Maximizing throughput in queueing networks with limited flexibility

Douglas G. Down [*] George Karakostas [**]

Department of Computing and Software
McMaster University
Hamilton, ON, Canada

Abstract. We study a queueing network where customers go through several stages of processing, with the *class* of a customer used to indicate the stage of processing. The customers are serviced by a set of *flexible* servers, i.e., a server may be capable of serving more than one class of customer and the sets of classes that the servers are capable of serving may overlap. We would like to choose an assignment of servers that achieves the maximal capacity of the given queueing network, where the maximal capacity is λ^* if the network can be stabilized for all arrival rates $\lambda < \lambda^*$ and cannot possibly be stabilized for all $\lambda > \lambda^*$. We examine the situation where there is a restriction on the number of servers that are able to serve a class, and reduce the maximal capacity objective to a maximum throughput allocation problem of independent interest: the Total Discrete Capacity Constrained Problem *(TDCCP)*. We prove that solving *TDCCP* is in general NP-complete, but we also give exact or approximation algorithms for several important special cases.

Keywords: Queueing networks, scheduling, approximation algorithms.

1 Introduction

Consider a system (which we will henceforth call a queueing network), in which discrete entities (or customers) progress through a series of operations (referred to as classes). At each class, a processing step must be performed that requires an amount of time that can be modelled as a random variable. There is infinite waiting room in a queue at each class. Processing at a class is performed by one or more servers. In a traditional queueing network, servers are dedicated to a class. If the queue at a class is empty, the dedicated server(s) there is forced to idle. In the operations research literature, there has been much recent interest in a generalization of this model, in which the servers are flexible, i.e. the customers progress through the network as before, but servers may be capable of performing processing at more than one class (and as such, a decision must be

[*] E-mail: `downd@mcmaster.ca`. Research supported by NSERC grant 239150-2001.
[**] E-mail: `karakos@mcmaster.ca`. Research supported by an NSERC Discovery grant.

made at each point in time as to where a server is located). A typical example of this is a production system where the classes are machines performing manufacturing steps, customers are the parts being produced, and the flexible servers are workers cross-trained to operate multiple machines (see Hillier and So [12], for example). Such models also arise in areas such as power control for wireless networks (Armony and Bambos [4]) and parallel computer systems (Squillante et al. [17]). For an extensive overview of the literature, see Andradóttir et al. [3] and Hopp and van Oyen [13]. A precise mathematical definition of the model is given in the next section.

The design problem in which we are interested is to choose a (dynamic) assignment of servers to classes to address a particular performance objective. In this paper, we are interested in determining the maximal capacity of a given network, where we define the maximal capacity to be λ^* if the network can be stabilized for all arrival rates $\lambda < \lambda^*$ and cannot possibly be stabilized for any $\lambda > \lambda^*$. A number of authors have examined flexible server systems with throughput as a performance measure. In addition to [3] and [8], these include Tassiulas and Ephremides [19], Tassiulas and Bhattacharya [18], Andradóttir, Ayhan, and Down [2], Andradóttir and Ayhan [1], Bischak [6], Zavadlav, McClain, and Thomas [20].

The above references do not constrain the number of servers that may be at a class. This is not a realistic assumption for most settings, as for example, one may have budgetary constraints for training and as a result, one would like to restrict the amount of cross-training, but still achieve reasonable throughput (as compared to a system with no such constraints). In order to address this issue, we need a means to calculate the maximal capacity of a constrained network. We believe that this is the first attempt to address such a problem.

In the area of queueing networks, the use of fluid limits to characterize stability is a standard technique, originating in the work of Rybko and Stolyar [15] and Dai [7]. The central idea in this approach is that one can equate stability of a (stochastic) queueing network with that of a related deterministic model (the fluid model). We emphasize here that determining stability conditions for the original stochastic queueing network is typically extremely difficult. The fluid model approach provides a rigorous connection between the two models and one hopes that the deterministic model is easy (or at the very least easier) to analyze. For the flexible server setting, the fluid limit methodology has been used to break down the determination of maximal capacity to two steps (see [3] and Dai and Lin [8]).

1. Determine the maximal capacity of the fluid model and an optimal allocation of each server's effort.
2. Use the allocation to construct a scheduling policy for the original network.

The framework in [3] gives a standard means by which to perform the second step. Also, if there is no constraint on the number of servers that can be at a class at any one time, it is shown in [3] that the first step reduces to solving a linear programming problem, so as a result the entire problem has been reduced to one that is very tractable. For the more realistic constrained problem considered here, we find that the analysis of the deterministic fluid model is much more difficult. We shall call such a problem the TOTAL DISCRETE CAPACITY CONSTRAINED PROBLEM *(TDCCP)*. For this problem, the second step in the above procedure is unchanged. It is the first step in the procedure which sees the most significant change, and the resulting optimization problem is the focus of this paper.

We show that this problem is NP-complete even for special cases. Hence we look for approximation algorithms for such hard special cases and for the general problem. We achieve an approximation factor of 1/10 for the important case of service rates that depend only on the servers (and not on the classes)[1], and these approximation techniques extend also to the general case (but with a worse approximation factor.) Even more importantly, some of these techniques give exactly the same approximation factors for the *budgetary constraint* version of the problem. In this generalization, a *per service unit cost* of assigning a particular server to a particular class is given, as well as a budget that our total assignment cost should not exceed. Some of our approximation algorithms produce solutions that are within the previous approximation factors without violating the budget.

2 Queueing Network Model

The model we consider is a generalization of that in Andradóttir, Ayhan, and Down [3]. For completeness, we present the model in its entirety.

2.1 Network topology

Consider a network where the location of a customer is given by its class k. We assume that there are K distinct classes, with a buffer of infinite size for each class. Arrivals to a class may occur from inside or outside of the network. Customers arriving from outside of the network do so according to an arrival process with independent and identically distributed (i.i.d.) interarrival times $\{\xi(n)\}$. The associated arrival rate is $\lambda = 1/E[\xi(1)]$. An arrival from outside of the network is routed to class k with probability $p_{0,k}$, with $\sum_{k=1}^{K} p_{0,k} = 1$. Within the network, customers circulate as follows. Upon completion of service at class i, a customer becomes one of class k with probability $p_{i,k}$. The customer leaves the network with probability $1 - \sum_{k=1}^{K} p_{i,k}$. We define the routing matrix P to have (i,k) entry $p_{i,k}$ for $i, k = 1, \ldots, K$ and I to be the $K \times K$ identity matrix. We assume that all customers eventually leave the network, which is equivalent to $(I - P')$ being invertible. (Note that the (i,k) entry of $(I - P')^{-1}$ is the expected number of future visits to class k of a class i customer.)

For technical reasons, we assume that the interarrival times are unbounded and spread out. For more details see [3].

[1] In Section 4.1 we show that the special case of service rates which depend only on the classes can be solved optimally.

2.2 Service mechanism

The network is populated by M servers which service customers within a class according to First Come, First Served order. When switching from class i to class k for the nth time, server j incurs a (possibly zero) switching time of $\zeta^j_{i,k}(n)$. It is assumed that the sequence $\{\zeta^j_{i,k}(n)\}$ is i.i.d. for every $j = 1, \ldots, M$ and $i, k = 1, \ldots, K$. Further, we assume that $\{\zeta^j_{i,i}(n)\}$ is identically zero for all i and j.

Several servers may be simultaneously at a class, in which case they work in parallel. If server j is capable of working at class k, the service time of the nth customer served by server j at class k is given by $\eta_{j,k}(n)$, where the sequence $\{\eta_{j,k}(n)\}$ is assumed to be i.i.d. for each j and k. The associated mean service time for server j at class k is $m_{j,k} = E[\eta_{j,k}(1)]$, with associated service rate $\mu_{j,k} = 1/m_{j,k}$. If server j cannot work at class k, we set $\mu_{j,k} = 0$. We only consider nonpreemptive policies.

The difference between the model in [3] and that considered here is that we put an upper limit, $c_k \leq M$, on the number of servers that can be assigned to a class (a server is assigned to a class if it spends any time at class k).

3 Total Discrete Capacity Constrained Problem

We are first interested in computing the *capacity.* A network operating under a service policy π is said to have capacity λ^π if the system is stable for all values of the arrival rate $\lambda < \lambda^\pi$. We wish to calculate a tight upper bound on the capacity that a given system can achieve (called the *maximal capacity*). In the course of doing so, we identify a means to construct server assignment policies that have capacity that is arbitrarily close to the maximal capacity.

3.1 The Allocation Program

First, we solve the traffic equations for the network, which give the total arrival rate to class k, λ_k, if the network is stable. Here we have

$$\lambda_k = p_{0,k}\lambda + \sum_{i=1}^{K} p_{i,k}\lambda_i,$$

for $k = 1, \ldots, K$. This system of equations is known to have a unique solution if $(I - P')$ is invertible. If we let a_i, $1 \leq i \leq K$ be the unique solution with $\lambda = 1$, then $\lambda_k = \lambda a_k$ is the unique solution of the traffic equations for an arbitrary value of λ.

Let $\delta_{j,k}$ be the proportion of time that server j is working at class k. These proportions exist under the policies considered below. The resulting optimization problem (with variables $\delta_{j,k}$ and λ) that will give us the assignment of servers to classes is:

$$
\begin{aligned}
&\max \lambda && \text{(MP)}\\
\text{s.t.} &\sum_{j=1}^{M} \mu_{j,k}\delta_{j,k} \geq \lambda a_k,\ k = 1,\dots,K && (1)\\
&\sum_{k=1}^{K} \delta_{j,k} \leq 1,\ j = 1,\dots,M, && (2)\\
&\delta_{j,k} \geq 0,\ k = 1,\dots,K, j = 1,\dots,M, && (3)\\
&\sum_{j=1}^{M} \chi\{\delta_{j,k} > 0\} \leq c_k,\ k = 1,\dots,K, && (4)
\end{aligned}
$$

where $\chi\{\cdot\}$ is the indicator function. The constraints in (MP) have the following interpretations. The first, (1), says that the service rate allocated to class k must be greater than the arrival rate. The second and third constraints, (2) and (3), prevent over allocations and negative allocations of a server, respectively. Finally, the constraint (4) limits the flexibility, by only allowing c_k servers to be assigned to work at class k. Let a solution of (MP) be given by λ^*, $\{\delta^*_{j,k}\}$. We will see that λ^* is the desired maximal capacity and $\{\delta^*_{j,k}\}$ is the set of proportional allocations of server j to classes k required to achieve λ^*.

Obviously, the difficulty in solving (MP) comes from the integral constraints (4). Note that in these constraints, although the allocation variables $\delta_{j,k}$ are fractional, the capacity each one is allocated is either 0 or 1 (depending on whether $\delta_{j,k}$ is 0 or not). To the best of our knowledge, we are not aware of other scheduling problems with such constraints. In Section 6 we show that even a simpler variant of the problem is NP-complete. First we consider special cases in Section 4: If the $\mu_{j,k}$'s are independent of j, i.e., $\mu_{j,k} = \mu_k$ for all j, then the problem can be solved in polynomial time. If the $\mu_{j,k}$'s are independent of k, i.e., $\mu_{j,k} = \mu_j$ for all k, then the problem is NP-complete, but can be approximated within a factor 1/10, or better under certain assumptions. For the general case, we show in Section 5 that in polynomial time one can find a solution within a factor $1/10w_{max}$, where $w_{max} := \max_j \max_{k_1,k_2,\mu_{j,k_2}\neq 0} \frac{\mu_{j,k_1}}{\mu_{j,k_2}}$. The bulk of the remainder of the paper is concerned with how one can solve (MP). Before doing this, we complete the connection between solving (MP) and the problem of finding the maximal capacity in the original queueing network.

For the original queueing network, we consider the set of generalized round robin policies. A generalized round robin policy π is given by a set of integers $\{\ell^\pi_{j,k}\}$ and an ordered list of classes V^π_j. Server j servers classes in V^π_j in cyclic order, with server j performing $\ell^\pi_{j,k}$ services at each class in V^π_j (unless server j idles, in which case the server moves to the next class in V^π_j). If the classes in V^π_j are all empty, the server idles at an arbitrary class in V^π_j. The details of how to construct a generalized round robin policy π given a set of required proportional allocations $\{\delta^*_{j,k}\}$ is given in Section 3.3 of [3]. As this can be used directly, we give no further discussion of the construction here.

Define $Q_k(t)$ to be the number of class k customers present at time t and $Q(t)$ be a vector with kth entry $Q_k(t)$. The following theorem gives the strong connection between maximizing capacity in the queueing network and the solution to (MP).

Theorem 1. **(i)** *Any capacity less than λ^* may be achieved. More specifically, for an arrival process with rate $\lambda < \lambda^*$, there exists a dynamic server assignment policy such that the distribution of the queue length process $\{Q(t)\}$ converges to a steady-state distribution φ as $t \to \infty$.*

(ii) *A capacity larger than λ^* cannot be achieved. More specifically, for an arrival process with rate $\lambda > \lambda^*$, as $t \to \infty$, $P(|Q(t)| \to \infty) = 1$.*

The proof of this theorem is a trivial extension of that of Theorem 1 in [3] and is thus omitted. The derivation of the additional constraint (4) is a straightforward exercise, the remainder of the proof is unchanged.

Theorem 1 says that the difficult stochastic optimization problem can be converted into a deterministic optimization problem. The mapping of the solution to the deterministic problem back to a solution to the original stochastic problem does not depend on the complexity of the deterministic problem (it simply uses the resulting solution). For the remainder of the paper, we thus focus on solving (MP). In [3], the deterministic problem is simply (MP), with the constraint (4) removed. This is easily seen to be a linear programming problem, and so there is the appealing result that a difficult stochastic problem becomes a simple deterministic problem. However, in our case, the resulting deterministic problem can also be difficult, as will be seen below. From this point, we refer to the required deterministic optimization problem as the TOTAL DISCRETE CAPACITY CONSTRAINED PROBLEM (TDCCP).

4 Solving TDCCP - special cases

It is instructive to first look at several special cases of TDCCP that give an idea of the inherent complexity.

4.1 The case $\mu_{j,k} = \mu_k$ for all j

Suppose that the service rates are independent of the server and that each server is capable of working at every class, so $\mu_{j,k} = \mu_k$ for $j = 1, \dots, M$. Here, (MP) can be rewritten as

$$\begin{array}{rll} & \max \lambda \text{ s.t.} & \\ \sum_{j=1}^M \delta_{j,k} & \geq \lambda a_k/\mu_k, & k = 1, \dots, K \\ \sum_{k=1}^K \delta_{j,k} & \leq 1, & j = 1, \dots, M, \\ \delta_{j,k} & \geq 0, & k = 1, \dots, K, j = 1, \dots, M, \\ \sum_{j=1}^M \chi\{\delta_{j,k} > 0\} & \leq c_k, & k = 1, \dots, K, \end{array}$$

where $\chi\{\cdot\}$ is the indicator function.

Proposition 1. *If $\mu_{j,k} \equiv \mu_k$, the maximal capacity is*

$$\lambda^* = \min\left(\frac{M}{\sum_{k=1}^{K} a_k/\mu_k}, \min_{1\le k\le K} \frac{c_k\mu_k}{a_k}\right).$$

The proof of Proposition 1 appears in the full version.

4.2 The case $\mu_{j,k} = \mu_j$ for all k

Suppose now that the service rates depend only on the server and that each server is capable of working at every class, so $\mu_{j,k} = \mu_j$ for $k = 1, \dots, K$. In this case (MP) can be written as

$$\begin{aligned} &\max\ \lambda \text{ s.t.} \\ \textstyle\sum_{j=1}^{M} x_{j,k} &\ge \lambda a_k,\ k = 1, \dots, K \\ \textstyle\sum_{k=1}^{K} x_{j,k} &\le \mu_j,\quad j = 1, \dots, M \\ x_{j,k} &\ge 0, \qquad k = 1, \dots, K, j = 1, \dots, M \\ \textstyle\sum_{j=1}^{M} \chi\{x_{j,k} > 0\} &\le c_k, \quad k = 1, \dots, K \end{aligned} \qquad (\text{MP}')$$

where we performed the substitution $x_{j,k} := \mu_j\delta_{j,k},\ \forall j, k$. This case is already NP-complete, as is shown in Theorem 2 in Section 6.

(MP′) actually is an instance of the *maximum concurrent multicommodity k-splittable flow problem* which can be stated as follows: Let $G = (V, E)$ be a directed or undirected graph with integral edge capacities $u_e > 0$, for all $e \in E$. There are l source-sink pairs $(s_i, t_i),\ i = 1, \dots, l$, one for each of l different commodities. For each commodity i there is also a demand d_i, and a bound k_i on the number of different paths allowed for this commodity. Then the maximum concurrent multicommodity k-splittable flow problem is asking for a flow assignment to paths in G that respects the edge capacities and the splittability bounds for the commodities, and routes the maximum possible fraction of all commodity demands *simultaneously.* This, together with several other versions of k-splittable problems, are studied in [5]. Also, when $k_i = 1,\ \forall i$, then these problems are called just *unsplittable* (instead of 1-splittable).

Problem (MP′) is a special case of the multicommodity k-splittable flow problem: the K classes can be seen as K commodities of demand $a_k,\ k = 1, \dots, K$, each with a splittability upper bound of $0 < c_k \le M$. These commodities are routed on the network of Figure 1. All commodities have the same source s, but commodity i has its own sink t_i. Each of the vertices $t_i,\ i = 1, \dots, K$ is connected to all vertices $u_j,\ j = 1, \dots, M$, and s is connected to all vertices $v_j,\ j = 1, \dots, M$. The edge (v_j, u_j) has capacity μ_j for all $j = 1, \dots, M$, while the rest of the edges have infinite capacity. Note that a solution to the maximum concurrent multicommodity k-splittable flow problem on this instance will also give a solution to our original problem (MP′), since every flow path that carries flow f of commodity k through edge (v_j, u_j) corresponds to setting $\delta_{j,k} := f$. And vice versa, a solution to (MP′) gives us also a path flow assignment that achieves the same value for the minimum fraction of commodity demand that is

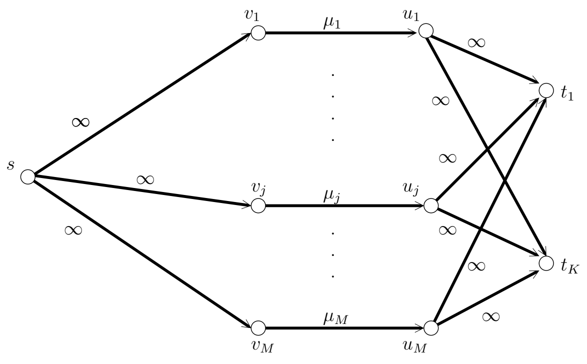

Fig. 1. The graph for our special k-splittable flow instance.

satisfied in the maximum concurrent multicommodity k-splittable flow problem above.

Baier et al. [5] show that any ρ-approximation algorithm for the maximum concurrent unsplittable flow problem yields a $\rho/2$-approximation algorithm for the maximum concurrent k-splittable flow problem. Dinitz et al. [9] present an algorithm that achieves an approximation factor of $\rho = 1/5$ in running time $O(KM(K+M))$, using ideas by Kolliopoulos and Stein [14]. Therefore the solution we get for our problem has a guaranteed *worst-case* performance of at least $1/10$ of the optimum. Note that in our case, the usual *balancing* assumption

$$\text{max demand} \leq \text{min capacity}$$

does not hold, hence the somewhat worse approximation ratios achieved, as compared to the ratios achieved if this assumption holds.

A different approximation algorithm The previous algorithm cannot take advantage of the better approximation factor of 2 for congestion in the unsplittable flow problem, because the balancing assumption doesn't hold in our case. Here we follow a different path, in order to provide an approximation algorithm that under certain assumptions achieves a factor better than $1/10$ for the case $\mu_{j,k} = \mu_j$ for all k. We will reduce our problem to the generalized assignment problem, and then we will use the approximation algorithm by Shmoys and Tardos [16].

The first step of the new algorithm is the same as before: we transform the given problem into an exactly-k-splittable flow problem, with a loss of a factor of $1/2$. Hence commodity k is split into c_k commodities (k,i), $i = 1, \ldots, c_k$, each with demand a_k/c_k.

During the second step, we solve the following concurrent flow problem in the network defined above, which in turn is a relaxation of the concurrent unsplittable flow:

$$\begin{array}{rl} & \max \lambda \text{ s.t.} \\ \sum_{j=1}^{M} x_{j,(k,i)} & \geq \lambda \frac{a_k}{c_k}, \forall k, i \\ \sum_{(k,i)} x_{j,(k,i)} & \leq \mu_j, \quad \forall j \\ x_{j,(k,i)} & \geq 0, \quad \forall i, j, k \end{array} \tag{LP-NEW}$$

If x^*, λ^* is the optimal solution for (LP-NEW), then define $\lambda_{(k,i)} := (\sum_{j=1}^{M} x^*_{j,(k,i)})/(a_k/c_k)$. Obviously $\lambda_{(k,i)} \geq \lambda^* > 0$, $\forall (k,i)$. Also, we define $y_{j,(k,i)} := \frac{c_k}{\lambda_{(k,i)} a_k} x^*_{j,(k,i)}$ and $p_{j,(k,i)} := \frac{a_k}{c_k \mu_j}, \forall i, j, k$. Then y satisfies the following system of inequalities:

$$\begin{array}{rl} \sum_{j=1}^{M} y_{j,(k,i)} & = 1, \qquad \forall k, i \\ \sum_{(k,i)} p_{j,(k,i)} y_{j,(k,i)} & \leq 1/\lambda^*, \forall j \\ y_{j,(k,i)} & \geq 0, \qquad \forall i, j, k \end{array}$$

This is exactly the relaxation of the problem (without costs) of scheduling unrelated parallel machines that [16] studies. We can think of the commodities (k,i) as jobs, the edges of capacities μ as machines, $p_{j,(k,i)}$ as the processing time of job (k,i) on machine j, $1/\lambda^*$ as the makespan, and y as a feasible (fractional) assignment of jobs to machines that achieves this makespan. Suppose that there is some $\rho > 0$ such that $p_{j,(k,i)} \leq \rho/\lambda^*, \forall i, j, k$. Then Theorem 2.1 of [16] implies that their algorithm produces an (integral) assignment of jobs to machines $\hat{y}$ with makespan at most $(1+\rho)/\lambda^*$. This algorithm is the third step of our algorithm.

Our solution assigns $\hat{x}_{j,(k,i)} := \frac{\lambda_{(k,i)} a_k}{c_k} \hat{y}_{j,(k,i)}$ (note that for every (k,i), these values are going to be 0 for all j except one.) It is easy to prove the following:

Lemma 1. *The solution produced by the algorithm above is within* $1/2(1+\rho)$ *of the optimum.*

Proof. The solution $\hat{x}$ satisfies the constraints of (LP-NEW) for $\lambda := \lambda^*/(1+\rho)$. Hence it approximates the maximum concurrent unsplittable flow within a factor of 1/2. Together with the approximation factor of 1/2 from the first step, this implies the lemma.

4.3 The case $\mu = \alpha \cdot \beta^T$

This case can be generalized to any $M \times K$ matrix μ which is the product of an $M \times 1$ vector α and the transpose of a $K \times 1$ vector β, i.e., $\mu = \alpha \cdot \beta^T$ (in other words, the service rates satisfy $\mu_{j,k} = \alpha_j \beta_k$). Then it is easy to see that the initial problem (MP) is equivalent to

$$\begin{array}{rl} & \max \lambda \text{ s.t.} \\ \sum_{j=1}^{M} x_{j,k} & \geq \lambda b_k, \; k = 1, \ldots, K \\ \sum_{k=1}^{K} x_{j,k} & \leq \alpha_j, \quad j = 1, \ldots, M \\ x_{j,k} & \geq 0, \quad k = 1, \ldots, K, j = 1, \ldots, M \\ \sum_{j=1}^{M} \chi\{x_{j,k} > 0\} & \leq c_k, \quad k = 1, \ldots, K \end{array} \tag{MP$''$}$$

where $x_{j,k} := \alpha_j \delta_{j,k}$, for all j, k, and $b_k := a_k/\beta_k$. (MP$''$) then falls into the case of Section 4.2.

4.4 Extension to TDCCP with costs

We can extend TDCCP by introducing *costs* to the assignment of servers to classes. Let $c_{j,k}$ be the per unit cost of assigning server j to class k. Hence, if $\delta_{j,k}$ is the fraction of its effort dedicated by j to k, then the cost incurred is $c_{j,k}\delta_{j,k}$. For example, the assignment of a worker to a machine where he has no expertise may incur a bigger cost (because of training needs, damages because of deficient products he produces etc.) than the cost of an experienced worker to the same machine. Together with these costs $c_{j,k}$, we are also given a *budget* that cannot be exceeded by our final assignment. Hence we are asked for an assignment of servers to classes that respect the given budget and maximizes the throughput.

The algorithms of [5], [16] used in Section 4.2 are *cost preserving*. When in the first step we transform the budget-constrained k-splittable flow problem into a budget-constrained exactly-k-splittable flow problem, [5] proves that the optimal solution of the latter is not only an 1/2 approximation of the former, but it also respects the initial budget constraint. Also the algorithm of [16] we use in Section 4.2 produces an assignment that always respects the budgetary constraint (although it may not produce the optimal makespan).[2]

5 Solving TDCCP - general case

For the general case, let

$$w_j := \frac{\mu_j^{max}}{\mu_j^{min}}, \; j = 1, 2, \ldots, M$$

where $\mu_j^{max} := \max_k\{\mu_{j,k}\}$ and $\mu_j^{min} := \min_k\{\mu_{j,k}\}$. Note that $\mu_{j,k} = 0$ implies that $\delta_{j,k} = 0$, so we will assume that $\mu_{j,k} > 0$ for all j, k. Also, let $w_{max} := \max_j\{w_j\}$, and let δ^*, λ^* be the optimal solution to (MP). Instead of the original problem (MP), we will try to solve (approximately) the following problem:

$$\begin{aligned} & \max \lambda \text{ s.t.} \\ \textstyle\sum_{j=1}^{M} \mu_{j,k}\delta_{j,k} &\geq \lambda a_k, \quad k = 1, \ldots, K \\ \textstyle\sum_{k=1}^{K} \mu_{j,k}\delta_{j,k} &\leq \mu_j^{max}, \; j = 1, \ldots, M \\ \delta_{j,k} &\geq 0, \quad\quad k = 1, \ldots, K, j = 1, \ldots, M \\ \textstyle\sum_{j=1}^{M} \chi\{\delta_{j,k} > 0\} &\leq c_k, \quad\quad k = 1, \ldots, K. \end{aligned} \qquad \text{(NEW MP)}$$

It is clear that, as in Section 4.2, we can set $x_{j,k} := \mu_{j,k}\delta_{j,k}$ in (NEW MP) to get exactly the same formulation as (MP$'$). Hence we can apply the same techniques we applied in Section 4.2, to obtain an approximate solution $\hat{x}, \hat{\lambda}$, which is within

[2] Obviously the costs in the budgetary constraint in each of the LP formulations above are scaled following the scaling of the assignment variables.

1/10 of the optimum solution (of (MP′)). Then we output the following solution to the original problem:

$$\delta_{j,k} := \frac{\hat{x}_{j,k}}{w_j \mu_{j,k}},\ \forall j, k. \tag{5}$$

The following proposition is proven in the full version:

Proposition 2. *Solution (5) is a feasible solution for (MP), and achieves a λ of value at least $\lambda^*/10w_{max}$.*

This result extends to the case of a budgetary constraint problem, i.e. the approximation factor can be achieved without violating the (given) budget (cf. Section 4.4.)

6 NP-completeness

We reduce a slight variation of the classical PARTITION problem (see [SP12] in [10]) to the version of our problem that is studied in Section 4.2, which, by abusing the terminology a little bit, we will call problem (MP′):

MP′
Instance: We are given (MP′) and $\lambda^* \in \mathbb{R}$.
Question: Is the solution of (MP′) greater than or equal to λ^*?

Obviously this problem is in NP (given λ^* and a solution to MP′, one can easily check whether its objective is greater than or equal to λ^*). The PARTITION problem variation we reduce it to is the following:

PARTITION
Instance: Finite set A of even cardinality and a size $s(a) \in \mathbb{Z}^+$ for each item $a \in A$.
Question: Is there a subset $A' \subseteq A$ of cardinality $|A|/2$ and such that $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$?

Given the PARTITION instance, we identify the elements of A with the numbers $1, 2, \ldots, |A|$. Let $S := \sum_{j=1}^{|A|} s(j)$ be the total size. We set $K := 2, M := |A|$ and $\mu_j := s(j),\ j = 1, \ldots, |A|$. We also set $c_1 = c_2 := |A|/2$, and $a_1 = a_2 := 1$. Finally we set $\lambda^* := S/2$. Therefore we get an instance of (MP′) in polynomial time. From now on, when we refer to (MP′), we actually refer to this specific instance we constructed. We can prove (proof in the full version) the following

Theorem 2. PARTITION *has a solution iff (MP′) achieves $\lambda \geq \lambda^*$.*

References

1. S. Andradóttir and H. Ayhan. Throughput maximization for tandem lines with two stations and flexible servers. *Operations Research*, 53:516–531, 2005.
2. S. Andradóttir, H. Ayhan, and D.G. Down. Server assignment policies for maximizing the steady-state throughput of finite queueing systems. *Management Science*, 47:1421-1439, 2001.

3. S. Andradóttir, H. Ayhan and D.G. Down. Dynamic server allocation for queueing networks with flexible servers. *Operations Research*, 51:952-968, 2003.
4. M. Armony and N. Bambos. Queueing networks with interacting service resources. *Proceedings of the 37th Annual Allerton Conference on Communications, Control, and Computing*, 42-51, 1999.
5. G. Baier, E. Köhler, and M. Skutella. On the k-splittable flow problem. *Proceedings of ESA'02*.
6. D.P. Bischak. Performance of a manufacturing module with moving workers. *IIE Transactions*, 28:723-733, 1996.
7. J.G. Dai. On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models. *Annals of Applied Probability*, 5:49-77, 1995.
8. J.G. Dai and W. Lin. Maximum pressure policies in stochastic processing networks. *Operations Research*, 53:197–218, 2005.
9. Y. Dinitz, N. Garg, and M. Goemans. On the single-source unsplittable flow problem. *Combinatorica*, 19 (1999), pp. 17–41.
10. M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to the theory of NP-Completeness.* W. H. Freeman and Co., 1979.
11. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*, Chapter 6. Springer-Verlag, 1993.
12. F.S. Hillier and K.C. So. On the simultaneous optimization of server and work allocations in production line systems with variable processing times. *Operations Research*, 44:435-443, 1996.
13. W.J. Hopp and M.P. van Oyen. Agile workforce evaluation: A framework for cross-training and coordination. *IIE Transactions*, 36:919–940, 2004.
14. S. G. Kolliopoulos and C. Stein. Approximation algorithms for single-source unsplittable flow. *SIAM J. Computing* 31:919-946, 2002.
15. A.N. Rybko and A.L. Stolyar. Ergodicity of stochastic processes describing the operation of open queueing networks. *Problems of Information Transmission*, 28:199-220, 1992.
16. D. B. Shmoys and . Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming A*, Vol. 62(3), pp. 461 - 474, 1993.
17. M.S. Squillante, C.H. Xia, D.D. Yao, and L. Zhang. Threshold-based priority policies for parallel-server systems with affinity scheduling. *Proceedings of the 2001 American Control Conference*, 2992-2999, 2001.
18. L. Tassiulas and P.B. Bhattacharya. Allocation of independent resources for maximal throughput. *Stochastic Models*, 16:27-48, 2000.
19. L. Tassiulas and A. Ephrimedes. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37:1936-1948, 1992.
20. E. Zavadlav, J.O. McClain, and L.J. Thomas. Self-buffering, self-balancing, self-flushing production lines. *Management Science*, 42:1151-1164, 1996.