1 Step Function

The system evolves according to a step function defined as:

$$\text{step} : \mathcal{T} \to \{0, 1, \text{fail}, \text{success}\}$$

The step function operates as follows:

- First checks if the current plan is consistent with all inequalities
- Verifies if all tasks have been assigned (complete plan)
- For unassigned tasks, adds them to the plan using the current time as the beginning point with a duration chosen according to π
- it always increase the current time by 1
- Returns:
  - fail: if plan becomes inconsistent
  - success: if plan is complete and consistent
  - 1: if the task completes in the current or in a previous step
  - 0: if the task is just started or if it is still running

Note that random_choices($\pi(T)$) selects a duration for task T according to the probability distribution π, and inconsistent$(P, \mathcal{C})$ returns true if and only there exists a temporal constraint $\mathcal{C}$ which is violated by P.

4 MDP for a stochastic Scheduling

A Markov Decision Process (MDP) M_{Sch} for a stochastic scheduling problem Sch is defined as a tuple $(Q, q_0, \mathcal{T}, \delta, \mathcal{L})$ where:

- Q is a finite set of states
- $q_0 \in Q$ is the initial state
- $\mathcal{T}$ is the set of tasks (action labels)
- $\delta : Q \times \mathcal{T} \times Q \to [0, 1]$ is the probabilistic transition function such that for all $q \in Q$ and $T \in \mathcal{T}$:

$$\sum_{q' \in Q} \delta(q, T, q') = 1$$

- $\mathcal{L} : Q \to \{0, 1, \text{success}, \text{fail}\}$ is the state labeling function

5 Positional Strategies for MDPs

A positional strategy for an MDP is a function $\varphi : Q \to \mathcal{T}$ that determines which task to select in each state. Given a strategy φ, we define the restricted transition relation δ_φ as:

$$\delta_\varphi(q, q') = \delta(q, \varphi(q), q')$$

We say that a state q' is reachable from q via δ_φ if there exists a sequence of states $q_1, ..., q_n$ with $q_1 = q$ and $q_n = q'$ such that for all $1 \leq i < n$ we have $\delta_\varphi(q_i, q_{i+1}) > 0$.

A positional strategy φ is *fair* if it satisfies:

- For any state q reachable from q_0 via δ_φ, there exists a state q' reachable from q via δ_φ such that:

$$\mathcal{L}(q') \in \{\text{success}, \text{fail}\}$$

- There exists a state q' reachable from q_0 via δ_φ such that:

$$\mathcal{L}(q') = \text{success}$$

In other words, a fair strategy ensures that:

1. Every reachable state can eventually reach either success or failure
2. Success is reachable from the initial state

6 Trace Generation and Definition

A trace for an MDP under strategy φ is a sequence $\mathcal{L}(q_0)\varphi(q_0)\mathcal{L}(q_1)\varphi(q_1)\ldots\varphi(q_{n-1})\mathcal{L}(q_n)$ where:

Algorithm 1 Step Function

```
1: function STEP(T : Task)
2:     if inconsistent(P, C) then return fail
3:     end if
4:     if Dom(P) = T then return success
5:     end if
6:     if T ∉ Dom(P) then
7:         P ← P ∪ {T ↦ [ct, ct + random_choices(π(T))]}
8:         if inconsistent(P, C) then return fail
9:         end if
10:        if Dom(P) = T then return success
11:        end if
12:        ct ← ct + 1 return 0
13:    end if
14:    if T ∈ Dom(P) then
15:        if ct ≥ e(P(T)) then
16:            r ← 1
17:        else
18:            r ← 0
19:        end if
20:        ct ← ct + 1
21:        return r
22:    end if
23: end function
```

- q_0 is the initial state
- For each $i < n$, $\delta_\varphi(q_i, q_{i+1}) > 0$
- $\mathcal{L}(q_n) \in \{\text{success}, \text{fail}\}$
- For all $i < n$, $\mathcal{L}(q_i) \in \{0, 1\}$

Given a fair strategy φ, we can generate a trace that respects the transition probabilities of the MDP using algorithm 2.

The algorithm works as follows:

- Starts from initial state q_0 and records its label
- While current state is not labeled success or fail:
  - Applies strategy φ to get next action
  - Randomly selects next state based on transition probabilities
  - Records action and next the label of the next state
- Returns complete trace ending in success or fail

Note that the random$(0, 1)$ function generates a uniform random number in $[0, 1]$, and the algorithm uses it to select next states according to their transition probabilities.

7 Exercise Assignment

The goal of this exercise is to implement and analyze a stochastic scheduling system through several interconnected steps:

a) Stochastic Schedule Example Design

- Create a simple stochastic schedule with at least 3 tasks
- Define temporal constraints between tasks
- Specify duration intervals and probability distributions
- Make sure that there exists at least one winning plan for the given stochastic schedule

b) Stochastic System Implementation

- Implement the step function as defined in previous sections

Algorithm 2 Generate Trace

function GENERATETRACE($M = (Q, q_0, \mathcal{T}, \delta, \mathcal{L}), \varphi$)
  $trace \leftarrow [\mathcal{L}(q_0)]$
  $current \leftarrow q_0$
  while $\mathcal{L}(current) \notin \{\text{success}, \text{fail}\}$ **do**
    $action \leftarrow \varphi(current)$
    $trace \leftarrow trace \cdot action$
    // $\cdot$ is the list append operation
    // Generate next state according to probability distribution
    $r \leftarrow \text{random}(0, 1)$ // Generate random number in [0,1]
    $sum \leftarrow 0$
    $next \leftarrow null$
    for $q' \in Q$ **do**
      $sum \leftarrow sum + \delta(current, action, q')$
      if $r \leq sum$ **then**
        $next \leftarrow q'$
        break
      end if
    end for
    $current \leftarrow next$
    $trace \leftarrow trace \cdot \mathcal{L}(current)$
  end while
  return *trace*
end function

- Create necessary data structures for plans and constraints
- Include functions for constraint checking

c) MDP Extraction via Automata Learning

- Use Automata Learning Library (e.g., AALpy) to learn the MDP structure from the stochastic system
- Extract and visualize the learned MDP

d) Fair Strategy Implementation

- Implement an algorithm to generate a fair strategy for the MDP
- Visualize the strategy decisions as a subgraph of the original MDP.

e) Trace Generation

- Implement the trace generation algorithm
- Generate at least 1000 traces using the fair strategy
- Store traces in a suitable format for process mining (e.g., PM4Py)

f) BPMN/Petri Net Mining

- Convert the generated traces for the selected process miner softaware (e.g., PM4Py)
- Apply process discovery algorithms
- Generate and visualize the BPMN/Petri Net model