As Soon As Probable

O. Maler, J.-F. Kempf, M. Bozga

VERIMAG
Grenoble, France

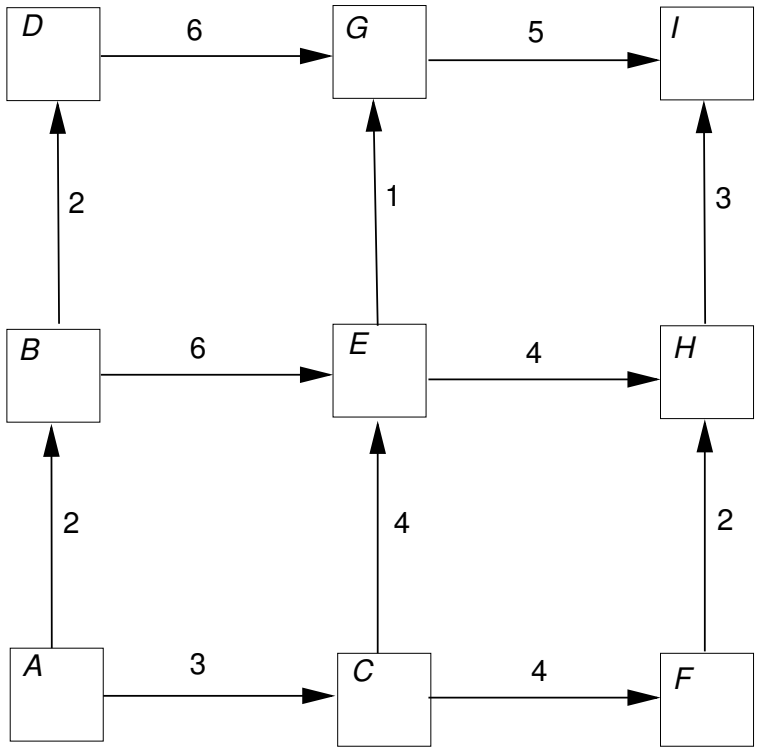
March 15, 2013

Executive Summary: Job Shop Scheduling under Uncertainty

- Several jobs, each being a sequence of steps that execute one after the other
- Each step duration is distributed **uniformly** over a bounded interval
- Some steps are conflicting (use the same resource) and cannot execute simultaneously
- When a step becomes enabled, a scheduler decides whether to **start** it or **wait** and let another job use the resource first
- A scheduler is evaluated according to the **expected** termination time of the last job (makespan)
- We synthesize optimal schedulers automatically using backward value/policy iteration (dynamic programming)

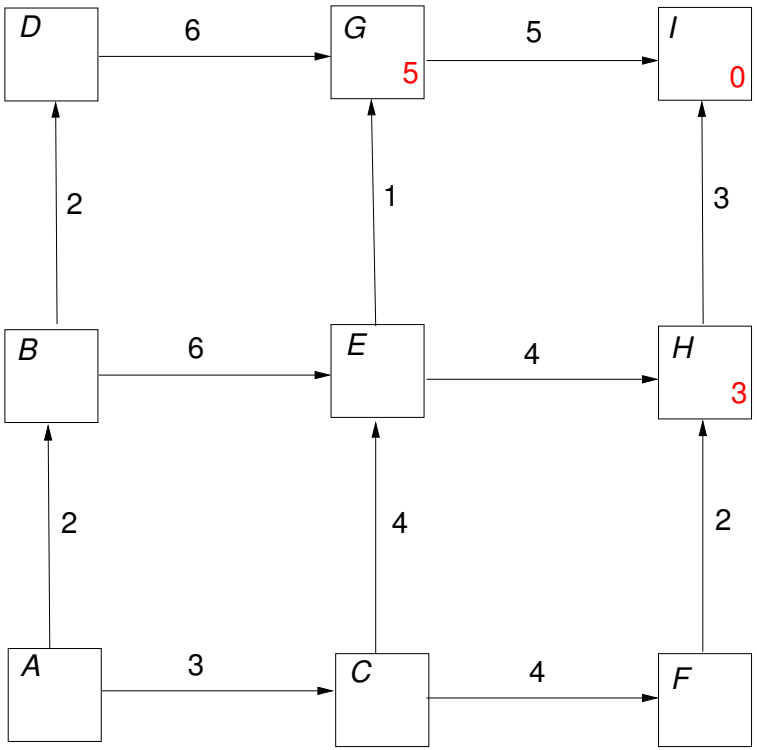
Plan

Computing Shortest Paths Backwards



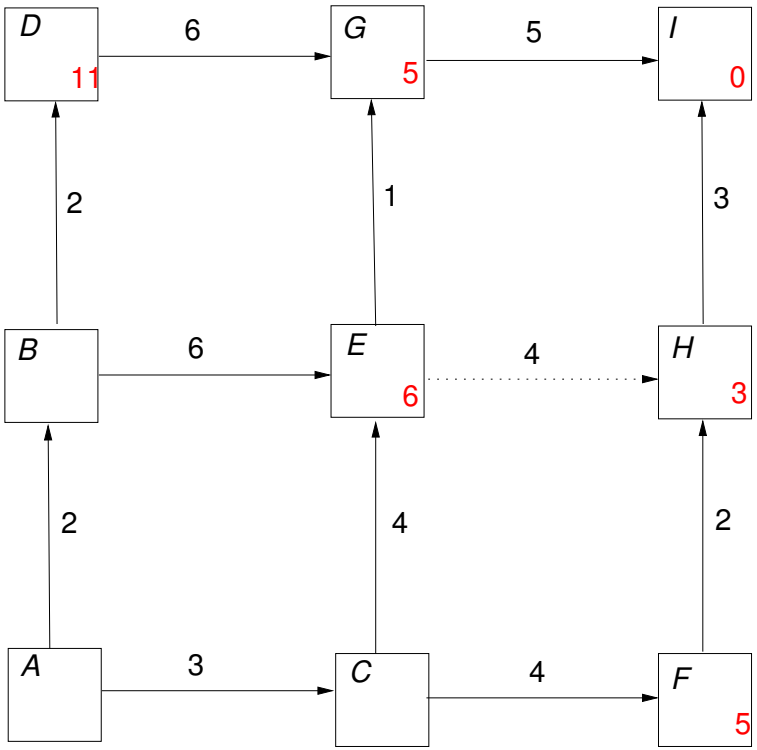
- We want to compute the shortest path from A to I
- We will do it backwards using a value function V on the nodes indicating the shortest path from the node to I
- Initially $V(I) = 0$
- Then $V(G) = 5 + V(I) = 5$ and $V(H) = 3 + V(I) = 3$

Computing Shortest Paths Backwards



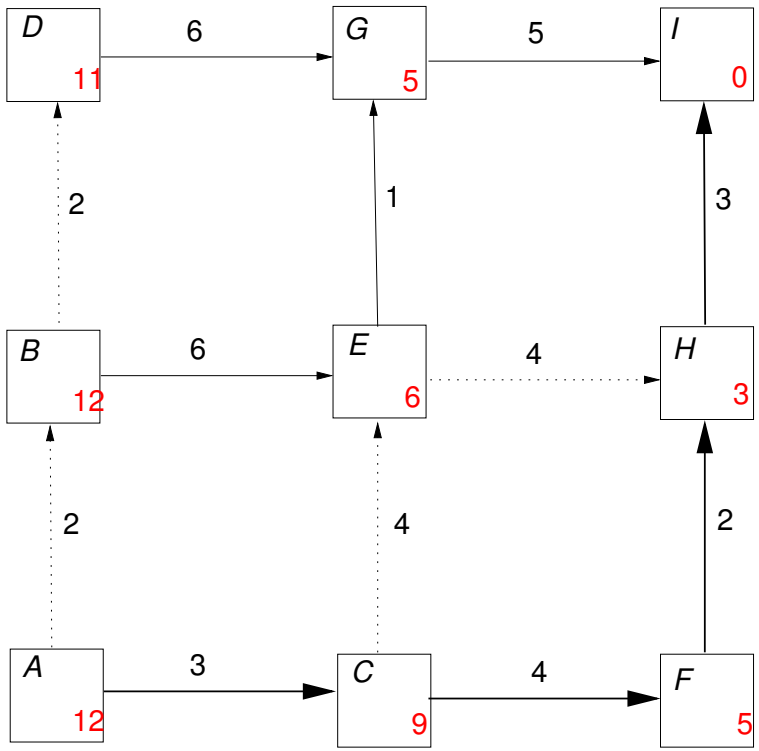
- Then $V(D) = 6 + V(G) = 11$ and $V(F) = 2 + V(H) = 5$
- To compute the value for E we need to make a local optimal choice:
- $V(E) = \min\{1 + V(G), 4 + V(H)\} = \min\{6, 7\} = 6$

Computing Shortest Paths Backwards



- $V(B) = \min\{2 + V(D), 6 + V(E)\} = \min\{13, 12\} = 12$
- $V(C) = \min\{4 + V(E), 4 + V(F)\} = \min\{10, 9\} = 9$
- $V(A) = \min\{2 + V(B), 3 + V(C)\} = \min\{14, 12\} = 12$

Computing Shortest Paths Backwards



- At the end we obtain the shortest path from *A* to *I*
- And in fact the shortest path from any state to *I*
- A “strategy”: which edge to choose in any state

From Deterministic to Stochastic Adversary

- We do something similar but more complex in two aspects
- In the graph example there was **no adversary** (or a trivial **deterministic** adversary):
- The length of an edge is fixed and known in advance
- Imagine that the length of an edge is drawn from a known bounded distribution
- We do not know the actual value when taking the decision

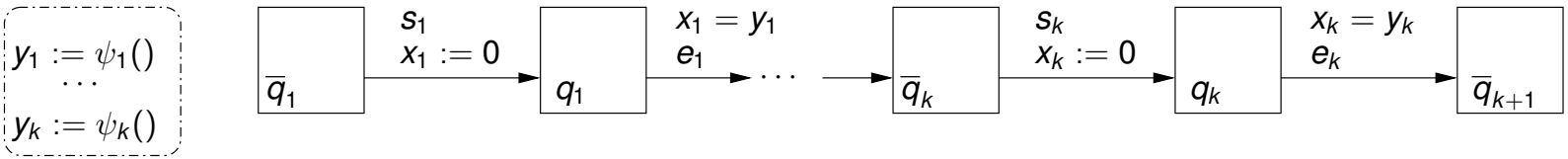
From Discrete to Continuous State-Space

- In the example there were only **discrete** decision points:
- For each node/state we have to choose an edge
- We work in **dense** time, several processes working **concurrently**
- We model scheduling problems in which whenever a task terminates, a scheduler should decide whether to **start** its successor or let it **wait**
- Such decision points, due to uncertainty, are spread **all over a continuous state-space** (with clock values)
- The value function and the strategy should be defined over all this uncountable state-space

Plan

A Single Process

- A job has k steps each with a duration ψ_j distributed uniformly over $I_j = [a_j, b_j]$
- A state-based representation by simple DPA:



- Waiting states $\overline{q}_j$ and active states q_j and two types of transitions:
- **start**: in idle state $\overline{q}_j$, a scheduler command s_j activates clock x and sets it to zero and moves to q_j
- **end**: in active state q_j, transition e_j, conditioned by the clock value $x = y_j$ moves to next waiting state
- We need some probabilistic preliminaries before we proceed

Time Densities

- A time density: a function $\psi : \mathbb{R}_+ \to \mathbb{R}_+$ satisfying

$$\int_0^\infty \psi[t]dt = 1$$

- Partial time density: $\int \psi[t]dt < 1$
- Bounded support: $\psi[t] \neq 0$ iff $t \in I = [a, b]$
- Uniform: $\psi[t] = 1/(b - a)$ inside its support $[a, b]$
- Distributions: $\psi[\leq t] = \int_0^t \psi[t']dt' \quad \psi[> t] = 1 - \psi[\leq t]$
- Time densities specify durations of steps as well as
- The remaining time to termination from a given **state**, an intermediate step for computing the value function

Operations on Time Densities: Convolution

- Convolution of two densities corresponds to the density of the duration of executing two steps one after the other
- For two densities ψ_1 and ψ_2 supported by $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$:
- The convolution $\psi_1 * \psi_2$ is a density ψ' supported by $I' = I_1 \oplus I_2 = [a_1 + a_2, b_1 + b_2]$
- $\psi'[t] = \int_0^t \psi_1[t']\psi_2[t - t']dt'$
- Intuition: rolling two dice

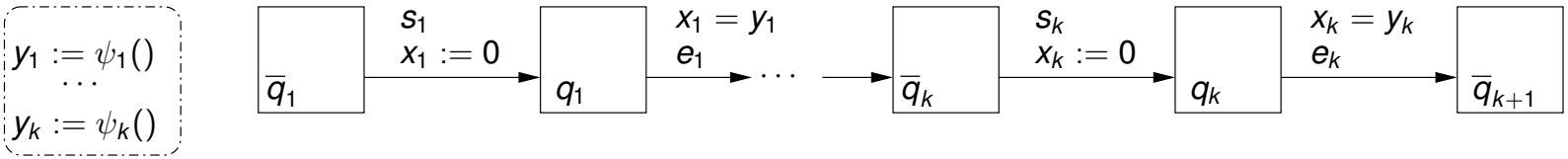
Operations on Time Densities: Shift

- For density ψ supported by $[a, b]$ the **residual** (conditional) density $\psi_{/x}$ is the time density **given** that $x < b$ time has already elapsed

$$\psi_{/x}[t] = \begin{cases} \psi[x+t] & \text{if } 0 < x \leq a \\ \psi[x+t] \cdot \frac{b-a}{b-x} & \text{if } a < x < b \end{cases}$$

- When $x < a$ it is simply a shift
- When $x > a$ we know that the duration is restricted to $[x, b]$ and have to normalize
- Remark: for exponential distribution: $\psi_{/x} = \psi$

The Problem



- We want to compute/optimize the (expected) arrival to the final state from **any** extended state
- An extended state is either a waiting state $(\overline{q}_j, \perp)$ or an active state (q_j, x) with x a clock value in $[0, b_j]$
- Without resource conflicts there is no use in waiting: **start** transitions are issued immediately by an optimal scheduler

Local Stochastic Time-to-Go

- The local stochastic time-to-go assigns to every state (q, x) a time density $\mu(q, x)$
- $\mu(q, x)[t]$ is the probability to terminate within t time *given* that we start from (q, x) and apply the optimal strategy

$$\mu(\overline{q}_{k+1}, \perp) = \mathbf{0} \tag{1}$$

$$\mu(\overline{q}_j, \perp) = \mu(q_j, 0) \tag{2}$$

$$\mu(q_j, x)[t] = \int_0^t \psi_{j/x}[t'] \cdot \mu(q_{j+1}, 0)[t - t']dt' \tag{3}$$

- Line (1) refers to the final state
- Line (2) says that in a waiting state you start immediately and inherit the time-to-go from the next state

Local Stochastic Time-to-Go

- $\mu(q, x)[t]$ is the probability to terminate within t time *given* that we start from (q, x) and apply the optimal strategy

$$\mu(q_j, x)[t] = \int_0^t \psi_{j/x}[t'] \cdot \mu(q_{j+1}, 0)[t - t']dt'$$

- The probability for termination at t is based on:
  - The probability of terminating the current step in some t'
  - The probability of the remaining time-to-go being $t - t'$
- Functionally speaking:

$$\mu(q_j, x) = \psi_{j/x} * \mu(q_{j+1}, 0)$$

- For the initial state this gives $\mu(q_1, 0) = \psi_1 * \cdots * \psi_k$

Expected Time-to-Go

- The local expected time-to-go function is $V : Q \times X \to \mathbb{R}_+$ defined as

$$V(q, x) = \int \mu(q, x)[t] \cdot t dt = \mathbb{E}(\mu(q, x))$$

- For the initial state this yields

$$V(q_1, 0) = \mathbb{E}(\psi_1 * \cdots * \psi_k) = \mathbb{E}(\psi_1) + \cdots + \mathbb{E}(\psi_k) = \sum_{j=1}^{k} (a_j + b_j)/2$$

- The same result we would obtain by forward computation (and common sense)
- But it is important that we can compute it backwards and for any clock value

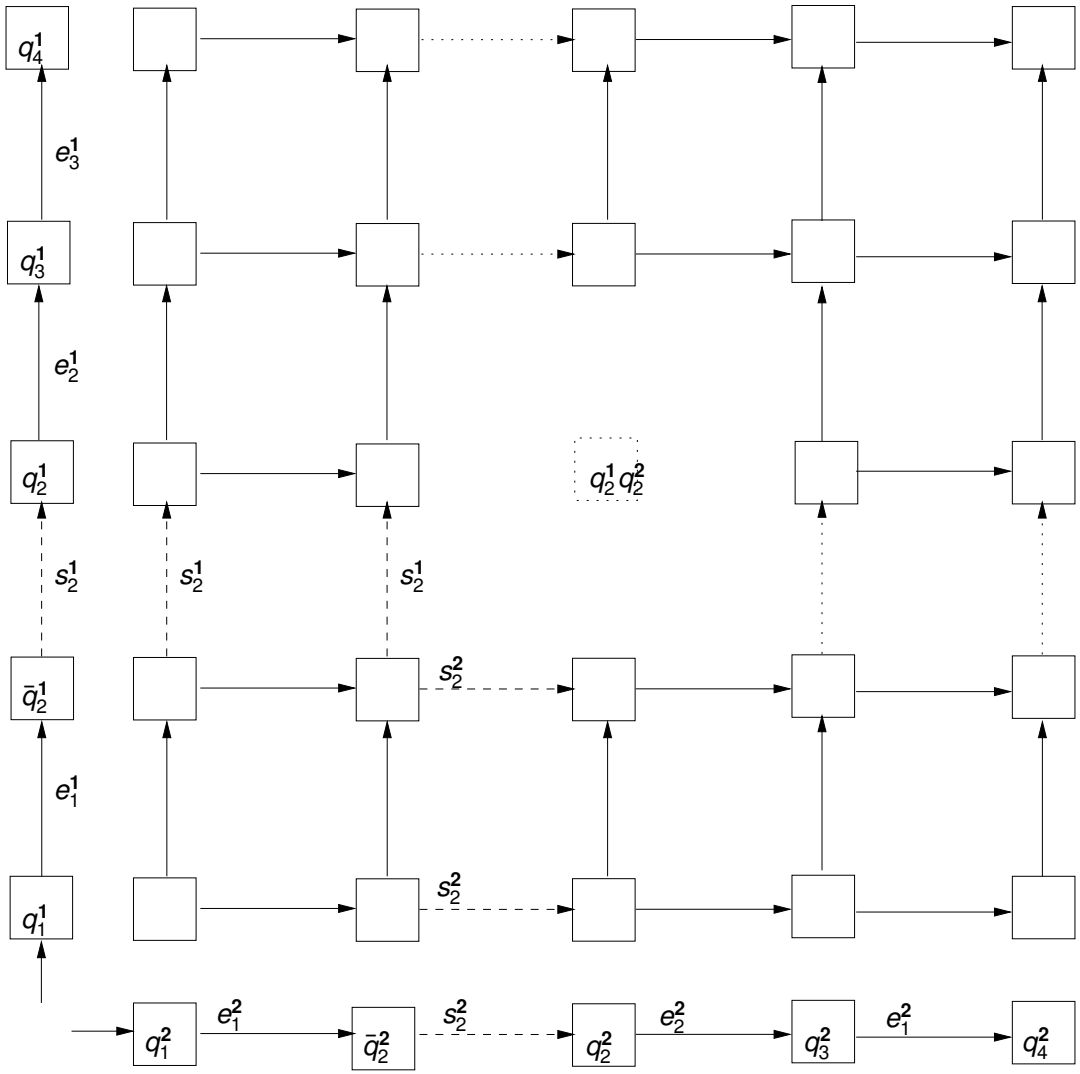
Plan

Multiple Processes

- Consider now a system consisting of n jobs $P^{\mathbf{1}} || \cdots || P^{\mathbf{n}}$ and a **conflict relation** between steps that use the same resource
- Two conflicting steps $P_j^{\mathbf{i}}$ and $P_{j'}^{\mathbf{i}'}$ cannot execute **simultaneously**
- The whole system is modeled as a product of the automata for the individual jobs
- Forbidden (conflicting) states are removed

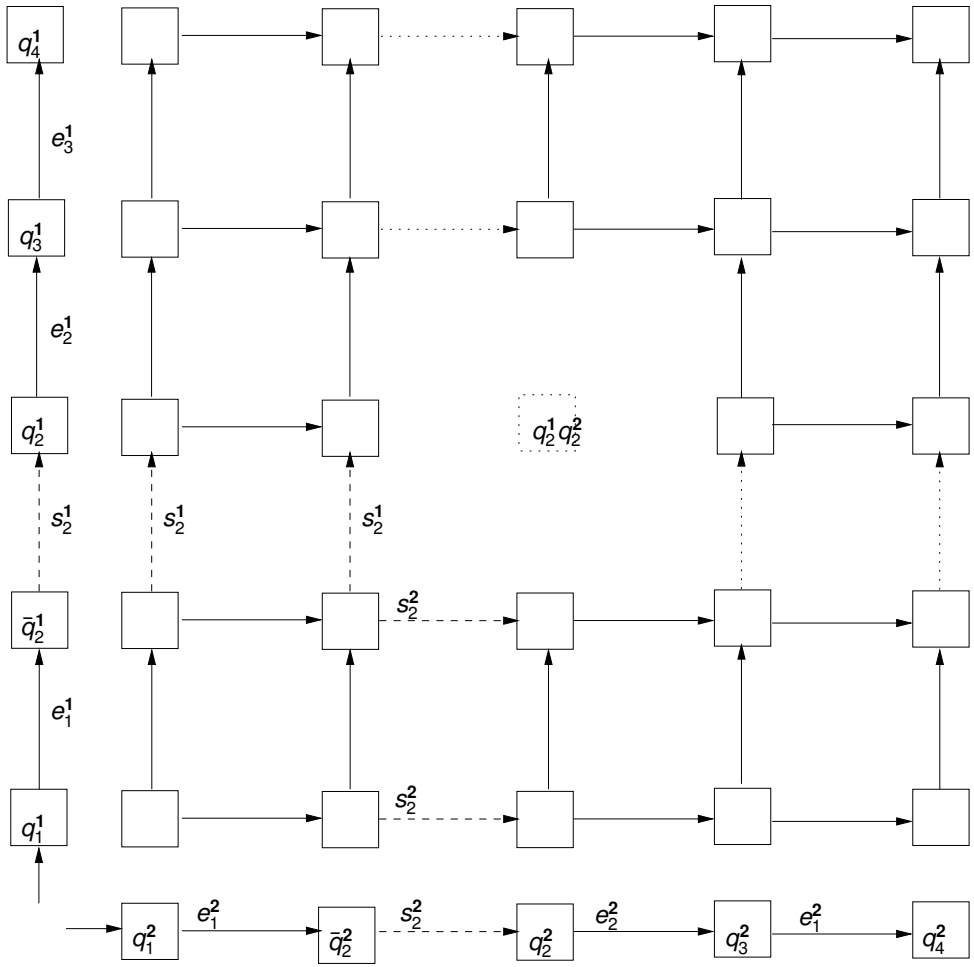
The Global Automaton

- Two jobs, and a conflict between their respective second steps



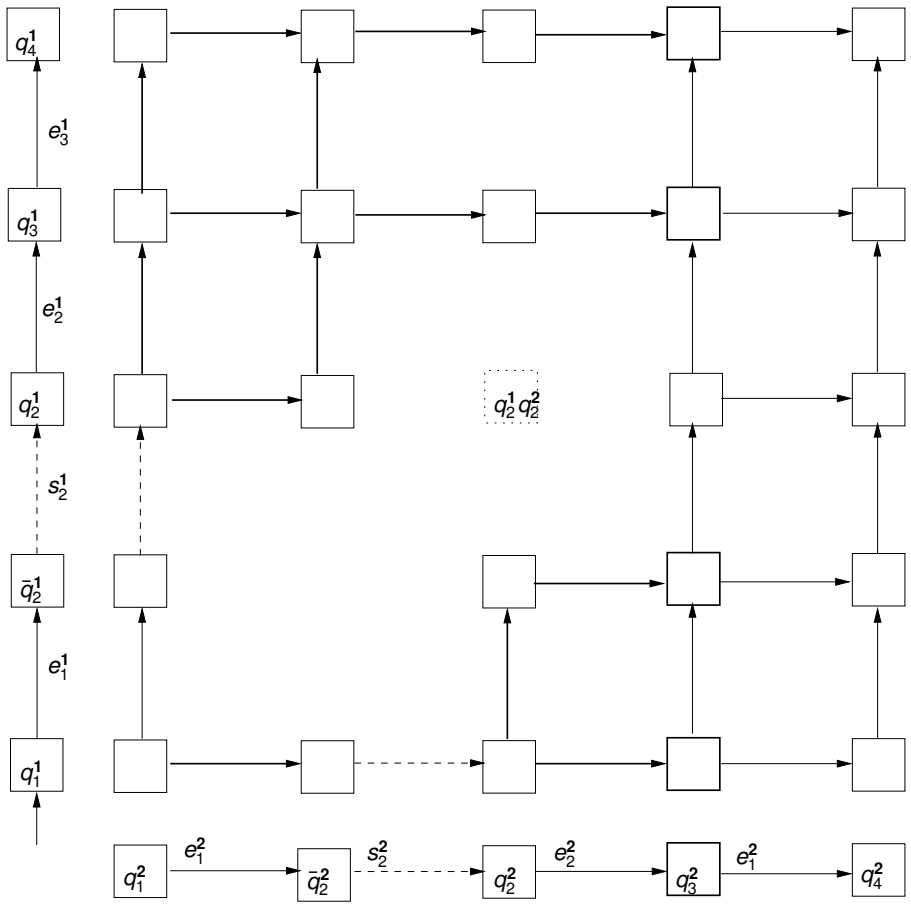
Probabilistically Incorrect

- There is set-theoretical non-determinism in states where we can either **start** or **wait**
- This non-determinism is resolved by a scheduling strategy



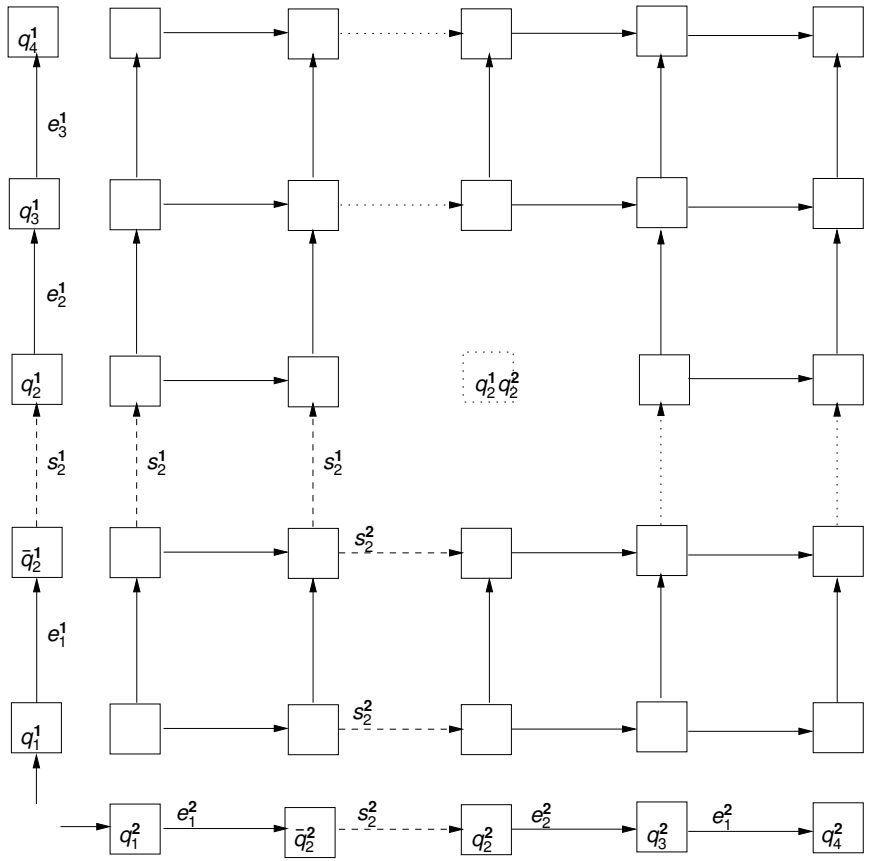
Schedulers

- A scheduler $\Omega : S \to \Sigma_s \cup \{\mathbf{w}\}$ says whether to start or wait
- Composing with a scheduler we have a stochastic process
- Every value of random variable y induces a single behavior
- Example: a FIFO scheduler (never wait):



Preview of Major Contribution

- Compute the optimal scheduler and its value backwards
- Any extended state (q, x) defines value functions: the distribution and expected value of termination times starting from (q, x)



Local Stochastic Time-to-Go

- Consider a state q such that the optimal strategy has been computed for all its successors
- With every $s \in \Sigma_s \cup \{\mathbf{w}\}$ enabled in q, we associate a time density

$$\mu^{\mathbf{i}}(q, x, s) : \mathbb{R}_+ \to [0, 1]$$

- It is the stochastic time-to-go for process $P^{\mathbf{i}}$ if the controller issues action s at state (q, x)
- and continues from there according to the optimal strategy

Global and Expected Time-to-Go

- For a state (q, x) and action s enabled in it:
- The stochastic time-to-go for total termination (makespan, termination of last step) :

$$\mu(q, x, s) = \max\{\mu^{\mathbf{1}}(q, x, s), \ldots, \mu^{\mathbf{n}}(q, x, s)\}$$

- The expected total termination time:

$$V(q, x, s) = \int t \cdot \mu(q, x, s)[t]dt$$

- We want to choose in each (q, x) the action s which minimizes $V(q, x, s)$

Plan

Abstract Algorithm

Input: A global state q such that $\Omega(q', x)$ and $\mu^{\mathbf{i}}(q', x)$ have been computed for each of its successors q' and every i

Output: $\Omega(q, x)$, $\mu^{\mathbf{i}}(q, x)$, $V(q, x)$ (strategy and value)

% COMPUTE:
forall $s \in \Sigma_s \cup \{\mathbf{w}\}$ enabled in q
compute $V(q, x, s)$ (expected makespan)
end

% OPTIMIZE:
forall $x \in Z_q$
$V(q, x) = \min_s(V(q, x, s))$
$s_* = \arg\min_s V(q, x, s)$
$\Omega(q, x) = s_*$
end

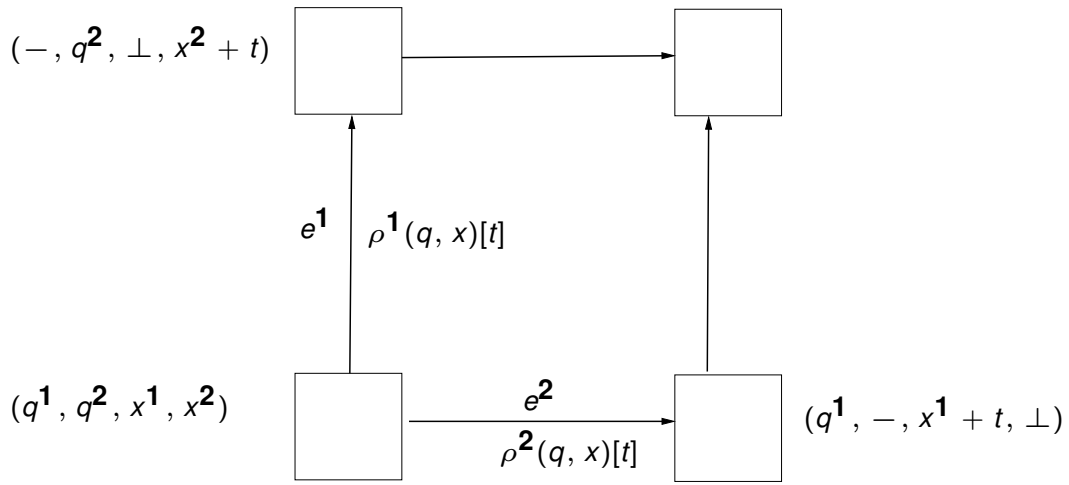
Computation and Optimization

- There are two main parts in the procedure:
- To **compute** the value of each action
  - This is immediate for **start** transitions
  - More complicated for **wait** when there are several active steps that may terminate in different orders
  - This require **race analysis**
- After that we need to **compare** the values of the actions and choose the optimal one
- The optimal action in a state may vary according to clock values

Race Winner

- Let q be a state where n processes are active, each in a step admitting a time density $\psi^{\mathbf{i}}$
- With every clock valuation $x = (x^{\mathbf{1}}, \ldots, x^{\mathbf{n}}) \leq (b^{\mathbf{1}}, \ldots, b^{\mathbf{n}})$ and every i define the partial density:

$$\rho^{\mathbf{i}}(q, x)[t] = \psi^{\mathbf{i}}{}_{/x^{\mathbf{i}}}[t] \cdot \prod_{i' \neq i} \psi^{\mathbf{i}'}{}_{/x^{\mathbf{i}'}}[> t]$$



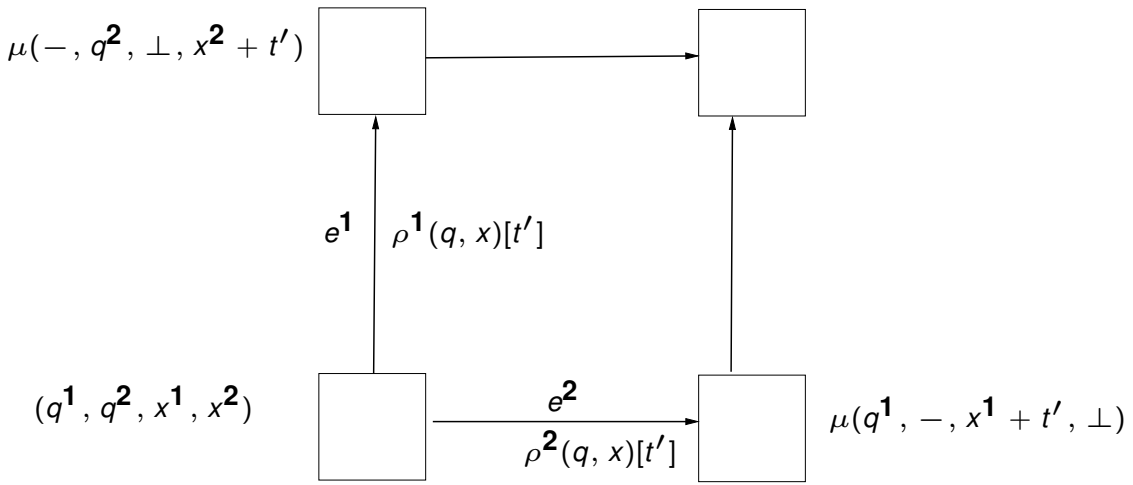
- This is the probability that $P^{\mathbf{i}}$ terminates in t time and every other process $P^{\mathbf{i}'}$ terminates within some $t' > t$

Computing Stochastic Time-to-go

- For every i, the function $\mu^{\mathbf{i}}$ is defined as

$$\mu^{\mathbf{i}}(q, x, \mathbf{w})[t] = \sum_{i'=1}^{n} \int_0^t \rho^{\mathbf{i}'}(q, x)[t'] \cdot \mu^{\mathbf{i}}(\sigma^{\mathbf{i}'}(t', q, x))[t - t']dt'$$

- $\sigma^{\mathbf{i}'}(t', q, x)$ is the extended state reached after waiting t' time and taking an $e^{\mathbf{i}'}$ transition



- This is **not** a convolution: the other clocks are **not** reset
- This operation is **computable** resulting in piecewise-continuous densities of a particular form

Zone-Polynomial Time Densities

- A function $\mu : Z \to (\mathbb{R}_+ \to [0,1])$ over a rectangular clock space Z is zone-polynomial if it can be written as

$$\mu(x^{\mathbf{1}}, \ldots, x^{\mathbf{n}})[t] =$$

$$\begin{cases} f_1(x^{\mathbf{1}}, \ldots, x^{\mathbf{n}})[t]) & \text{if} \quad Z_1(x^{\mathbf{1}}, \ldots, x^{\mathbf{n}}) \quad \text{and} \quad l_1 \leq t \leq u_1 \\ \ldots & \\ f_L(x^{\mathbf{1}}, \ldots, x^{\mathbf{n}})[t]) & \text{if} \quad Z_L(x^{\mathbf{1}}, \ldots, x^{\mathbf{n}}) \quad \text{and} \quad l_L \leq t \leq u_L \end{cases}$$

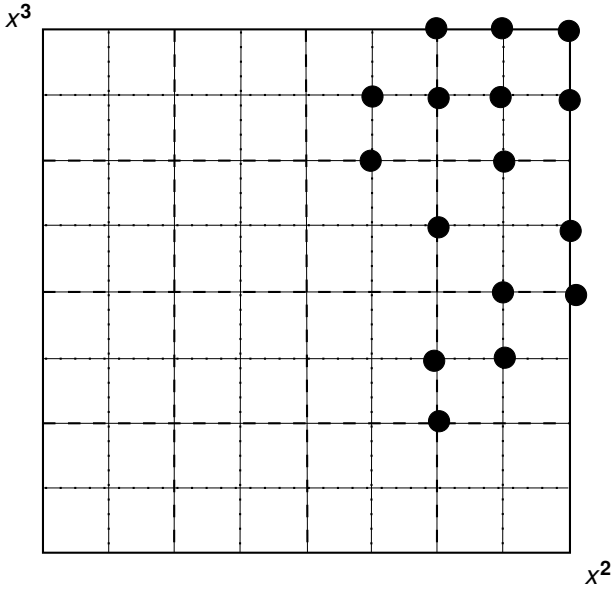
- Every $Z_r(x^{\mathbf{1}}, ..., x^{\mathbf{n}})$ is a zone included in the rectangle Z, satisfying either $Z_r \subseteq [x^{\mathbf{i}} \leq a^{\mathbf{i}}]$ or $Z_r \subseteq [a^{\mathbf{i}} \leq x^{\mathbf{i}}]$
- The bounds l_r, u_r of the t interval are either nonnegative integers c or terms of the form $c - x^{\mathbf{i}}$
- For every r, $f_r(x^{\mathbf{1}}, ..., x^{\mathbf{n}})[t] = \sum_k \frac{P_k(x^{\mathbf{1}},...,x^{\mathbf{n}})}{Q_r(x^{\mathbf{1}},...,x^{\mathbf{n}})} t^k$
- P_k are arbitrary polynomials and Q_r is a characteristic polynomial associated with zone Z_r defined as $\prod_i (b^{\mathbf{i}} - \max\{x^{\mathbf{i}}, a^{\mathbf{i}}\})$.
- Theorem: these are closed under the defined operations

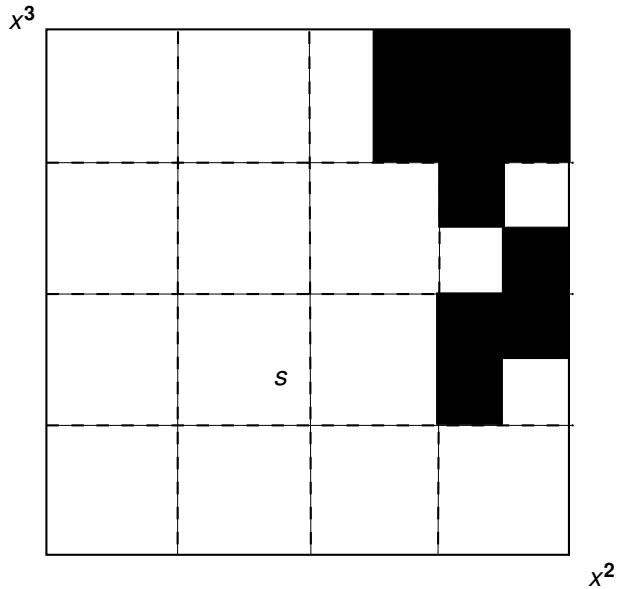
Optimization

- In a state when we have to choose between $s^{\mathbf{i}}$ and $\mathbf{w}$ we need to compare $V(q, x, s^{\mathbf{i}})$ and $V(q, x, \mathbf{w})$
- We need to partition the clock space of the active processes into $\Omega^{-1}(s^{\mathbf{i}})$ and $\Omega^{-1}(\mathbf{w})$
- The boundary is defined by a polynomial equality that we have not yet characterized
- We define an approximate strategy, whose error is bounded by the following lemma:
- Let V be the value function, then for every (q, x) and i $\frac{\partial V}{\partial x^{\mathbf{i}}}(q, x) \geq -1$
- You cannot progress to termination faster than the speed of time

Approximation

- Choosing between s^1 and $\mathbf{w}$ based on x^2 and x^3



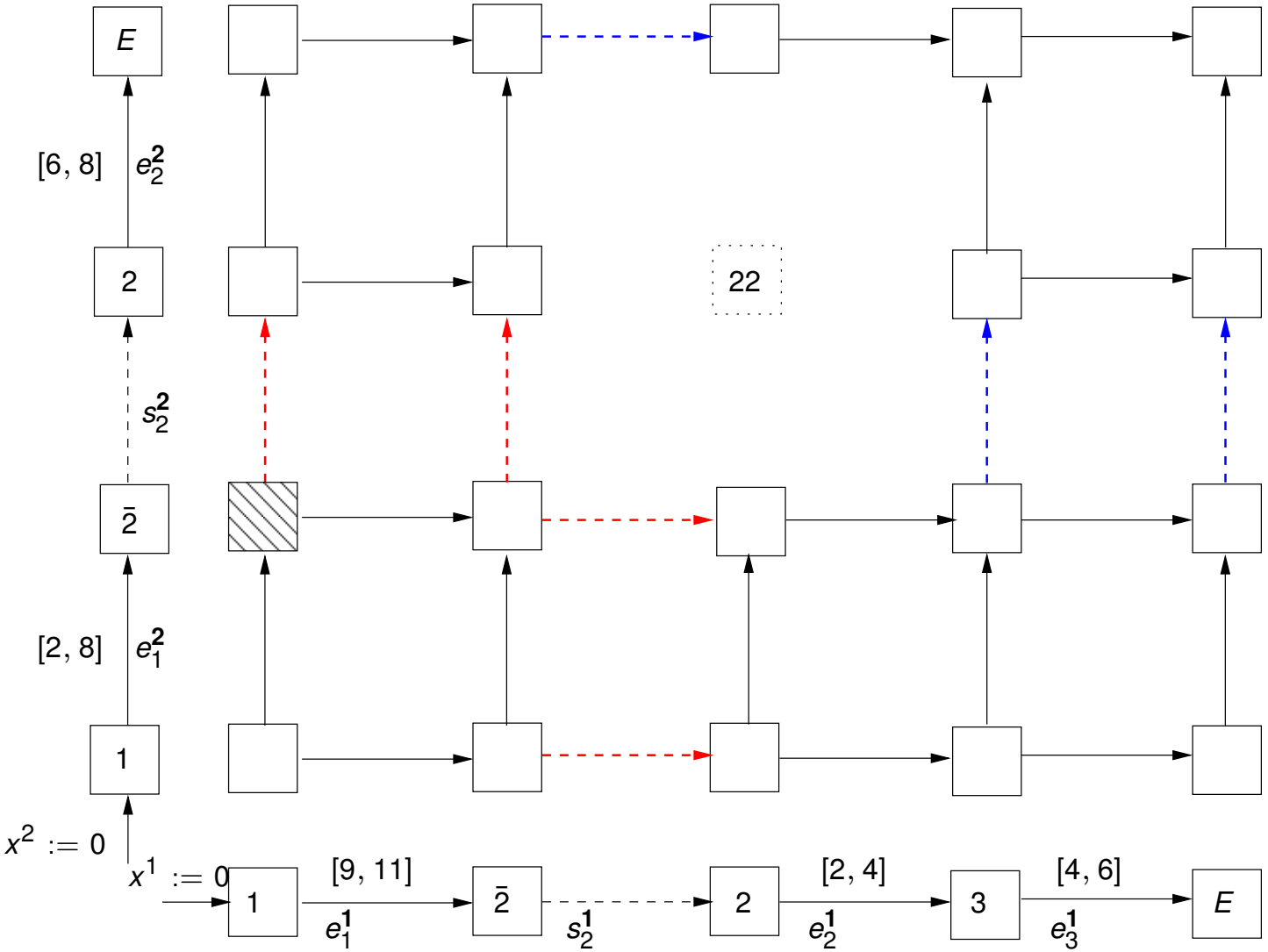


- Grid points with the dark circles are in $\Omega^{-1}(\mathbf{w})$
- The dark cubes indicate $\Omega'^{-1}(\mathbf{w})$
- The relation between the value of the optimal strategy and the approximate satisfies $V'(q, x') - V(q, x') \leq \varepsilon$

Summary of Results

- Theorem:
- Let Ω be the expected-time optimal scheduler whose value at the initial state is V
- For any ε, one can compute a scheduler Ω' whose value V' satisfies $V' - V \leq \varepsilon$
- Theorem:
- The optimum is attainable by non-lazy (active) schedulers
- If a step is kept waiting at some point there is no use in executing it later if the resource has not been used meanwhile (extended from the deterministic to the stochastic case)
- Implementation: almost complete with a dedicated symbolic integration package

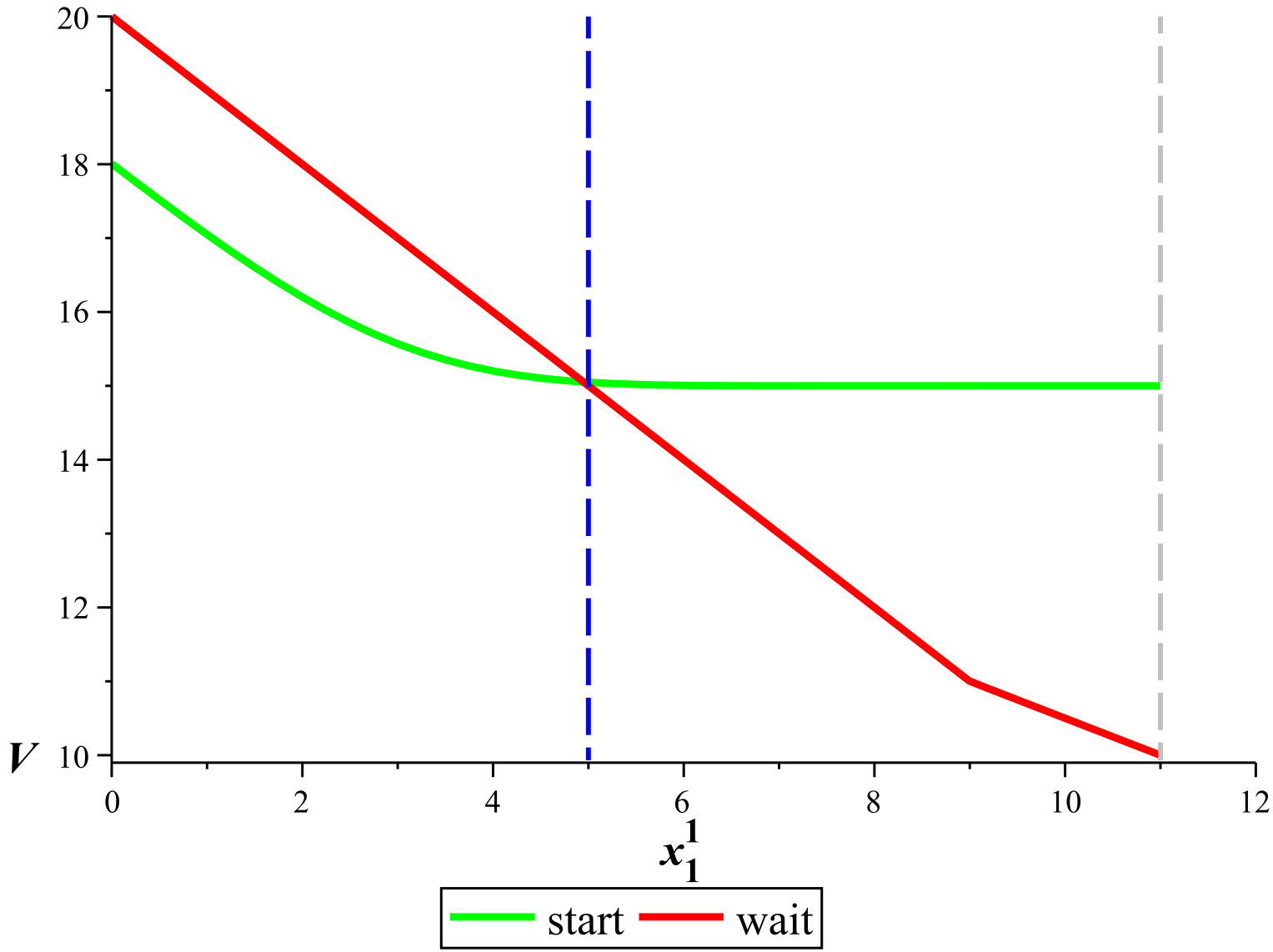
Example I



- When to do s^2 and when to wait

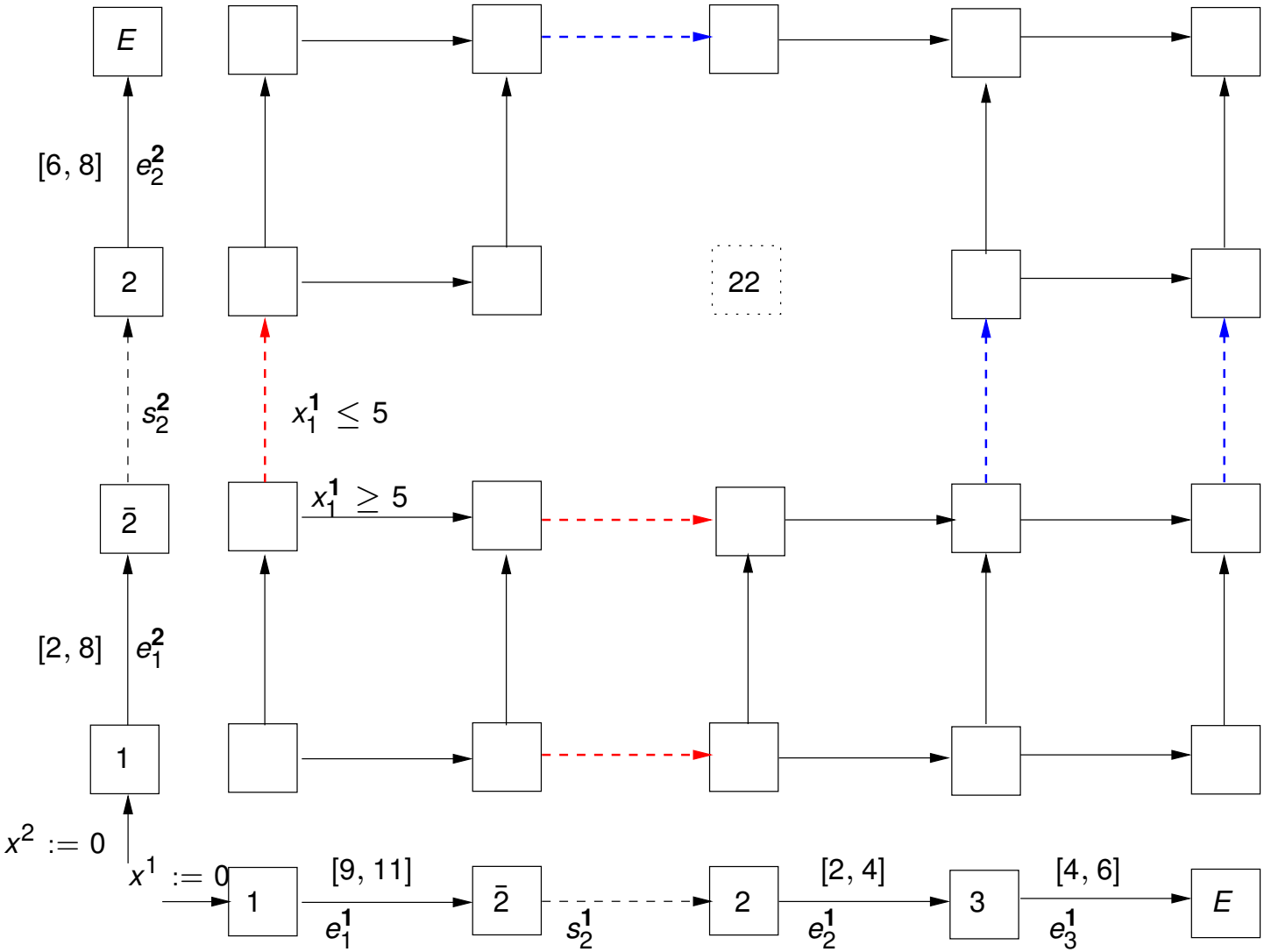
Example II

- Values of waiting and starting



Example III

- The optimal scheduler



Plan

Related work

- Timed controller synthesis (Maler, Pnueli and Sifakis) extended to time optimality (Asarin and M) and cost (Bouyer, Larsen et al)
- Scheduling with timed automata (Abdeddaim, Asarin and M)
- Computation with “non Markovian” distributions (Alur and Bernadsky, Vicario et al)
- Duration probabilistic automata and density transformers (M, Larsen and Krogh 2010)
- Clock-free computation of expected termination time for a **given** scheduler (M, Kempf and Bozga 2011)

Future Work

- Complete the implementation
- Empirical comparison with techniques based on Monte-Carlo simulation that evaluate schedules based on sampling the duration space
- Extending to cyclic non-terminating jobs (which requires new definitions of performance measures)

Thank You