Dynamic Scheduling via Polymatroid Optimization

David D. Yao

Columbia University, New York, NY 10027, USA,
yao@ieor.columbia.edu,
http://www.ieor.columbia.edu/∼yao

Abstract. Dynamic scheduling of multi-class jobs in queueing systems has wide ranging applications, but in general is a very difficult control problem. Here we focus on a class of systems for which *conservation laws* hold. Consequently, the performance space becomes a *polymatroid* — a polytope with a matroid-like structure, with all the vertices corresponding to the performance under priority rules, and all the vertices are easily identified. This structure translates the optimal control problem to an optimization problem, which, under a linear objective, becomes a special linear program; and the optimal schedule is a priority rule. In a more general setting, conservation laws extend to so-called generalized conservation laws, under which the performance space becomes more involved; however, the basic structure that ensures the optimality of priority rules remains intact. This tutorial provides an overview to the subject, focusing on the main ideas, basic mathematical facts, and computational implications.

1 Polymatroid

1.1 Equivalent Definitions and Properties

We start with three equivalent definitions of a polymatroid. Definition 1 is the most standard one; Definition 2 will later motivate the definition for EP; Definition 3 provide a contrast against the structure of the EP in §4 (refer to Definition 7).

Throughout, $E = \{1, ..., n\}$ is a finite set; A^c denotes the complement of set A: $A^c = E \setminus A$; and the terms, "increasing" and "decreasing" are used in the non-strict sense, meaning "non-decreasing" and "non-increasing", respectively.

Definition 1. *(Welsh [47], Chapter 18) The following polytope*

$$\mathcal{P}(f) = \{ \ x \geq 0 : \sum_{i \in A} x_i \leq f(A), \ A \subseteq E \ \} \tag{1}$$

is termed a polymatroid if the function $f : 2^E \mapsto \Re_+$ satisfies the following properties:

(i) *(normalized)* $f(\emptyset) = 0$*;*
(ii) *(increasing)* *if* $A \subseteq B \subseteq E$*, then* $f(A) \le f(B)$*;*
(iii) *(submodular)* *if* $A, B \subseteq E$*, then* $f(A) + f(B) \ge f(A \cup B) + f(A \cap B)$*.*

In matroid parlance, a function f that satisfies the above properties is termed a "rank function." Also note that a companion to submodularity is *supermodularity*, defined as when the inequality in (iii) holds in the opposite direction ($\le$).

We now present the second definition for polymatroid. Given a set function $f : 2^E \mapsto \Re_+$, with $f(\emptyset) = 0$, and a permutation π of $\{ 1, 2, \cdots, n \}$, the elements of the set E, we define a vector x^π with the following components (to simplify notation, x_{π_i} below is understood to be $x^\pi_{\pi_i}$):

$$\begin{aligned}
x_{\pi_1} &= f(\{\pi_1\}) \\
x_{\pi_2} &= f(\{\pi_1, \pi_2\}) - x_{\pi_1} = f(\{\pi_1, \pi_2\}) - f(\{\pi_1\}) \\
&\ \vdots \\
x_{\pi_n} &= f(\{\pi_1, \pi_2, \cdots, \pi_n\}) - f(\{\pi_1, \pi_2, \cdots, \pi_{n-1}\})
\end{aligned}$$

x^π is termed a "vertex" of the polytope $\mathcal{P}(f)$ in (1). Note, however, that this terminology could be misleading, since *a priori* there is no guarantee that x^π necessarily belongs to the polytope, since we simply do not know, as yet, whether or not x^π defined as above satisfies the set of inequalities that define $\mathcal{P}(f)$ in (1). In fact, this is the key point in the second definition of polymatroid below.

Definition 2. $\mathcal{P}(f)$ *of (1) is a polymatroid if* $x^\pi \in \mathcal{P}(f)$ *for all permutation* π*.*

Here is a third definition.

Definition 3. $\mathcal{P}(f)$ *of (1) is a polymatroid if for any* $A \subset B \subseteq E$*, there exists a point* $x \in \mathcal{P}(f)$*, such that*

$$\sum_{i \in A} x_i = f(A) \qquad \text{and} \qquad \sum_{i \in B} x_i = f(B).$$

Below we show the three definitions are equivalent.

Theorem 1. *The above three definitions for polymatroid are equivalent.*

Proof. (Definition 1 $\Longrightarrow$ Definition 2)

That $x_{\pi_i} \ge 0$ for all i follows directly from the increasing property of f.

For any $A \subseteq E$ and $\pi_i \in A$, since f is submodular, we have

$$\begin{aligned}
&f(A \cap \{\pi_1, \cdots, \pi_i\}) + f(\{\pi_1, \cdots, \pi_{i-1}\}) \\
&\ge f(A \cap \{\pi_1, \cdots, \pi_{i-1}\}) + f((A \cap \{\pi_1, \cdots, \pi_i\}) \cup \{\pi_1, \cdots, \pi_{i-1}\}) \\
&= f(A \cap \{\pi_1, \cdots, \pi_{i-1}\}) + f(\{\pi_1, \cdots, \pi_i\}),
\end{aligned}$$

which implies

$$\begin{aligned}
&f(\{\pi_1, \cdots, \pi_i\}) - f(\{\pi_1, \cdots, \pi_{i-1}\}) \\
&\le f(A \cap \{\pi_1, \cdots, \pi_i\}) - f(A \cap \{\pi_1, \cdots, \pi_{i-1}\}).
\end{aligned}$$

Summing over $\pi_i \in A$, we have

$$\begin{aligned}\sum_{\pi_i \in A} x_{\pi_i} &= \sum_{\pi_i \in A} (f(\{\pi_1, \cdots, \pi_i\}) - f(\{\pi_1, \cdots, \pi_{i-1}\}) \\ &\le \sum_{\pi_i \in A} f(A \cap \{\pi_1, \cdots, \pi_i\}) - f(A \cap \{\pi_1, \cdots, \pi_{i-1}\}) = f(A).\end{aligned}$$

Hence, $x^\pi \in \mathcal{P}(f)$, and Definition 2 follows.

(Definition 2 $\Longrightarrow$ Definition 3)

For any given $A \subset B \subseteq E$, from Definition 2, it suffices to pick a vertex x^π, such that its first $|A|$ components constitute the set A, and its first $|B|$ components constitute the set B.

(Definition 3 $\Longrightarrow$ Definition 1)

Taking $A = \emptyset$ in Definition 3 yields $f(\emptyset) = 0$. Monotonicity is trivial, since $x_i \ge 0$. For submodularity, take any $A, B \subseteq E$, $A \ne B$; then there exists $x \in \mathcal{P}(f)$ such that

$$\sum_{A \cup B} x_i = f(A \cup B), \quad \text{and} \quad \sum_{A \cap B} x_i = f(A \cap B),$$

since $A \cap B \subset A \cup B$. Therefore,

$$f(A \cup B) + f(A \cap B) = \sum_{A \cup B} x_i + \sum_{A \cap B} x_i = \sum_{i \in A} x_i + \sum_{i \in B} x_i \le f(A) + f(B),$$

where the inequality follows from $x \in \mathcal{P}(f)$.

1.2 Optimization

Here we consider the optimization problem of maximizing a linear function over the polymatroid $\mathcal{P}(f)$.

$$\begin{aligned}\textbf{(P)} \qquad \max \ & \sum_{i \in E} c_i \, x_i \\ \text{s.t.} \ & \sum_{i \in A} x_i \le f(A), \qquad \text{for all } A \subseteq E, \\ & x_i \ge 0, \qquad \text{for all } i \in E.\end{aligned}$$

Assume

$$c_1 \ge c_2 \ge \cdots \ge c_n \ge 0, \tag{2}$$

without loss of generality, since any negative c_i clearly results in the corresponding $x_i = 0$. Let $\pi = (1, 2, \cdots, n)$. Then, we claim that the vertex x^π in Definition 2 is the optimal solution to $(\mathbf{P})$.

To verify the claim, we start with writing down the dual problem as follows:

$$\begin{aligned} \textbf{(D)} \qquad & \min \sum_{A\subseteq E} y_A\, f(A) \\ & \text{s.t.} \sum_{A\ni i} y_A \geq c_i, \qquad \text{for all } i\in E, \\ & \qquad y_A \geq 0, \qquad \text{for all } A\subseteq E. \end{aligned}$$

Define y^π, a candidate dual solution, componentwise as follows:

$$\begin{aligned} y^\pi_{\{1\}} &= c_1 - c_2, \\ y^\pi_{\{1,2\}} &= c_2 - c_3, \\ &\vdots \\ y^\pi_{\{1,\ldots,n-1\}} &= c_{n-1} - c_n, \\ y^\pi_{\{1,\ldots,n\}} &= c_n; \end{aligned}$$

and set $y^\pi_A = 0$, for all other $A \subseteq E$.

Now the claimed optimality follows from

(1) primal feasibility: x^π is a vertex of the polymatroid $\mathcal{P}(f)$, and hence is feasible by definition (refer to Definition 2);
(2) dual feasibility: that y^π is feasible is easily checked (in particular, non-negativity follows from (2));
(3) complementary slackness: also easily checked, in particular, the n binding constraints in **(P)** that define the vertex x^π correspond to the n non-zero (not necessarily zero, to be precise) components of y^π listed above.

It is also easy to verify that the primal and the dual objectives are equal: letting $c_{n+1} := 0$, we have

$$\begin{aligned} \sum_{i\in E} c_i\, x^\pi_i &= \sum_{i\in E} c_i\, [f(\{1,\cdots,i\}) - f(\{1,\cdots,i-1\})] \\ &= \sum_{i=1}^{n} (c_i - c_{i+1}) f(\{1,\cdots,i\}) = \sum_{A\subseteq E} y^\pi_A f(A). \end{aligned}$$

To summarize, x^π is optimal for **(P)** and y^π is optimal for **(D)**. It is important to note that

(a) Primal feasibility is always satisfied, by definition of the polymatroid.
(b) It is the dual feasibility that determines the permutation π, which, by way of complementary slackness, points to a vertex of $\mathcal{P}(f)$ that is optimal.

More specifically, the sum of the dual variables yields the cost coefficients:

$$y^\pi_{\{1,\ldots,i\}} + \cdots + y^\pi_{\{1,\ldots,n\}} = c_i, \qquad i = 1, \ldots, n; \tag{3}$$

the order of which [cf. (2)] decides the permutation π.

2 Conservation Laws

2.1 Polymatroid Structure

To relate to the last section, here $E = \{1, 2, ..., n\}$ denotes the set of all job classes, and x denotes the vector of performance measures of interest. For instance, x_i is the (long-run) average delay or throughput of job class i.

The conservation laws defined below were first formalized in Shanthikumar and Yao [39], where the connection to polymatroid was made. In [39], as well as subsequent papers in the literature, these laws are termed "strong conservation laws." Here, we shall simply refer to these as conservation laws.

Verbally, conservation laws can be summarized into the following two statements:

(i) the total performance (i.e., the sum) over all job classes in E is invariant under any admissible policy;
(ii) the total performance over any given subset, $A \subset E$, of job classes is minimized (or maximized) by offering priority to job classes in this subset over all other classes.

As a simple example, consider a system of two job classes. Each job (of either class) brings a certain amount of "work" (service requirement) to the system. Suppose the server serves (i.e., depletes work) at unit rate. Then it is not difficult to see that (i) the total amount of work, summing over all jobs of both classes that are present in the system, will remain invariant regardless of the actual policy that schedules the server, as long as it is non-idling; and (ii) if class 1 jobs are given preemptive priority over class 2 jobs, then the amount of work in system summing over class 1 jobs is minimized, namely, it cannot be further reduced by any other admissible policy.

We now state the formal definition of conservation laws. For any $A \subseteq E$, denote by $|A|$ the cardinality of A. Let $\mathcal{A}$ denote the space of all *admissible* policies — all non-anticipative and non-idling policies (see more details below), and x^u the performance vector under an admissible policy $u \in \mathcal{A}$. As before, let π denote a permutation of the integers $\{1, 2, ..., n\}$. In particular, $\pi = (\pi_1, ..., \pi_n)$ denotes a priority rule, which is admissible, and in which class π_1 jobs are assigned the highest priority, and class π_n jobs, the lowest priority.

Definition 4. *(Conservation Laws) The performance vector x is said to satisfy conservation laws, if there exists a set function b (or respectively f): $2^E \mapsto \Re_+$, satisfying*

$$b(A) = \sum_{i \in A} x_{\pi_i}, \quad \forall \pi : \{\pi_1, ..., \pi_{|A|}\} = A, \quad \forall A \subseteq E; \tag{4}$$

or respectively,

$$f(A) = \sum_{i \in A} x_{\pi_i}, \quad \forall \pi : \{\pi_1, ..., \pi_{|A|}\} = A, \quad \forall A \subseteq E; \tag{5}$$

(when $A = \emptyset$, by definition, $b(\emptyset) = f(\emptyset) = 0$); such that for all $u \in \mathcal{A}$ the following is satisfied:

$$\sum_{i \in A} x_i^u \geq b(A), \quad \forall A \subset E; \quad \sum_{i \in E} x_i^u = b(E); \tag{6}$$

or respectively,

$$\sum_{i \in A} x_i^u \leq f(A), \quad \forall A \subset E; \quad \sum_{i \in E} x_i^u = f(E). \tag{7}$$

Note that whether the function b or the function f applies in a particular context is determined by whether the performance in question is minimized or maximized by the priority rules. (For instance, b applies to delay, and f applies to throughput.) It is important to note that this minimal (or maximal) performance is required to be independent of the priority assignment among the classes within the subset A on the one hand and the priority assignment among the classes within the subset $E \setminus A$ on the other hand, as long as any class in A has priority over any class in $E \setminus A$. This requirement is reflected in the qualifications imposed on π in defining $b(A)$ and $f(A)$ in (4) and (5). In particular, the definition requires that $b(A)$ and $f(A)$ be respectively, the minimal and the maximal total performance summing over all job classes in the subset A that are given priority over all the other classes.

For the time being, ignore the b part of Definition 4. It is clear that when x satisfies the conservation laws, the performance space, as defined by the polytope in (7), is a polymatroid. This is because following (5) and (7), all the vertices x^π indeed, *by definition*, belong to the polytope. In fact, the polytope in (7) is the polymatroid $\mathcal{P}(f)$ of (1) restricted to the hyperplane $\sum_{i \in E} x_i = f(E)$ (instead of the half-plane $\sum_{i \in E} x_i \leq f(E)$), and is hence termed the *base* of the polymatroid $\mathcal{P}(f)$, denoted $\mathcal{B}(f)$ below. Furthermore, following Theorem 1, we know that when x satisfies conservation laws, the function $f(\cdot)$ as defined in (5) is increasing and submodular.

Next, consider the b part of Definition 4. Note that subtracting the inequality constraint from the equality constraint in (6), we can express these constraints in the same form as in (7), by letting $f(A) := b(E) - b(E \setminus A)$, or equivalently, $b(A) := b(E) - f(E \setminus A)$. Hence, the polytope in (6) is also (the base of) a polymatroid. Furthermore, the increasingness and submodularity of f translate into the increasingness and supermodularity of b.

To sum up the above discussion, we have

Theorem 2. *If the performance vector x satisfies conservation laws, then its feasible space (i.e., the achievable performance region) constitutes the base polytope of a polymatroid, $\mathcal{B}(f)$ or $\mathcal{B}(b)$, of which the vertices correspond to the priority rules. Furthermore, the functions f and b, which are the performance functions corresponding to priority rules, are, respectively, increasing and submodular, and increasing and supermodular.*

2.2 Examples

Consider a queueing system with n different job classes which are denoted by the set E. Let u be the control or scheduling rule that governs the order of service among different classes of jobs. Let $\mathcal{A}$ denote the class of admissible controls, which are required to be non-idling and non-anticipative. That is, no server is allowed to be idle when there are jobs waiting to be served, and the control is only allowed to make use of past history and current state of the system. Neither can an admissible control affect the arrival processes or the service requirements of the jobs. Otherwise we impose no further restrictions on the system. For instance, the arrival processes and the service requirements of the jobs can be arbitrary. Indeed, since the control cannot affect the arrival processes and the service requirements, all the arrival and service data can be viewed as generated *a priori* following any given (joint) distribution and with any given dependence relations. We allow multiple servers, and multiple stages (e.g., tandem queues or networks of queues). We also allow the control to be either preemptive or non-preemptive. (Some restrictions will be imposed on individual systems to be studied below.)

Let x_i^u be a performance measure of class i ($i \in E$) jobs under control u. This need not be a steady-state quantity or an expectation; it can very well be a sample-path realization over a finite time interval, for instance, the delay (sojourn time) of the first m class i jobs, the number of class i jobs in the system at time t, or the number of class i job completions by time t. Let $x^u := (x_i^u)_{i\in E}$ be the performance vector.

For any given permutation $\pi \in \Pi$, let x^π denote the performance vector under a priority scheduling rule that assigns priority to the job classes according to the permutation π, i.e., class π_1 has the highest priority, ..., class π_n has the lowest priority. Clearly any such priority rule belongs to the admissible class.

In all the queueing systems studied below, the service requirements of the jobs are mutually independent, and are also independent of the arrival processes. (One exception to these independence requirements is Example 1 below, where these independence assumptions are not needed.) No independence assumption, however, is required for the arrival processes, which can be arbitrary. When a performance vector satisfies conservation laws, whether its state space is $\mathcal{B}(b)$ (6) or $\mathcal{B}(f)$ (7) depends on whether the performance of a given subset of job classes is minimized or maximized by giving priority to this subset. This is often immediately evident from the context.

Example 1 Consider a $G/G/1$ system that allows preemption. For $i \in E$, let $V_i(t)$ denote the amount of work (processing requirement) in the system at time t due to jobs of class i. (Note that for any given t, $V_i(t)$ is a random quantity, corresponding to some sample realization of the work-load process.) Then it is easily verified that for any t, $x := [V_i(t)]_{i\in E}$ satisfies conservation laws.

Example 2 Continue with the last example. For all $i \in E$, let $N_i(t)$ be the number of class i jobs in the system at time t. When the service times follow exponential distributions, with mean $1/\mu_i$ for class i jobs, we have $\mathsf{E}N_i(t) =$

$\mu_i \mathsf{E} V_i(t)$. Let W_i be the steady-state sojourn time in system for class i jobs. From Little's Law we have $\mathsf{E}W_i = \mathsf{E}N_i/\lambda_i = \mathsf{E}V_i/\rho_i$, where λ_i is the arrival rate of class i jobs, $\rho_i := \lambda_i/\mu_i$, N_i and V_i are the steady-state counterparts of $N_i(t)$ and $V_i(t)$, respectively. Hence, the following x also satisfies conservation laws:

(i) for any given t, $x := [\mathsf{E}N_i(t)/\mu_i]_{i \in E}$;
(ii) $x := [\rho_i \mathsf{E}W_i]_{i \in E}$.

Example 3 In a $G/M/c$ ($c > 1$) system that allows preemption, if all job classes follow the same exponential service-time distribution (with mean $1/\mu$), then it is easy to verify that for any t, $x := [\mathsf{E}N_i(t)]_{i \in E}$ satisfies conservation laws. In this case, $\mathsf{E}V_i(t) = \mathsf{E}N_i(t)/\mu$ and $\mathsf{E}W_i = \mathsf{E}N_i/\lambda_i$. Hence, x defined as follows satisfies conservation laws:

(i) for any given t, $x := [\mathsf{E}N_i(t)]_{i \in E}$, $x := [\mathsf{E}V_i(t)]_{i \in E}$;
(ii) $x := [\lambda_i \mathsf{E}W_i]_{i \in E}$.

(If the control is restricted to be non-preemptive, the results here still hold true. See Example 6 below.)

Example 4 The results in Example 3 still hold when the system is a network of queues, provided all job classes follow the same exponential service-time distribution and the same routing probabilities at each node (service-time distributions and routing probabilities can, however, be node dependent); (external) job arrival processes can be arbitrary and can be different among the classes.

Example 5 Another variation of Example 3 is the queue, $G/M/c/K$, where $K \geq c$ denotes the upper limit on the total number of jobs allowed in the system at any time. In this system, higher priority jobs can preempt lower priority jobs not only in service but also in occupancy. That is, whenever a higher priority job finds (on its arrival) a fully occupied system, a lower priority job within the system (if any) will be removed from the system and its occupancy given to the higher priority job. If there is no lower priority job, then the arrived job is rejected and lost. As in Example 3, all jobs follow the same exponential service-time distribution. Let $R_i(t)$ and $D_i(t)$ ($i \in E$) denote, respectively, the (cumulated) number of rejected/removed class i jobs and the (cumulated) number of class i departures (service completions) up to time t. Then, for any given t, (i) $x := [\mathsf{E}R_i(t)]_{i \in E}$ and (ii) $x := [\mathsf{E}D_i(t)]_{i \in E}$ satisfy conservation laws.

We next turn to considering cases where the admissible controls are restricted to be non-preemptive.

Example 6 Consider the $G/G/c$ system, $c \geq 1$. If all job classes follow the same service-time distribution, then it is easy to see that the scheduling of the servers will not affect the departure epochs of jobs (in a pathwise sense); although it *will* affect the identity (class) of the departing jobs at those epochs. (See Shanthikumar and Sumita [38], §2, for the $G/G/1$ case; the results there also hold true for the $G/G/c$ case.) Hence, for any given t, $x := [N_i(t)]_{i \in E}$ satisfies conservation laws.

Example 7 Comparing the above with Example 3, we know that the results there also hold for non-preemptive controls. However, in contrast to the extension of Example 3 to the network case in Example 4, the above can only be extended to queues in tandem, where overtaking is excluded. Specifically, the result in Example 6 also holds for a series of $G/G/c$ queues in tandem, where at each node all job classes have the same service-time distribution, which, however, can be node dependent. External job arrival processes can be arbitrary and can be different among classes. The number of servers can also be node dependent.

Example 8 With non-preemptive control, there is a special case for the $G/G/1$ system with only two job classes ($n = 2$) which may follow *different* service-time distributions: for any given t, $x := [V_i(t)]_{i\in E}$ satisfies conservation laws.

For steady-steady measures, from standard results in $GI/G/1$ queues (see, e.g., Asmussen [1], Chapter VIII, Proposition 3.4), we have

$$\mathsf{E}V_i = \mu_i^{-1}[\mathsf{E}N_i - \rho_i] + \rho_i\mu_i m_i/2$$

and

$$\mathsf{E}V_i = \rho_i[\mathsf{E}W_i - \mu_i^{-1} + \mu_i m_i/2],$$

where m_i is the second moment of the service time of class i jobs. Hence, following the above, we know that $x = [\mathsf{E}N_i/\mu_i]_{i\in E}$ and $x = [\rho_i\mathsf{E}W_i]_{i\in E}$ also satisfy conservation laws.

Example 9 Two more examples that satisfy conservation laws:

(i) for the $G/G/1$ system with preemption,

$$x := [\int_0^t \exp(-\alpha\tau)V_i(\tau)d\tau]_{i\in E};$$

(ii) for the $G/M/1$ system with preemption,

$$x := [\mathsf{E}\int_0^t \exp(-\alpha\tau)N_i(\tau)d\tau/\mu_i]_{i\in E},$$

where in both (i) and (ii) $\alpha > 0$ is a discount rate, and t is any given time.

Finally, note that in all the above examples, with the exception of Example 5, whenever $[\mathsf{E}N_i(t)]_{i\in E}$ satisfies conservation laws, $[\mathsf{E}D_i(t)]_{i\in E}$ also satisfies conservation laws, since in a no-loss system the number of departures is the difference between the number of arrivals (which is independent of the control) and the number in system.

Evidently, based on the above discussions, the state space of the performance vectors in each of the examples above is a polymatroid.

2.3 Optimal Scheduling

Theorem 3. *Consider the optimal control (scheduling) of n jobs classes in the set E:*

$$\max_{u\in\mathcal{A}}\sum_{i\in E} c_i x_i^u \qquad [\text{or} \quad \min_{u\in\mathcal{A}}\sum_{i\in E} c_i x_i^u\,],$$

where x is a performance measure that satisfies conservation laws, and the cost coefficients c_i ($i \in E$) satisfy, without loss of generality, the ordering in (2). Then, this optimal control problem can be solved by solving the following linear program (LP):

$$\max_{x\in\mathcal{B}(f)}\sum_{i\in E} c_i x_i \qquad [\text{or} \quad \min_{x\in\mathcal{B}(b)}\sum_{i\in E} c_i x_i\,].$$

The optimal solution to this LP is simply the vertex $x^{\pi} \in \mathcal{B}(f)$, with $\pi = (1, ..., n)$ being the permutation corresponding to the decreasing order of the cost coefficients in (2). And the optimal control policy is the corresponding priority rule, which assigns the highest priority to class 1 jobs, and the lowest priority to class n jobs.

Example 10 ($c\mu$-rule) Consider one of the performance vectors in Example 2, $x := [\mathsf{E}(N_i)/\mu_i]_{i\in E}$, where N_i is the number of jobs of class i in the system (or, "inventory") in steady state, and μ_i is the service rate. Suppose our objective is to minimize the total inventory cost,

$$\min\sum_{i\in E} c_i\mathsf{E}(N_i),$$

where c_i is the inventory holding cost rate for class i jobs. We then rewrite this objective as

$$\min\sum_{i\in E} c_i\mu_i x_i.$$

(Note that $(N_i)_{i\in E}$ does not satisfy conservation laws; $(x_i)_{i\in E}$ does.) Then, we know from the above theorem that the optimal policy is a priority rule, with the priorities assigned according to the $c_i\mu_i$ values — the larger the value, the higher the priority. This is what is known as the "$c\mu$-rule". When all jobs have the same cost rate, the priorities follow the μ_i values, i.e., the faster the processing rate (or, the shorter the processing time), the higher the priority, which is the so-called SPT (shortest processing time) rule.

The connection between conservation laws and polymatroid, as specified in Theorem 2, guarantees that any admissible control will yield a performance vector that belongs to the polymatroid. Furthermore, the converse is also true: any performance vector that belongs to the polymatroid can be realized by an admissible control. This is because since $\mathcal{B}(f)$ (or $\mathcal{B}(b)$) is a convex polytope, any vector in the performance space can be expressed as a convex combination of the vertices. Following Caratheodory's theorem (refer to, e.g., Chvátal [8]), any vector in the performance space can be expressed as a convex combination of no more than $n+1$ vertices. In other words, any performance vector can be realized by a control that is a *randomization* of at most $n+1$ priority rules, with the convex combination coefficients being the probabilities for the randomization.

In terms of implementation, however, randomization can be impractical. First, computationally, there is no easy way to derive the randomization coefficients. Second, in order to have an unbiased implementation, randomization will have to be applied at the beginning of each regenerative cycle, e.g., a busy period. In heavy traffic, busy periods could be very long, making implementation extremely difficult, and also creating large variance of the performance.

In fact, one can do better than randomization. It is known (e.g., Federgruen and Groenevelt [16]) that any interior point of the performance space can be realized by a particular dynamic scheduling policy, due originally to Kleinrock [30,31], in which the priority index of each job present in the system grows proportionately to the time it has spent waiting in queue, and the server always serves the job that has the highest index. This scheduling policy is completely specified by the proportionate coefficients associated with the jobs classes, which, in turn, are easily determined by the performance vector (provided it is at the interior of the performance space). In terms of practical implementation, there are several versions of this scheduling policy, refer to [18,19].

3 Generalized Conservation Laws

3.1 Motivation and Definition

Although conservation laws apply to the many examples in the last section, there are other interesting and important problems that do not fall into this category. A primary class of such examples includes systems with feedback, i.e., jobs may come back after service completion. For example, consider the so-called Klimov's problem: a multi-class $M/G/1$ queue in which jobs, after service completion, may return and switch to another class, following a Bernoulli mechanism. Without feedback, we know this is a special case of Example 1, and the work in system, $[V_i(t)]_{i\in E}$, satisfies conservation laws. With feedback, however, the conservation laws as defined in Definition 4, need to be modified.

Specifically, with the possibility of feedback, the work of a particular job class, say class i, should not only include the work associated with class i jobs that are present in the system, it should also take into account the *potential* work that will be generated by feedback jobs, which not only include class i jobs but also all other classes that may feedback to become class i. With this modification, the two intuitive principles of conservation laws listed at the beginning of §2.1 will apply.

To be concrete, let us paraphrase here the simple example at the beginning of §2.1 with two job classes, allowing the additional feature of feedback. As before, suppose the server serves at unit rate. Then it is not difficult to see that (i) the total amount of potential work, summing over both classes, will remain invariant regardless of the actual schedule that the server follows, as long as it is a non-idling schedule; and (ii) if class 1 jobs are given (preemptive) priority over class 2 jobs, then the amount of potential work due to class 1 jobs is minimized, namely, it cannot be further reduced by any other scheduling rule. And the same holds for class 2 jobs, if given priority over class 1 jobs.

Another way to look at this example: Let T be the first time there is no class 1 jobs left in the system. Then, T is minimized by giving class 1 jobs (preemptive) priority over class 2 jobs. In particular, T is no smaller than the potential work of class 1 generated by class 1 jobs (only); T is equal to the latter if and only if class 1 jobs are given priority over class 2 jobs.

Therefore, with this modification, the conservation laws in Definition 4 can be generalized. The net effect, as will be demonstrated in the examples below, is that the variables x_i in Definition 4 will have to be multiplied with different coefficients a_i^A that depend on both the job classes (i) and the subsets (A). In particular, when x_i is, for instance, the average number of jobs of class i, a_i^A denotes the rate of potential work of those classes in set A that is generated by class i jobs.

We now state the formal definition of generalized conservation laws (GCL), using the same notation wherever possible as in Definition 4.

Definition 5. *(Generalized Conservation Laws) The performance vector x is said to satisfy generalized conservation laws (GCL), if there exists a set function b (or respectively f): $2^E \mapsto \Re_+$, and a matrix $(a_i^S)_{i\in E,\, S\subseteq E}$ (which is in general different for b and f, but we will not make this distinction below for notational simplicity) satisfying:*

$$a_i^S > 0,\ i \in S; \quad \text{and} \quad a_i^S = 0,\ i \notin S; \qquad \forall S \subseteq E;$$

such that

$$b(A) = \sum_{i\in A} a_{\pi_i}^A x_{\pi_i}, \quad \forall \pi : \{\pi_1, ..., \pi_{|A|}\} = A, \quad \forall A \subseteq E; \tag{8}$$

or respectively,

$$f(A) = \sum_{i\in A} a_{\pi_i}^A x_{\pi_i}, \quad \forall \pi : \{\pi_1, ..., \pi_{|A|}\} = A, \quad \forall A \subseteq E; \tag{9}$$

such that for all $u \in \mathcal{A}$ the following is satisfied:

$$\sum_{i\in A} a_i^A x_i^u \geq b(A), \quad \forall A \subset E; \quad \sum_{i\in E} a_i^E x_i^u = b(E); \tag{10}$$

or respectively,

$$\sum_{i\in A} a_i^A x_i^u \leq f(A), \quad \forall A \subset E; \quad \sum_{i\in E} a_i^E x_i^u = f(E). \tag{11}$$

It is obvious from the above definition that GCL reduces to the conservation laws if $a_i^A = 1$ for all $i \in A$, and all $A \subseteq E$.

3.2 Examples

Example 11 (Klimov's problem [32]) This concerns the optimal control of a system in which a single server is available to serve n classes of jobs. Class i jobs arrive according to a Poisson process with rate α_i, which is independent of other classes of jobs. The service times for class i jobs are independent and identically distributed with mean μ_i. When the service of a class i job is completed, it either returns to become a class j job, with probability p_{ij}, or leaves the system with probability $1 - \sum_j p_{ij}$. Denote $\alpha = (\alpha_i)_{i\in E}$, $\mu = (\mu_i)_{i\in E}$, and $P = [p_{ij}]_{i,j\in E}$.

Consider the class of non-preemptive policies. The performance measure is

$$x_i^u = \text{long-run average number of class } i \text{ jobs in system under policy } u.$$

The objective is to find the optimal policy that minimizes $\sum_j c_j x_j^u$. Klimov proved that a priority policy is optimal and gave a recursive procedure for obtaining the priority indices.

Tsoucas [42] showed that the performance space of Klimov's problem is the following polytope:

$$\{x \geq 0 : \sum_{i\in S} a_i^S x_i \geq b(S), S \subset E; \sum_{i\in E} a_i^E x_i = b(E)\},$$

where the coefficients are given as $a_i^S = \lambda_i \beta_i^S$, with $\lambda = (\lambda)_{i\in E}$ and $\beta^S = (\beta)_{i\in S}$ obtained as follows:

$$\lambda = (I - P')^{-1}\alpha \quad \text{and} \quad \beta^S = (I - P_{SS})^{-1}\mu_S,$$

where P_{SS} and μ_S are, respectively, the restriction of P and μ to the set S. Note that here, λ_i is the overall arrival rate of class i jobs (including both external arrivals and feedback jobs), β_i^S is the amount of potential work of the classes in S generated by a class i job. (Hence, this potential work is generated at rate α_i in the system.) Summing over $i \in S$ yields the total amount of potential work of the classes in S (generated by the same set of jobs), which is minimized when these jobs are given priority over other classes. This is the basic intuition as to why x satisfies GCL.

Example 12 (Branching bandit process) There are m projects at time 0. They are of K classes, labeled $k = 1, \cdots, K$. Each class k project can be in one of a finite number of states, with E_k denoting the state space. Classifying different project classes or projects of the same class but in different states as different "classes," we denote $E = \cup_k E_k = \{1, \cdots, n\}$ as the set of all project classes. A single server works on the projects one at a time. Each class i project keeps the server busy for a duration of v_i time units. Upon completion, the class i project is replaced by N_{ij} projects of class j. The server then has to decide which next project to serve, following a scheduling rule (control) u. The collection $\{(v_i, N_{ij}), j \in E\}$, follows a general joint distribution, which is independent and identically distributed for all $i \in E$.

Given $S \subseteq E$, the S-descendants of a project of class $i \in S$ refers to all of its immediate descendants that are of classes belonging to S, as well as the immediate descendants of those descendents, and so on. (If a project in S transfers into a class that is not in S, and later transfers back into a class in S, it will not be considered as an S-descendant of the original project.) Given a class i project, the union of the time intervals in which its S-descendants are being served is called an (i, S) period. Let T_i^S denote the length of an (i, S) period. It is the "potential work" of the classes in the set S generated by the class i project. And we use T_m^S to denote the time until the system has completely cleared all classes of projects in S class — under a policy that gives priority to those classes in S over other classes. Note, in particular, that T_m^E represents the length of a busy period.

In the discounted case, the expected reward associated with the control u is $\sum_{i\in E} c_i x_i^u$, where

$$x_i^u = \mathsf{E}_u[\int_0^\infty e^{-\alpha t} I_i^u(t)dt]$$

$\alpha > 0$ is the discount rate and

$$I_i^u(t) = \begin{cases} 1, & \text{if a class } i \text{ project is being served at time } t \\ 0, & \text{otherwise} \end{cases}$$

Bertsimas and Niño-Mora [3] showed that $x^u = (x_i^u)_{i\in E}$, as defined above, satisfy the GCL, with coefficients

$$a_i^S = \frac{\mathsf{E}[\int_0^{T_i^{S^c}} e^{-\alpha t}dt]}{\mathsf{E}[\int_0^{v_i} e^{-\alpha t}dt]}, \quad i \in S \subseteq E,$$

and

$$b(S) = \mathsf{E}\left[\int_0^{T_m^E} e^{-\alpha t}dt\right] - \mathsf{E}\left[\int_0^{T_m^{S^c}} e^{-\alpha t}dt\right].$$

Intuitively, the GCL here says that the time until all the S^c-descendents of all the projects in S are served is minimized by giving project classes in S^c priority over those in S.

An undiscounted version is also available in [3]. (This includes Klimov's problem, the last example above, as a special case.) The criterion here is to minimize the total expected cost incurred under control u during the first busy period (of the server) $[0, T]$, $\sum_{i\in E} c_i x_i^u$, with

$$x_i^u = \mathsf{E}_u\left[\int_0^\infty t I_i^u(t)dt\right].$$

Following [3], x^u satisfy GCL with coefficients

$$a_i^S = \mathsf{E}[T_i^{S^c}]/\mathsf{E}[v_i], \quad i \in S \subseteq E,$$

and

$$b(S) = \frac{1}{2}\mathsf{E}[(T_m^E)^2] - \frac{1}{2}\mathsf{E}[(T_m^{S^c})^2] + \sum_{i\in S} b_i(S),$$

where

$$b_i(S) = \frac{\mathsf{E}[v_i]\mathsf{E}[v_i^2]}{2}\left(\frac{\mathsf{E}[T_i^{S^c}]}{\mathsf{E}[v_i]} - \frac{\mathsf{E}[(T_i^{S^c})^2]}{\mathsf{E}[v_i^2]}\right), \quad i \in S.$$

The intuition is similar to the discounted case.

4 Extended Polymatroid

4.1 Equivalent Definitions

Recall the space of any performance measure that satisfies conservation laws is a polymatroid. Analogously, one can ask what is the structure of the performance space under GCL, i.e., what is the structure of the following polytopes:

$$\mathcal{EP}(b) = \{\ x \ge 0 : \sum_{i\in S} a_i^S x_i \ge b(S),\ S \subseteq E\ \}, \tag{12}$$

$$\mathcal{EP}(f) = \{\ x \ge 0 : \sum_{i\in S} a_i^S x_i \le f(S),\ S \subseteq E\ \}. \tag{13}$$

The most natural route to approach this issue appears to be mimicking Definition 2 of polymatroid (and this is indeed the route taken in [3]). Similar to the definition of x^π preceding Definition 2, here, given a permutation π (of $\{\ 1, 2, \cdots, n\ \}$), we can generate a vertex x^π as follows.

$$\begin{aligned}
x_{\pi_1} &= f(\{\pi_1\})/a_{\pi_1}^{\{\pi_1\}} \\
x_{\pi_2} &= \Big(f(\{\pi_1,\pi_2\}) - a_{\pi_1}^{\{\pi_1,\pi_2\}} x_{\pi_1}\Big)\Big/ a_{\pi_2}^{\{\pi_1,\pi_2\}} \\
&\ \ \vdots \\
x_{\pi_n} &= \left(f(\{\pi_1,\cdots,\pi_n\}) - \sum_{i=1}^{n-1} a_{\pi_i}^{\{\pi_1,\cdots,\pi_n\}} x_{\pi_i}\right)\Bigg/ a_{\pi_n}^{\{\pi_1,\cdots,\pi_n\}}.
\end{aligned}$$

Same as in the polymatroid case, we should emphasize here that as yet, x^π does not necessarily belong to the polytope in (13). The vertices for $\mathcal{EP}(b)$ are analogously generated, with $f(\cdot)$ replaced by $b(\cdot)$.

Definition 6. *$\mathcal{EP}(f)$ (respectively $\mathcal{EP}(b)$) is an extended polymatroid (EP) if x^π as generated above (respectively with b replacing f) belongs to the polytope $\mathcal{EP}(f)$, (respectively $\mathcal{EP}(b)$), for any permutation π.*

(The term, "extended polymatroid," was previously used to refer to a polymatroid without the requirement that $x \ge 0$; e.g., see [26], p. 306. Since [3, 42] and other works in the queueing literature, it has been used to refer to the polytopes defined above. Also, in [3], the EP corresponding to the b function is termed "extended contra-polymatroid," with the term "extended polymatroid" reserved for the f function. For simplicity, we do not make such a distinction here and below.)

With the above definition for EP, the right hand side functions b and f are not necessarily increasing and supermodular/submodular. In other words, we do not have a counterpart of Definition 1 for EP (more on this later). On the other hand, the counterpart for Definition 3 does apply.

Definition 7. *$\mathcal{EP}(f)$ is an extended polymatroid if the following is satisfied: for any $A \subset B \subset E$, there exists a point $x \in \mathcal{EP}(f)$, such that*

$$\sum_{i\in A} a_i^A x_i = f(A) \quad \text{and} \quad \sum_{i\in B} a_i^B x_i = f(B).$$

Theorem 4. *The two definitions of EP in 6 and 7 are equivalent.*

Proof. If $\mathcal{EP}(f)$ is EP, then the stated condition in Definition 7 is obviously satisfied: just pick the vertex x^π such that the first $|A|$ components in π constitute the set A, and the first $|B|$ components constitute the set B.

For the other direction, i.e., if the stated condition in Definition 7 holds, then $\mathcal{EP}(f)$ is EP, we use induction on $n = |E|$. That this holds for $n = 1$ is trivial.

Suppose this holds for $n = k$, i.e. for a polytope of the kind in (13) with k variables. Now consider such a polytope with $k + 1$ variables, i.e., $|E| = k + 1$. Without loss of generality, consider the permutation $\pi = (1, 2, ..., k + 1)$. We want to show that the corresponding x^π (i.e., generated from the triangulation above)) is in the polytope $\mathcal{EP}(f)$.

Since $x_1^\pi = f(\{1\})/a_1^{\{1\}}$, we substitute it into the other x_i^π expressions, $i \neq 1$, to arrive at the following polytope of k variables:

$$\mathcal{EP}(\tilde{f}) = \{x \geq 0 : \sum_{i\in S, i\neq 1} a_i^S x_i \leq \tilde{f}(S),\ \{1\} \in S \subseteq E\},$$

where

$$\tilde{f}(S) := f(S) - \frac{f(\{1\})}{a_1^{\{1\}}} a_1^S.$$

Clearly, since the stated condition in Definition 7 is assumed to hold for $\mathcal{EP}(f)$ (the one with $k+1$ variables), it also holds for $\mathcal{EP}(\tilde{f})$ (the one with k variables), since the equations in question all differ by an amount $f(\{1\})a_1^{\{1\}}/a_1^S$ on both sides. Hence, the induction hypothesis confirms that $\mathcal{EP}(\tilde{f})$ is an EP. This implies that $(x_2^\pi, ..., x_n^\pi) \in \mathcal{EP}(\tilde{f})$, which is equivalent to $x^\pi = (x_1^\pi, x_2^\pi, ..., x_n^\pi)$ satisfying all the constraints in $\mathcal{EP}(f)$ that involve $S \subseteq E$ with $1 \in S$.

We still need to check that x^π satisfies all the other constraints in $\mathcal{EP}(f)$ corresponding to $S \subseteq E$ with $1 \notin S$. To this end, consider the following polytope:

$$\{x \geq 0 : \sum_{i\in S} a_i^S x_i \leq f(S),\ S \subseteq E \setminus \{1\}\}. \tag{14}$$

The above is another polytope with k variables. Obviously the stated condition in Definition 7, which is assumed to hold for the polytope $\mathcal{EP}(f)$, holds for the above polytope as well (since the defining inequalities in the latter are just part of those in $\mathcal{EP}(f)$). Hence, based on the induction hypothesis, the polytope in (14) is also an EP. This implies that $(x_2^\pi, ..., x_n^\pi)$, and hence x^π, satisfies all the inequalities involved in (14).

Hence, we have established that given the stated condition in Definition 7, x^π does satisfy all the constraints in $\mathcal{EP}(f)$, for each permutation π. Therefore, $\mathcal{EP}(f)$ is an EP.

The above theorem leads immediately to the following:

Corollary 1. *If $\mathcal{EP}(f)$ is an extended polymatroid, then*

$$\mathcal{EP}^-(f) := \{x \geq 0 : \sum_{i \in S} a_i^S x_i \leq f(S),\, S \subseteq E \setminus E_0\}$$

is also an extended polymatroid, for any $E_0 \subset E$.

Proof. Simply verify Definition 7. Since $\mathcal{EP}(f)$ is an EP, we can pick any $A \subset B \subseteq E \setminus E_0 \subset E$, and there exists an $x \in \mathcal{EP}(f)$, such that $\sum_{i \in A} a_i^A x_i = f(A)$ and $\sum_{i \in B} a_i^B x_i = f(B)$. But this is exactly what is required for $\mathcal{EP}^-(f)$ to be EP.

In summary, we have

Theorem 5. *If the performance vector x satisfies GCL, then the performance polytope is an EP, of which the vertices correspond to the performance under priority rules, and the functions $b(A)$ and $f(A)$ correspond to the performance of job classes in set A when A is given priority over all other classes in $E \setminus A$.*

5 Optimization over EP

Here we consider the optimization problem of maximizing a linear function over the EP, $\mathcal{EP}(f)$, defined in (13):

$$\textbf{(PG)} \qquad \begin{aligned} \max \; & \sum_{i \in E} c_i \, x_i \\ \text{s.t.} \; & \sum_{i \in A} a_i^A x_i \leq f(A), \quad \text{for all}\, A \subseteq E, \\ & x_i \geq 0, \quad \text{for all}\, i \in E. \end{aligned}$$

The dual problem can be written as follows:

$$\textbf{(DG)} \qquad \begin{aligned} \min \; & \sum_{A \subseteq E} y_A \, f(A) \\ \text{s.t.} \; & \sum_{A \ni i} y_A a_i^A \geq c_i, \quad \text{for all} \in E, \\ & y_A \geq 0, \quad \text{for all}\, A \subseteq E. \end{aligned}$$

Let us start with $\pi = (1, 2, \cdots, n)$, and consider x^π, the vertex defined at the beginning part of the last section. Below we write out the objective function of **(PG)** at x^π, and use the expression, along with complementary slackness, to identify a candidate for the dual solution. From dual feasibility, we then identify the conditions under which π is the optimal permutation. Collectively, these steps constitute an algorithm that finds the optimal π.

For simplicity, write x for x^π below. We first write out x_n in the objective function:

$$\sum_{i=1}^{n} c_i x_i = c_n \left(f(\{1, \cdots, n\}) - \sum_{i=1}^{n-1} a_i^{\{1,\cdots,n\}} x_i \right) \Big/ a_n^{\{1,\cdots,n\}} + \sum_{i=1}^{n-1} c_i x_i$$
$$= y_{\{1,\cdots,n\}} f(\{1, \cdots, n\}) + \sum_{i=1}^{n-1} \left(c_i - y_{\{1,\cdots,n\}} a_i^{\{1,\cdots,n\}} \right) x_i,$$

where we set

$$y_{\{1,\cdots,n\}} = c_n / a_n^{\{1,\cdots,n\}}.$$

Next, we write out x_{n-1} in the summation above, and set

$$y_{\{1,\cdots,n-1\}} = (c_{n-1} - y_{\{1,\cdots,n\}} a_{n-1}^{\{1,\cdots,n\}}) / a_{n-1}^{\{1,\cdots,n-1\}},$$

to reach the following expression:

$$\sum_{i=1}^{n} c_i x_i = y_{\{1,\cdots,n\}} f(\{1, \cdots, n\}) + y_{\{1,\cdots,n-1\}} f(\{1, \cdots, n-1\})$$
$$+ \sum_{i=1}^{n-2} \left(c_i - y_{\{1,\cdots,n\}} a_i^{\{1,\cdots,n\}} - y_{\{1,\cdots,n-1\}} a_i^{\{1,\cdots,n-1\}} \right) x_i^\pi.$$

This procedure can be repeated to yield the following:

$$\sum_{i=1}^{n} c_i x_i = y_{\{1,\cdots,n\}} f(\{1, \cdots, n\}) + y_{\{1,\cdots,n-1\}}) f(\{1, \cdots, n-1\})$$
$$+ \cdots + y_{\{1,2\}} f(\{1.2\}) + y_{\{1\}} f(\{1\}), \tag{15}$$

where

$$y_{\{1,\cdots,k\}} = \left(c_k - \sum_{j=k+1}^{n} y_{\{1,\cdots,j\}} a_k^{\{1,\cdots,j\}} \right) \Big/ a_k^{\{1,\cdots,k\}}, \tag{16}$$

for $k = 1, ..., n$. (When $k = n$, the vacuous summation in (16) vanishes.) Furthermore, set $y_A := 0$ for all other $A \subseteq E$.

With the above choice of x and y, it is easy to check that complementary slackness is satisfied. Also, primal feasibility is automatic — guaranteed by the definition of EP, since x is a vertex. Hence, we only need to check dual feasibility.

From the construction of y in (16), we have

$$\sum_{j=i}^{n} y_{\{1,\cdots,j\}} a_i^{\{1,\cdots,j\}} = c_i, \qquad i \in E,$$

satisfying the first set of constraints in **(DG)**. So it suffices to show that the n non-zero dual variables in (16) are non-negative. To this end, we need to be specific about the construction of the permutation $\pi = (1, ..., n)$.

Let us start from the last element in π. Note that from (16), we have

$$y_{\{1,\cdots,n\}} = \frac{c_n}{a_n^{\{1,\cdots,n\}}} \geq 0.$$

Next, to ensure $y_{\{1,\cdots,n-1\}} \geq 0$, the numerator of its expression in (16) must be non-negative, i.e.,

$$\frac{c_{n-1}}{a_{n-1}^{\{1,\cdots,n\}}} \geq y_{\{1,\cdots,n\}} = \frac{c_n}{a_n^{\{1,\cdots,n\}}}.$$

Therefore, the index n has to be:

$$n = \arg\min_i \frac{c_i}{a_i^{\{1,\cdots,n\}}}.$$

Note that this choice of n guarantees $y_{\{1,\cdots,n-1\}} \geq 0$, independent of the ordering of the other $n-1$ elements in the permutation.

Similarly, to ensure $y_{\{1,\cdots,n-2\}} \geq 0$, from (16), we must have

$$c_{n-2} - y_{\{1,\cdots,n-1\}} a_{n-2}^{\{1,\cdots,n-1\}} - y_{\{1,\cdots,n\}} a_{n-2}^{\{1,\cdots,n\}} \geq 0,$$

or

$$\frac{c_{n-2} - y_{\{1,\cdots,n\}} a_{n-2}^{\{1,\cdots,n\}}}{a_{n-2}^{\{1,\cdots,n-1\}}} \geq y_{\{1,\cdots,n-1\}}.$$

Hence, the choice of $n-1$ has to be:

$$n-1 = \arg\min_{i \leq n-1} \frac{c_i - y_{\{1,\cdots,n\}} a_i^{\{1,\cdots,n\}}}{a_i^{\{1,\cdots,n-1\}}}.$$

This procedure can be repeated until all elements of the permutation is determined. In general, the index k is chosen in the order of $k = n, n-1, ..., 1$, and it has to satisfy:

$$k = \arg\min_{i \leq k} \frac{c_i - \sum_{j=k+1}^{n} y_{\{1,\cdots,j\}} a_i^{\{1,\cdots,j\}}}{a_i^{\{1,\cdots,k\}}}.$$

Formally, the following algorithm solves the dual problem **(DG)** in terms of generating the permutation π, along with the dual solution y^π. The optimal primal solution is then the vertex, x^π, corresponding to the permutation π.

Algorithm 1 [for **(DG)**]

```
(i)   Initialization: $S(n) = E$, $k = n$;
(ii)  If $k = 1$, stop, and output $\{\pi, S(k);\ y^\pi(S(k)\}$; else, set
          $$\pi_k := \arg\min_i \frac{c_i - \sum_{j=k+1}^n y^\pi_{S(j)} a_i^{S(j)}}{a_i^{S(k)}}$$
          $$y^\pi_{S(k)} := \min_i \frac{c_i - \sum_{j=k+1}^n y^\pi_{S(j)} a_i^{S(j)}}{a_i^{S(k)}};$$
(iii) $k \leftarrow k - 1$, $S(k) = S(k+1) \setminus \{\pi_k\}$; goto (ii).
```

Theorem 6. *Given an extended polymatroid $\mathcal{EP}(f)$, the above algorithm solves the primal and dual LP's,* **(PG)** *and* **(DG)** *in $O(n^2)$ steps, with x^π and y^π being the optimal primal-dual solution pair.*

Proof. Following the discussions preceding the algorithm, it is clear that we only need to check $y^\pi_{S(k)} \geq 0$, for $k = 1, \cdots, n$.

When $k = n$, following the algorithm, we have $S(n) = E$, and

$$\pi_n = \arg\min_i \{c_i / a_i^E\}, \qquad y^\pi_E = c_{\pi_n} / a^E_{\pi_n} \geq 0.$$

Inductively, suppose $y^\pi_{S(j)} \geq 0$, for $j = k+1, ..., n$, have all been determined. The choice of π_{k+1} and hence $y^\pi_{S(k+1)}$ in the algorithm guarantees

$$c_k - \sum_{j=k}^n y^\pi_{S(j)} a_k^{S(j)} \geq 0,$$

and hence $y^\pi_{S(k)} \geq 0$.

That the optimal solution is generated in O(n^2) steps is evident from the description of the algorithm.

To summarize, the two remarks at the end of §1.2 for the polymatroid optimization also apply here: (i) primal feasibility is automatic, by way of the definition of EP; and (ii) dual feasibility, along with complementary slackness, identifies the permutation π that defines the (primal) optimal vertex.

Furthermore, there is also an analogy to (3), i.e., the sum of dual variables yields the priority index. To see this, for concreteness consider Klimov's problem, with the performance measure x_i being the (long-run) average number of class i jobs in the system. (For this example, we are dealing with a minimization problem over the EP $\mathcal{EP}(b)$. But all of the above discussions, including the algorithm, still apply, mutatis mutandis, such as changing f to b and max to min, etc.) The optimal policy is a priority rule corresponding to the permutation π generated by the above algorithm, with the jobs of class π_1 given the highest priority, and jobs of class π_n, the lowest priority. Let y^* be the optimal dual solution generated by the algorithm. Define

$$\gamma_i := \sum_{S \ni i} y^*_S, \qquad i \in E.$$

Then, we have

$$\gamma_{\pi_i} = y^*_{\{\pi_1,\cdots,\pi_i\}} + \cdots + y^*_{\{\pi_1,\cdots,\pi_n\}}, \quad i \in E. \tag{17}$$

Note that γ_{π_i} is decreasing in i, since the dual variables are non-negative. Hence, the order of γ_{π_i}'s is in the same direction as the priority assignment. In other words, (17) is completely analogous to (3): just like the indexing role played by the cost coefficients in the polymatroid case, in the EP case here $\{\gamma_i\}$ is also a set of indices upon which the priorities are assigned: at each decision epoch, the server chooses to serve, among all waiting jobs, the job class with the highest γ index.

Finally, we can synthesize all the above discussions on GCL and its connection to EP, and on optimization over an EP, to come up with the following generalization of Theorem 3.

Theorem 7. *Consider the optimal control problem in Theorem 3:*

$$\max_{u \in \mathcal{A}} \sum_{i \in E} c_i x_i^u \qquad [\text{or} \quad \min_{u \in \mathcal{A}} \sum_{i \in E} c_i x_i^u\,].$$

Suppose x is a performance measure that satisfies GCL. Then, this optimal control problem can be solved by solving the following LP:

$$\max_{x \in \mathcal{EP}(f)} \sum_{i \in E} c_i x_i \qquad [\text{or} \quad \min_{x \in \mathcal{EP}(b)} \sum_{i \in E} c_i x_i\,].$$

The optimal solution to this LP is simply the vertex $x^\pi \in \mathcal{B}(f)$, with π being the permutation identified by Algorithm 1; and the optimal policy is the corresponding priority rule, which assigns the highest priority to class π_1 jobs, and the lowest priority to class π_n jobs.

Applying the above theorem to Klimov's model we can generate the optimal policy, which is a priority rule dictated by the permutation π, which, in turn, is generated by Algorithm 1.

6 Notes and Comments

The materials presented here are drawn from Chapter 11 of the book by Chen and Yao [7], to which the reader is also referred for preliminaries in queueing networks. A standard reference to matroid, as well as polymatroid, is Welsh [47]. The equivalence of the first two definitions of the polymatroid, Definitions 1 and

2, is a classical result; refer to, e.g., Edmonds [13], Welsh [47], and Dunstan and Welsh [12].

The original version of conservation laws, due to Kleinrock [31], takes the form of a single equality constraint, $\sum_{i \in E} x_i = b(E)$ or $= f(E)$. In the works of Coffman and Mitrani [9], and Gelenbe and Mitrani [20], the additional inequality constraints were introduced, which, along with the equality constraint, give a full characterization of the performance space. In a sequence of papers, Federgruen and Groenevelt [15,16,17], established the polymatroid structure of the performance space of several queueing systems, by showing that the RHS (right hand side) functions are increasing and submodular.

Shanthikumar and Yao [39] revealed the *equivalence* between conservations laws and the polymatroid nature of the performance polytope. In other words, the increasingness and submodularity of the RHS functions are not only sufficient but also necessary conditions for conservation laws. This equivalence is based on two key ingredients: On the one hand, the polymatroid Definition 2 asserts that if the "vertex" x^{π} — generated through a triangular system of n linear equations (made out of a total of $2^n - 1$ inequalities that define the polytope) — belongs to the polytope (i.e., if it satisfies all the other inequalities), for every permutation, π, then the polytope is a polymatroid. On the other hand, in conservation laws the RHS functions that characterize the performance polytope can be defined in such a way that they correspond to those "vertices". This way, the vertices will automatically belong to the performance space, since they are achievable by priority rules.

The direct implication of the connection between conservation laws and polymatroid is the translation of the scheduling (control) problem into an optimization problem. In the case of a linear objective, the optimal solution follows immediately from examining the primal-dual pair: primal feasibility is guaranteed by the polymatroid property — all vertices belong to the polytope, and dual feasibility, along with complementary slackness, yields the priority indices.

Motivated by Klimov's problem, Tsoucas [42], and Bertsimas and Ninõ-Mora [3] extended conservation laws and related polymatroid structure to GCL and EP. The key ingredients in the conservation laws/polymatroid theory of [39] are carried over to GCL/EP. In particular, EP is defined completely analogous to the polymatroid Definition 2 mentioned above, via the "vertex" x^{π}; whereas GCL is such that for every permutation π, x^{π} corresponds to a priority rule, and thereby guarantees its membership to the performance polytope. The equivalent definitions for EP in Definition 7 are due to Lu [34] and Zhang [52] (also see [51]).

Dynamic scheduling of a multi-class stochastic network is a complex and difficult problem that has continued to attract much research effort. A sample of more recent works shows a variety of different approaches to the problem, from Markov decision programming (e.g., Harrison [27], Weber and Stidham [45]), monotone control of generalized semi-Markov processes (Glasserman and Yao [24,25]), to asymptotic techniques via diffusion limits (Harrison [28], and Harrison and Wein [29]). This chapter presents yet another approach, which is based on polymatroid optimization. It exploits, in the presence of conservation laws and GCL, the polymatroid or EP structure of the performance polytope and turns the dynamic control problem into a static optimization problem.

The $c\mu$-rule in Example 10 is a subject with a long history that can be traced back to Smith [40], and the monograph of Cox and Smith [10]; also see, e.g., [5,6]. More ambitious examples of applications that are based on Theorem 3 include: scheduling in a Jackson network ([36]), scheduling and load balancing in a distributed computer system ([37]), and scheduling multi-class jobs in a flexible manufacturing system ([50]).

Klimov's problem generalizes the $c\mu$-rule model by allowing completed jobs to feedback and change classes. Variations of Klimov's model have also been widely studied using different techniques; e.g., Harrison [27], Tcha and Pliska [41]. The optimal priority policy is often referred to as the "Gittins index" rule, as the priority indices are closely related to those indices in dynamic resource allocation problems that are made famous by Gittins ([21,22,23]).

Klimov's model, in turn, belongs to the more general class of branching bandit problems, (refer to §3), for which scheduling rules based on Gittins indices are optimal. There is a vast literature on this subject; refer to, e.g., Lai and Ying [33], Meilijson and Weiss [35], Varaiya *et al.* [43], Weber [44], Weiss [46], Whittle [48,49]; as well as Gittins [21,22], and Gittins and Jones [23].

GCL corresponds to the so-called "indexable" class of stochastic systems, including Klimov's model and branching bandits as primary examples; refer to [3,4]. Beyond this indexable class, however, the performance space is not even an EP. There have been recent studies that try to bound such performance space by more structured polytopes (e.g., polymatroid and EP), e.g., Bertsimas [2], Bertsimas *et al* [4], and Dacre *et al* [11].

References

1. Asmussen, S., *Applied Probability and Queues.* Wiley, Chichester, U.K., 1987.
2. Bertsimas, D., The Achievable Region Method in the Optimal Control of Queueing Systems; Formulations, Bounds and Policies. *Queueing Systems,* **21** (1995), 337–389.
3. Bertsimas, D. and Niño-Mora, J., Conservation Laws, Extended Polymatroid and Multi-Armed Bandit Problems: A Unified Approach to Indexable Systems. *Mathematics of Operations Research*, **21** (1996), 257–306.
4. Bertsimas, D. Paschalidis, I.C. and Tsitsiklis, J.N., Optimization of Multiclass Queueing Networks: Polyhedral and Nonlinear Characterization of Achievable Performance. *Ann. Appl. Prob.*, **4** (1994), 43–75.
5. Baras, J.S., Dorsey, A.J. and Makowski, A.M., Two Competing Queues with Linear Cost: the μc Rule Is Often Optimal. *Adv. Appl. Prob.,* **17** (1985), 186–209.
6. Buyukkoc, C., Varaiya, P. and Walrand, J., The $c\mu$ Rule Revisited. *Adv. Appl. Prob.,* **30** (1985), 237–238.
7. Chen, H. and Yao, D.D., *Fundamentals of Queueing Networks: Performance, Asymptotics and Optimization.* Springer-Verlag, New York, 2001.
8. Chvátal, V., *Linear Programming.* W.H. Freeman, New York, 1983.

9. Coffman, E. and Mitrani, I., A Characterization of Waiting Time Performance Realizable by Single Server Queues. *Operations Research,* **28** (1980), 810–821.
10. Cox, D.R. and Smith, W.L., *Queues.* Methunen, London, 1961.
11. Dacre, K.D., Glazebrook, K.D., and Ninõ-Mora, J., The Achievable Region Approach to the Optimal Control of Stochastic Systems. *J. Royal Statist. Soc.* (1999).
12. Dunstan, F.D.J. and Welsh, D.J.A., A Greedy Algorithm for Solving a Certain Class of Linear Programmes. *Math. Programming,* **5** (1973), 338–353.
13. Edmonds, J., Submodular Functions, Matroids and Certain Polyhedra. *Proc. Int. Conf. on Combinatorics (Calgary)*, Gordon and Breach, New York, 69-87, 1970.
14. Federgruen, A. and Groenevelt, H., The Greedy Procedure for Resource Allocation Problems: Necessary and Sufficient Conditions for Optimality. *Operations Res.,* **34** (1986), 909–918.
15. Federgruen, A. and Groenevelt, H., The Impact of the Composition of the Customer Base in General Queueing Models. *J. Appl. Prob.,* **24** (1987), 709–724.
16. Federgruen, A. and Groenevelt, H., M/G/c Queueing Systems with Multiple Customer Classes: Characterization and Control of Achievable Performance under Non-Preemptive Priority Rules. *Management Science,* **34** (1988), 1121–1138.
17. Federgruen, A. and Groenevelt, H., Characterization and Optimization of Achievable Performance in Queueing Systems. *Operations Res.,* **36** (1988), 733–741.
18. Fong, L.L. and Squillante, M.S., Time-Function Scheduling: A General Approach to Controllable Resource Management. IBM Research Report RC-20155, IBM Research Division, T.J. Watson Research Center, Yorktown Hts., New York, NY 10598, 1995.
19. Franaszek, P.A. and Nelson, R.D., Properties of Delay Cost Scheduling in Timesharing Systems. IBM Research Report RC-13777, IBM Research Division, T.J. Watson Research Center, Yorktown Hts., New York, NY 10598, 1990.
20. Gelenbe, E. and Mitrani, I., *Analysis and Synthesis of Computer Systems.* Academic Press, London, 1980.
21. Gittins, J.C., Bandit Processes and Dynamic Allocation Indices (with discussions). *J. Royal Statistical Society, Ser. B*, **41** (1979), 148–177.
22. Gittins, J.C., *Multiarmed Bandit Allocation Indices.* Wiley, Chichester, 1989.
23. Gittins, J.C. and Jones, D.M., A Dynamic Allocation Index for the Sequential Design of Experiments. In: *Progress in Statistics: European Meeting of Statisticians, Budapest, 1972,* J. Gani, K. Sarkadi and I. Vince (eds.), North-Holland, Amsterdam, 1974, 241–266.
24. Glasserman, P. and Yao, D.D., *Monotone Structure in Discrete-Event Systems.* Wiley, New York, 1994.
25. Glasserman, P. and Yao, D.D., Monotone Optimal Control of Permutable GSMP's. *Mathematics of Operations Research*, **19** (1994), 449–476.
26. Grötschel, M., Lovász, L and Schrijver, A., *Geometric Algorithms and Combinatorial Optimization*, second corrected edition. Springer-Verlag, Berlin, 1993.
27. Harrison, J.M., Dynamic Scheduling of a Multiclass Queue: Discount Optimality. *Operations Res.,* **23** (1975), 270–282.
28. Harrison, J.M., The BIGSTEP Approach to Flow Management in Stochastic Processing Networks. In: *Stochastic Networks: Theory and Applications,* Kelly, Zachary, and Ziedens (eds.), Royal Statistical Society Lecture Note Series, #4, 1996, 57–90.
29. Harrison, J.M. and Wein, L., Scheduling Networks of Queues: Heavy Traffic Analysis of a Simple Open Network. *Queueing Systems,* **5** (1989), 265–280.
30. Kleinrock, L., A Delay Dependent Queue Discipline. *Naval Research Logistics Quarterly,* **11** (1964), 329–341.

31. Kleinrock, L., *Queueing Systems,* Vol. 2. Wiley, New York, 1976.
32. Klimov, G.P., Time Sharing Service Systems, *Theory of Probability and Its Applications,* **19** (1974), 532–551 (Part I) and **23** (1978), 314–321 (Part II).
33. Lai, T.L. and Ying, Z., Open Bandit Processes and Optimal Scheduling of Queueing Networks. *Adv. Appl. Prob.,* **20** (1988), 447-472.
34. Lu, Y., *Dynamic Scheduling of Stochastic Networks with Side Constraints.* Ph.D. Thesis, Columbia University, 1998.
35. Meilijson, I. and Weiss, G., Multiple Feedback at a Single-Server Station. *Stochastic Proc. and Appl.,* **5** (1977), 195–205.
36. Ross, K.W. and Yao, D.D., Optimal Dynamic Scheduling in Jackson Networks. *IEEE Transactions on Automatic Control,* **34** (1989), 47-53.
37. Ross, K.W. and Yao, D.D., Optimal Load Balancing and Scheduling in a Distributed Computer System. *Journal of the Association for Computing Machinery,* **38** (1991), 676–690.
38. Shanthikumar, J.G. and Sumita, U., Convex Ordering of Sojourn Times in Single-Server Queues: Extremal Properties of FIFO and LIFO Service Disciplines. *J. Appl. Prob.,* **24** (1987), 737–748.
39. Shanthikumar J.G. and Yao D.D., Multiclass Queueing Systems: Polymatroid Structure and Optimal Scheduling Control. *Operation Research*, **40** (1992), Supplement 2, S293–299.
40. Smith, W.L., Various Optimizers for Single-Stage Production. *Naval Research Logistics Quarterly,* **3** (1956), 59–66.
41. Tcha, D. and Pliska, S.R., Optimal Control of Single-Server Queueing Networks and Multiclass M/G/1 Queues with Feedback. *Operations Research,* **25** (1977), 248–258.
42. Tsoucas, P., The Region of Achievable Performance in a Model of Klimov. IBM Research Report RC-16543, IBM Research Division, T.J. Watson Research Center, Yorktown Hts., New York, NY 10598, 1991.
43. Varaiya, P., Walrand, J. and Buyyokoc, C., Extensions of the Multiarmed Bandit Problem: The Discounted Case. *IEEE Trans. Automatic Control,* **30** (1985), 426–439.
44. Weber, R., On the Gittins Index for Multiarmed Bandits. *Annals of Applied Probability,* (1992), 1024–1033.
45. Weber, R. and Stidham, S., Jr., Optimal Control of Service Rates in Networks of Queues. *Adv. Appl. Prob.,* **19** (1987), 202–218.
46. Weiss, G., Branching Bandit Processes. *Probability in the Engineering and Informational Sciences,* **2** (1988), 269–278.
47. Welsh, D., *Matroid Theory*, (1976), Academic Press, London.
48. Whittle, P., Multiarmed Bandits and the Gittins Index. *J. Royal Statistical Society, Ser. B*, **42** (1980), 143–149.
49. Whittle, P., *Optimization over Time: Dynamic Programming and Stochastic Control,* vols. I, II, Wiley, Chichester, 1982.
50. Yao, D.D. and Shanthikumar, J.G., Optimal Scheduling Control of a Flexible Machine. *IEEE Trans. on Robotics and Automation,* **6** (1990), 706–712.
51. Yao, D.D. and Zhang, L., Stochastic Scheduling and Polymatroid Optimization, *Lecture Notes in Applied Mathematics,* **33**, G. Ying and Q. Zhang (eds.), Springer-Verlag, 1997, 333–364.
52. Zhang, L., *Reliability and Dynamic Scheduling in Stochastic Networks.* Ph.D. Thesis, Columbia University, 1997.