**BUSI 573: Stochastic Models in Operations Management** **Spring 2026**

# Lecture 7: Foster-Lyapunov Techniques & Dynamic Matching Models

*Lecturer: Süleyman Kerimov* *Date: February 24, 2026*

**Disclaimer**: *These notes are not meant to be complete or fully rigorous; some proofs are not given, incomplete, or only outlined, as they are discussed in class.*

We will attempt to study Foster-Lyapunov techniques via dynamic matching models (also known as matching queues). Before we proceed with the dynamic matching model, here is one informal version of the Foster-Lyapunov criteria. Let $X$ be an irreducible discrete-time Markov chain with a countable state space $\mathcal{S}$. Let $\mathbb{P}(x, A) = \mathbb{P}(X(t) \in A | X(t-1) = x)$ be a transition operator. Let $h : S \to \mathbb{R}_{\geq 0}$ be some function. Then we denote the drift of $h(\cdot)$ at $x \in \mathcal{S}$ by

$$\Delta h(x) = \int P(x, dy) h(y) - h(x).$$

It follows that $X$ is positive recurrent if for all $x \in S$, there exists some non-negative function $f$, a finite set $C$, and constants $\delta, b > 0$ such that

$$\Delta h(x) \leq -\epsilon f(x) + b \mathbb{1}_{\{x \in C\}}.$$

This is a powerful result that we can use to establish ergodicity for various stochastic processes. One can further show that $\mathbb{E}_\pi[f(X)] \leq \frac{b}{\epsilon}$. We will be more formal soon.

## 7.1 Two-way matching model

Consider the following two-way matching model. There is a finite set of *agent types* $\mathcal{A} = \{1, 2, \ldots, n\}$, a finite set of *matches* $\mathcal{M} = \{1, \ldots, d\}$, and a *match value* $r_m > 0$ for each match $m \in \mathcal{M}$. Each match $m \in \mathcal{M}$ is characterized by *two* participating agent types, denoted by the set $\mathcal{A}(m)$. The *network topology* is specified by a *matching matrix* $M \in \{0, 1\}^{n \times d}$, where $M_{im} = 1$ if and only if $i \in \mathcal{A}(m)$. There is no harm in assuming that each agent type participates in at least one match. Each agent type $i \in \mathcal{A}$ is associated with an *arrival probability* $\lambda_i > 0$; $\sum_{i \in \mathcal{A}} \lambda_i = 1$. We refer to the tuple $\mathcal{G} = (M, \lambda, r)$ as the *matching network*.

The matching network induces a weighted undirected simple graph, where the set of vertices is $\mathcal{A}$ and the set of edges is $\mathcal{M}$: there is an edge between $i, j \in \mathcal{A}$ with weight $r_m$ if and only if there exists $m \in \mathcal{M}$ such that $\mathcal{A}(m) = \{i, j\}$. We can assume without of loss generality that $\mathcal{G}$ is connected.

**Dynamics.** Time is discrete, and there is a single agent arrival every period. The arriving agent is of type $i \in \mathcal{A}$ with probability $\lambda_i$. We maintain a separate queue for each agent type, and agents join their type-dedicated queues upon arrival. All queues are empty at time $t = 0$.

Match $m \in \mathcal{M}$ is *available* at time $t$ if and only if the queues of both agent types in $\mathcal{A}(m)$ are non-empty at that time. Performing $m \in \mathcal{M}$ once requires one agent from each type in $\mathcal{A}(m)$ and generates a value of $r_m$. Matched agents leave the market immediately.

The process $A_i^t$ counts the number of arrivals to queue $i \in \mathcal{A}$ until (and including) time $t$. The sequence of events in a time period is: an agent arrival is realized, then matches are performed, and queue-lengths are updated. The process $Q_i^t$ tracks the number of agents waiting in queue $i \in \mathcal{A}$ at time $t$, *after* all matches for this period have been performed.

**Matching policy.** A matching *policy* is a mapping from histories of arrivals and performed matches to a (possibly empty) set of matches. Given the history, the matching policy determines how many times each match is performed at each time period. An *admissible* matching policy is an increasing non-anticipative process $D^t := (D_m^t : m \in \mathcal{M}, t \geq 0)$, where $D_m^t$ is the number of times match $m \in \mathcal{M}$ is performed by time $t$; $D^t$ must satisfy

$$Q^t = A^t - MD^t \text{ for all } t \geq 0. \tag{7.1}$$

We assume that $D^t$ is right-continuous with left limits (RCLL). $\Delta D_m^t := D_m^t - D_m^{t-1}$ is then the number of times match $m \in \mathcal{M}$ is performed at time $t > 0$. We add the superscript $D$ on expectations to make explicit the dependence on the policy, where the superscript is omitted when the context is clear. The family of all admissible matching policies is denoted by $\Pi$.

Greedy policies are a large family of admissible policies. These policies perform, whenever possible, a match among those available within a prespecified set. The reason of defining a prespecified set will be clear later.

**Definition 7.1** (greedy policy). *Given a matching network $\mathcal{G}$ and a subset $\mathcal{S} \subseteq \mathcal{M}$ (not necessarily strict), we say that a policy $D$ is a* greedy policy *with respect to $\mathcal{S}$, if*

*(i) a match is performed whenever at least one match becomes available to perform in $\mathcal{S}$, and*

*(ii) matches in $\mathcal{M}\backslash\mathcal{S}$ are never performed, i.e., $D_m^t = 0$ for all $m \in \mathcal{M}\backslash\mathcal{S}$ and for all $t \geq 0$.*

**Optimality criterion.** The expected *total value* generated by time $t$ under a policy $D$ is given by

$$\mathcal{R}^{D,t} := \mathbb{E}^D[r \cdot D^t].$$

For any *fixed* $t$, the *optimal value* $\mathcal{R}^{*,t} := \max_{D\in\Pi} \mathcal{R}^{D,t}$ is trivially attained by the policy, which takes no action until time $t$ and follows an optimal (static) weighted matching at time $t$. That is,

$$\mathcal{R}^{*,t} := \mathbb{E}\left[\begin{array}{ll} \max & r \cdot y \\ \text{s.t.} & My \leq A^t \\ & y \in \mathbb{Z}^d_{\geq 0} \end{array}\right],$$

where the expectation is taken over all realizations of $A^t$.

The function $\mathcal{R}^{*,t}$ can be interpreted as the *hindsight upper bound* at time $t$, i.e., the decision maker is allowed to correct past decisions so that previously performed matches may be revoked to perform new ones at all times. A matching policy is *hindsight optimal* if it is, *at all times*, *almost* as good as the optimal value.

**Definition 7.2** (hindsight optimality). *A matching policy $D$ is* hindsight optimal *if*

$$\mathcal{R}^{*,t} - \mathcal{R}^{D,t} = \mathcal{O}(1) \text{ for all } t > 0,$$

*which implies, in particular, $\mathcal{R}^{D,t}/\mathcal{R}^{*,t} = 1 - \mathcal{O}(1/t)$ for all $t > 0$.*

The existence of a hindsight optimal matching policy means that the tension between short- and long-term objectives is essentially moot; a good performance at time $t_0$ does not necessitate a significant compromise at time $t_1 > t_0$. Observe that a hindsight optimal matching policy is also optimal in the long-run average sense:

$$\frac{\mathcal{R}^{*,T} - \mathcal{R}^{D,T}}{\mathcal{R}^{*,T}} = \mathcal{O}(1/T) \to 0 \text{ as } T \to \infty. \tag{7.2}$$

### 7.1.1 Static planning problem and general position condition

Relaxing the integrality constraints and applying Jensen's inequality gives the following upper bound on $\mathcal{R}^{*,t}$:

$$\mathcal{R}^{*,t} = \mathbb{E}\left[\begin{array}{ll} \max & r \cdot y \\ \text{s.t.} & My \le A^t \\ & y \in \mathbb{Z}^d_{\ge 0} \end{array}\right] \le \begin{array}{ll} \max & r \cdot x \\ \text{s.t.} & Mx \le \lambda t \\ & x \in \mathbb{R}^d_{\ge 0}. \end{array}$$

With the change of variables $z = x/t$, we can write the upper bound in standard form as follows:

$$\begin{array}{ll} \max & r \cdot z \\ \text{s.t.} & Mz + s = \lambda \\ & z \in \mathbb{R}^d_{\ge 0}, s \in \mathbb{R}^n_{\ge 0}. \end{array} \tag{SPP}$$

We refer to this formulation as the *static-planning problem* (SPP). The following definition introduces the notion of *general position* that captures the level of stability in a matching network and plays a crucial role in our main results. In fact, general position is a necessary condition to achieve hindsight optimality (next lecture).

**Definition 7.3** (general position). *A matching network $\mathcal{G}$ satisfies the* general position *condition* (**GP**) *if* (SPP) *has a unique non-degenerate optimal solution $(z^*, s^*)$, i.e., all $n$ basic variables in this solution are strictly positive. Define the sets*

$$\mathcal{M}_+ := \{m \in \mathcal{M} : z^*_m > 0\},\ \mathcal{M}_0 := \mathcal{M} \backslash \mathcal{M}_+,\ \mathcal{Q}_+ := \{j \in \mathcal{A} : s^*_j > 0\} \text{ and } \mathcal{Q}_0 := \mathcal{A} \backslash \mathcal{Q}_+,$$

*where $\mathcal{M}_+$ is the set of* active *matches, $\mathcal{M}_0$ is the set of* redundant *matches, $\mathcal{Q}_+$ is the set of* under-demanded (non-empty) *queues, and $\mathcal{Q}_0$ is the set of* over-demanded (empty) *queues. The* general position gap *is defined as*

$$\epsilon := \min_{m \in \mathcal{M}_+} z^*_m \wedge \min_{j \in \mathcal{Q}_+} s^*_j.$$

**Residual graph.** To achieve hindsight optimality, any matching policy must mostly avoid performing redundant matches. Accordingly, the policies that we will propose are greedy with respect to the set $\mathcal{S} = \mathcal{M}_+ \subsetneq \mathcal{M}$. Let $\mathcal{G}' := \mathcal{G} - \mathcal{M}_0$ be the (SPP)-*residual graph*, which is obtained from $\mathcal{G}$ by removing all redundant matches (every $m \in \mathcal{M}$ with $z^*_m = 0$). The (SPP)-residual graph $\mathcal{G}'$ is then a union of (possibly) multiple components, and we write $\mathcal{G}' = \cup_{k\in[K]}\mathcal{C}_k$, where $\mathcal{C}_k$ is the $k^{th}$ component of $\mathcal{G}'$. Since $\mathcal{G}$ is a simple graph, any edge (match) removal can increase the number of components at most by 1; $K \leq |\mathcal{M}_0|+1$. Let $\mathcal{A}(\mathcal{C}_k)$ be the set of all vertices (queues) in $\mathcal{C}_k$, and let $\mathcal{M}(\mathcal{C}_k)$ be the set of all edges (matches) in $\mathcal{C}_k$ for all $k \in [K]$.

The (SPP)-residual graph $\mathcal{G}'$ has some useful properties, which will be crucial in the design and analysis of our policies.

**Lemma 7.4.** *Assume that $\mathcal{G}$ satisfies* **GP**. *Then each component $\mathcal{C}_k$, $k \in [K]$, of the* (SPP)*–residual graph $\mathcal{G}'$ satisfies the following properties: (i) $\mathcal{C}_k$ contains at most one cycle, (ii) if $\mathcal{C}_k$ does not contain a cycle, then $\mathcal{C}_k$ is a tree and $|\mathcal{A}(\mathcal{C}_k) \cap \mathcal{Q}_+|= 1$, and (iii) if $\mathcal{C}_k$ contains a cycle, then the cycle is of odd length and $|\mathcal{A}(\mathcal{C}_k) \cap \mathcal{Q}_+|= 0$.*

As an important consequence of the general position condition, bounding the all-time regret of a policy can be boiled down to analyzing the total length of the over-demanded queues, provided that the policy is restricted to active matches.

**Lemma 7.5.** *Suppose that $\mathcal{G}$ satisfies the general position condition, and let $(z^*, s^*)$ be a non-degenerate optimal solution of $(\lambda)$. Suppose that the following conditions hold under a policy $D$:*

1. *Only matches in $\mathcal{M}_+$ are performed, and*
2. *$\sum_{i\in\mathcal{Q}_0} \mathbb{E}[Q_i(t)] \leq B$ for every $t > 0$, where $B > 0$ does not depend on $t$.*

*Then, $\mathcal{R}^{*,t} - \mathcal{R}^{D,t} \leq r_{\max} nB$, where $r_{\max} \triangleq \max_{m\in\mathcal{M}_+} r_m$.*

The optimality test lemma should already hint you that Foster-Lyapunov techniques will be very useful to bound stationary expectations of queue-lengths so that we can establish constant regret bounds.

### 7.1.2 Candidate matching policies

**Definition 7.6** (longest-queue policy)**.** *Given a matching network $\mathcal{G}$, the longest-queue policy, denoted by $LQ(\mathcal{M}_+)$, is a greedy policy with respect to $\mathcal{M}_+$ such that*

*(i) At any time $t > 0$, upon arrival of an agent (say type$-i$), perform the available match $m \in \mathcal{M}_+$ such that $A(m) = \{i, j\}$ and $j \in \{Q^t_k : A(m') = \{i, k\}$ for some $m' \in \mathcal{M}_+\}$, where ties are broken arbitrarily, and*

*(ii) at the end of each time period (after a match is performed), all agents of types $i \in \mathcal{Q}_+$ leave the market unmatched.*

**Definition 7.7** (static priority policy)**.** *Given a matching network $\mathcal{G}$, the static priority policy, denoted by $SP(\mathcal{M}_+, p)$, is a greedy policy with respect to $\mathcal{M}_+$ such that*

*(i)* $p : \mathcal{M}_+ \to \{1, \ldots, |\mathcal{M}_+|\}$ *is a bijective static priority order. We say that $m \in \mathcal{M}_+$ has a higher priority than $m' \in \mathcal{M}_+$ if and only if $p(m) < p(m')$,*

*(ii) at any time $t > 0$, upon arrival of an agent (say type$-i$), perform the highest priority match $m \in \mathcal{M}_+$ among those available, where $m \in \{p(m') : i \in A(m')\}$, and*

*(iii) at the end of each time period (after a match is performed), all agents of type$-i$, $i \in \mathcal{Q}_+$, leave the market unmatched.*

**Discussion 7.8.** *What are other candidate matching policies? How do they differ in terms of operational costs?*