Finite horizon control of processing networks via fluid approach: Separated continuous linear programs, infinite virtual buffers and maximum pressure policies

Gideon Weiss *
Department of Statistics
The University of Haifa
Mount Carmel 31905, Israel.
gweiss@stat.haifa.ac.il

September 31, 2005
Preprint — comments are welcome

Abstract
We consider systems in which many items are evolving over time by sharing common resources, and the problem of how to control such systems by allocating resources to various activities which schedule, route and process these items. We represent this by a processing network as defined by Harrison, with the added feature of infinite virtual buffers, which can model exogenous input and output. We address the problem of transient control of such processing networks over a finite time horizon. We use a fluid approach, in which we approximate the processing network by a deterministic continuous linear fluid model and formulate its optimization as a separated continuous linear program. This can be solved by a new simplex type algorithm of Weiss, and the solution consists of piecewise constant allocations of the activities, with a finite number of breakpoints. This optimal fluid solution is then used to control the original system. We use the maximum pressure policy of Dai and Lin to track the optimal fluid solution. We prove asymptotic optimality of this fluid approach: As the numbers of items in the system and the processing rates increase, the scaled system converges almost surely to the optimal fluid solution, and the scaled objective value converges to the optimum of the system.

Keywords: Queueing, manufacturing, communication networks, vehicle traffic control, control of multiclass queueing networks, processing networks, infinite virtual buffers, maximum pressure policies, fluid approximations, finite horizon online control of systems, continuous linear programming, asymptotic optimality.

1 Introduction

A fundamental problem in operations research is to find ways to control systems in which many items undergo changes in location, changes in state, splitting or combining, and this evolution

*Research supported in part by Israel Science Foundation Grant 249/02
of the items is achieved by some activities which receive, transmit, transport, assemble, disassemble, and process the items, and all the activities are sharing the consumption of some limited resources. The system is to be controlled by allocating resources to activities over time so as to achieve some goals and optimize some performance measures.

Examples of application areas in which this type of problem is crucial include: Manufacturing systems where parts in process share machines, and we need to determine scheduling, processing rates, and routing of the parts through the plant, on a timescale of hours or days [14, 73, 78]. City wide vehicle traffic where individual cars are sharing the use of roads and of intersections, on their way from departure points to destinations, and we need to schedule traffic lights at intersections, and route cars between entry and exit, over the rush hour period [13, 24, 25, 44, 65]. Communication networks where individual messages need to be routed from source to destination, sharing finite capacity links, over short time horizons [28, 43, 47, 74]. Data switches in high speed routers [20]. Internet services where sessions are in progress, in which data is down- or up-loaded, between service providers and customers, and sessions need to be scheduled and service rates need to be allocated so as to satisfy quality of service requirements [64]. Wireless networks where time slots are to be allocated to customers at each broadcasting base station. Supply chain management where orders are moved along the supply chain to satisfy demands. Multi project scheduling, where the resources of a company are allocated to activities of the various projects to guarantee their timely progress [31].

In practice such systems are controlled by a combination of a large variety of methods, some decentralized and some centralized, using tools from information tecnology to keep track of the system, simulation, pert-CPM, MRP, dispatch rules, protocols, etc. Optimization is usually only used for small subsystems, and much is left to ad hoc methods.

One theoretical approach to such problems is to consider them as deterministic, discrete optimization problems of scheduling and/or routing [26, 30, 32, 42, 53, 71]. In practice systems are often too large, sometimes by many orders of magnitude, to be solved in this way. Furthermore, an optimal solution to the deterministic discrete problem may not withstand the trial of application: As it is implemented over time, inaccuracies in the data and unexpected events (many small ones and a few large ones) will accumulate and interfere with the solution, and there is no theory to say how close or far from optimum the result may be.

Another theoretical approach is to model these problems by discrete stochastic systems and solve them as Markov decision problems, or approximate them on a diffusion scale by a continuous stochastic Brownian control problem [33, 34, 38, 39, 41, 45, 51, 52, 55, 76, 79]. Again, often the problems are much too large to be handled in this way. Furthermore, Markov decision problems or Brownian control problems usually focus on the optimization of the steady state of the system. However, in many practical problems of this form the system never reaches a steady state, and it certainly does not forget its initial state over the time horizon of interest.

In [82] we have suggested in outline a middle road approach which avoids some of these difficulties. We discard much of the detailed information of the system and classify the items into a limited number of classes, which we model as a stochastic system. In our control we do not distinguish between different items of the same class. We call this online control, as our decisions use only the current count of items in each class. However, we retain the deterministic approach objective, of optimizing the system over a finite time horizon. Such
problems of control of a stochastic system over a transient period are harder than either the
deterministic finite horizon or the stochastic steady state problems. For this reason we suggest
solving them approximately through a fluid approach. Fluid approximations have been a
major tool in the research on multi-class queueing networks, they have been used to verify
the stability of networks, to evaluate performance in steady state, and to control multi-class
queueing networks so as to improve their steady state performance, some references include
[9, 14, 15, 16, 22, 27, 56, 59, 60, 61, 62, 63]. We believe that our approach which considers
the fluid approximation as a tool to optimize the transient system over a finite time horizon is
novel [80, 81, 82].

In [82] we suggested the following three steps: (i) Model the system as a stochastic pro-
cessing network, and approximate the system by a deterministic continuous fluid model. (ii)
Find the optimal fluid solution for the fluid model. (iii) Apply an online control to the original
system which will track the fluid solution. At that time we did not have an algorithm to solve
the fluid model, we did not have a policy to track the fluid solution, and we were therefore also
unable to assess the performance of our proposed approach.

In the intervening years there has been progress in two areas: We have (Weiss [85]) found
a finite exact algorithm to solve separated continuous linear programs, a problem which has
been open for 50 years. This enables us to calculate optimal fluid solutions. Dai and Lin
[19] put forward the maximum pressure policy which is a decentralized control policy for
stochastic processing networks that is guaranteed to be stable, and is also conjectured to be
asymptotically optimal on a diffusion scale, thus achieving significant progress on a problem
paramount in stochastic networks research for more than 10 years. The maximum pressure
policy, combined with the concept of infinite virtual buffers (introduced by Weiss [48, 84])
enables us to track the fluid solution. This now enables us to carry out the program outlined
in [82]. In this paper we describe the various steps of our fluid approach, and prove that it is
asymptotically optimal.

The rest of the paper is structured as follows: In Section 2 we illustrate the fluid approach
by a small example. In Section 3.1 we introduce separated continuous linear programs (SCLP),
discuss their theory, and point out those features of their solution which are important for the
fluid approach. In Section 3.2 we define processing networks with infinite virtual buffers which
extend the definition of stochastic processing networks given by Harrison [35, 36, 37]. In
Section 3.3 we describe maximum pressure policies, and state the stability result of Dai and
Lin, which continues to hold for processing networks with infinite virtual buffers. Section 4
provides the recipe for control of a multi-class queueing network by our fluid approach: In
Section 4.1 we formulate the problem and its fluid approximation, in Section 4.2 we describe
the fluid solution, and in Section 4.3 we prescribe how to use maximum pressure policies to
track the optimal fluid solution. Section 5 contains the theoretical result of this paper: we show
that our procedure is asymptotically optimal as the number of items in the stochastic system
tends to infinity. In Section 6 we extend these results to more general processing networks.

This paper establishes that, at least in theory, the fluid approach makes it possible to
control a large and complex system in its entirety over a finite time horizon, by solving an
SCLP centrally, to obtain an optimal fluid solution, and tracking the fluid solution with a
decentralized maximum pressure policy, and this fluid approach is asymptotically optimal.
The paper is mainly expository, as the main results on solution of SCLP and on maximum pressure policy, as well as processing networks and infinite virtual buffers have been discussed in other papers — here we just formulate the various results, putting them together, and prove the asymptotic optimality (which follows easily from the results of Dai and Lin).

This leaves us with the challenging problem to implement the fluid approach to some pilot examples in various application areas, so as to establish its viability and relevance, on the way to its use in practice.

2 Example

To illustrate our approach we consider a two machine three step reentrant line manufacturing system. Each item in the two machine three step reentrant line undergoes three processing steps, the first in machine 1, the second in machine 2, and the third back in machine 1. We wish to schedule the processing of items so as to minimize the total holding costs of the items in the system over a finite time horizon.

Consider first a deterministic finite horizon version of this problem. Here one has a set of \( \ell = 1, \ldots, N \) items, with processing times \( a_\ell, b_\ell, c_\ell \) for processing steps 1, 2, 3, and we wish to complete processing of all \( N \) items so as to minimize the sum of the completion times (flowtime). This problem is NP-hard. In fact in the special case of \( a_\ell = 0 \) this is the problem of minimizing flowtime on a two machine flowshop which is known to be (strongly) NP-hard [29]. Some special cases are easily solved, for example, if \( a_\ell = b_\ell = c_\ell, \ell = 1, \ldots, N \) then priority to buffer 3 over 1 and shortest processing time first at each buffer is optimal.

Consider next a stochastic formulation, which has been studied extensively [80, 21, 15, 2, 83]. Denote by \( Q_k(t), k = 1, 2, 3 \) the number of items in buffer \( k \), waiting for step \( k \) (including those in service). Discard the detailed information of processing times and replace it by a processing time distribution for each step \( k = 1, 2, 3 \), with average processing time \( m_k \), and processing rate \( \mu_k = 1/m_k \), so that processing times of items are all independently drawn from these distributions. Model arrivals as a stochastic renewal process of rate \( \alpha \). If \( \alpha(m_1 + m_3) \leq 1, \alpha m_2 \leq 1 \) then any policy which is non-idling (machine does not idle if there is work which it can do) will keep this system stable [21]. One wishes to schedule processing by machine 1 (performing steps 1 or 3) over a long time horizon, so as to minimize average holding costs or equivalently minimize average cycle times. The optimal policy, even in the case that the interarrival and the processing times are exponential, is complicated. It can be obtained by solving a Markov decison problem (MDP). Solving the MDP will yield a policy which optimizes the steady state of the system: It will give a switching curve, which will give priority to buffer 3 over buffer 1, if \( Q_3(t) > H(Q_1(t), Q_2(t)) \). The actual calculation of the switching curve \( H \) is hard. If the processing times and interarrivals are not exponential then the MDP is totally intractable. If \( \alpha, \frac{1}{m_1 + m_3}, \frac{1}{m_2} \) are all close to each other the system is operating under balanced heavy traffic conditions. In that case one can approximate the problem by a Brownian control problem. Such Brownian control problems have been suggested as approximations for the queueing system, and solved [40, 79].

We come now to our fluid approach. We discard the detailed data of the deterministic
model, and use the stochastic model to describe the system. However, similar to the deterministic problem, we wish to minimize the total holding costs for a finite number of items \( N \), over a finite time horizon \((0, T)\). The system is described by the stochastic network process \((Q(t), T(t))\), where \(Q_k(t)\) is the number of items in buffer \( k \) at time \( t \), and \( T_k(t)\) is the cumulative processing time given to buffer \( k \) during \((0, t)\). The dynamics of the system are:

\[
\begin{align*}
Q_1(t) &= Q_1(0) - S_1(T_1(t)) \\
Q_2(t) &= Q_2(0) + S_1(T_1(t)) - S_2(T_2(t)) \\
Q_3(t) &= Q_3(0) + S_2(T_2(t)) - S_3(T_3(t))
\end{align*}
\] (2.1)

where \(Q_1(0) + Q_2(0) + Q_3(0) = N\), \(0 < t < T\), and we let \(S_k(s)\) count the number of items in buffer \( k \) which are completed by processing duration \( s \). We wish to allocate \( T(t) \) so as to:

\[
\min E\left[ \int_0^T (Q_1(t) + Q_2(t) + Q_3(t)) dt \right].
\] (2.2)

It is easy to appreciate that the computational burden of solving this problem exactly, is higher than either the deterministic scheduling problem or the stochastic Markov decision problem.

The fluid problem which approximates our problem is:

\[
\begin{align*}
q_1(t) &= q_1(0) - \int_0^t \mu_1 u_1(s) ds \\
q_2(t) &= q_2(0) - \int_0^t (\mu_2 u_2(s) - \mu_1 u_1(s)) ds \\
q_3(t) &= q_3(0) - \int_0^t (\mu_3 u_3(s) - \mu_2 u_2(s)) ds \\
u_1(t) + u_3(t) &\leq 1 \\
u_2(t) &\leq 1 \\
u(t), q(t) &\geq 0, t \in (0, T).
\end{align*}
\] (2.3)

Here \(q_k(t)\) is the fluid level in buffer \( k \) which approximates \(Q_k(t)\). \( \mu_k \) is the deterministic processing rate for buffer \( k \), and \( u_k(t)\) is the fraction of the machine which is allocated to the processing of buffer \( k \) at time \( t \), so that \(\int_0^t u_k(s) ds\) approximates \(T_k(t)\).

This fluid problem which is a deterministic optimal control problem is in fact a separated continuous linear program of the kind solved in Weiss [85].

The solution depends on the parameters \(m_1, m_2, m_3, q_1(0), q_2(0), q_3(0), T\). If we assume \(m_2 > m_1 + m_3\), and some specific values of \(q_1(0), q_2(0), q_3(0), T\), the fluid solution is typically as follows: The fluid solution partitions the time horizon into four intervals, and uses constant
allocations $u^m_k$ within each interval, given by:

\begin{align*}
  u^1_1 &= 0, & u^1_2 &= 0, & u^1_3 &= 1, & 0 < t < t_1, \\
  u^2_1 &= 0, & u^2_3 &= 1, & u^2_3 &= 1, & t_1 < t < t_2, \\
  u^3_1 &= \frac{\mu_2}{\mu_1}, & u^3_2 &= 1, & u^3_3 &= 1 - \frac{\mu_2}{\mu_1}, & t_2 < t < t_3, \\
  u^4_1 &= \frac{\mu_2}{\mu_1}, & u^4_2 &= 1, & u^4_3 &= \frac{\mu_2}{\mu_3}, & t_3 < t < T.
\end{align*}

The resulting fluid buffer levels $q(t)$ are continuous piecewise linear. These fluid levels are described in Fig 1, in which we plot $q_1, q_1 + q_2, q_1 + q_2 + q_3$ against time.

Note that in this fluid solution the system is not empty at the time $T$, and that the entire solution will change if we change the value of $T$, since this will cause the breakpoints at which the controls are changed to shift.

The exact values of the breakpoints $t_1, t_2, t_3$ and the lengths of the 4 intervals between the breakpoints $\tau_m = t_m - t_{m-1}, m = 1, \ldots, 4$ are calculated from the linear equations:

$$\mu_2 \tau_2 = q_2(0)$$
\[
\mu_3 \tau_1 + (\mu_3 - \mu_2) \tau_2 + \frac{m_2 - m_1 - m_3}{m_2 m_3} \tau_3 = q_3(0) \\
- \frac{\mu_2}{\mu_1} \tau_3 + \frac{\mu_2}{\mu_3} \tau_4 = 0 \\
\tau_1 + \tau_2 + \tau_3 + \tau_4 = T
\]  

(2.5)

It is important to interpret these equations. At \( t_2, t_3 \) buffers 2 and 3 become empty, and this is expressed in the first two equations. The time \( t_1 \) is more subtle. The optimal processing of fluid from buffer 2 between \( 0, t_2 \) is not unique, but if we assume that holding fluid in buffer 2 is marginally more expensive than in buffer 3, then \( t_1 \) is the time that the dual shadow price of machine 2 is changing from a value of zero in the interval \( (0, t_1) \) to a positive value in the intervals \( (t_1, t_2) \), and therefore from that moment onwards we need to utilize machine 2 fully (it needs to have slack capacity zero). The third equation expresses the fact that the shadow price of machine 2 is zero at \( t_1 \). The last equation simply states that the length of all the intervals is \( T \), and it expresses the dependence of the solution on the time horizon \( T \).

We use a fluid tracking maximum pressure policy (see [19]) to track the fluid solution. We first define a residual process \( \tilde{Q}(t) \), which will measure the deviation of the state of the actual queueing network from the fluid solution. Initially, \( \tilde{Q}(0) = 0 \). In each of the 4 intervals of the fluid solution the definition of \( \tilde{Q}(t) \) depends on which fluid buffers are empty, and which are non-empty, as detailed in the following table. We give the values of \( d\tilde{Q} \), the rate of change of \( \tilde{Q} \), which is constant throughout each interval, except for the times at which processing of items is completed, when it may have jumps of \( \pm 1 \):

<table>
<thead>
<tr>
<th>( 0 &lt; t &lt; t_1 )</th>
<th>( d\tilde{Q}_1(t) )</th>
<th>( d\tilde{Q}_2(t) )</th>
<th>( d\tilde{Q}_3(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 &lt; t &lt; t_2 )</td>
<td>0</td>
<td>( \mu_2 - dS_2(T_2(t)) )</td>
<td>( \mu_3 - dS_3(T_3(t)) )</td>
</tr>
<tr>
<td>( t_2 &lt; t &lt; t_3 )</td>
<td>( \frac{\mu_2}{\mu_1} - dS_1(T_1(t)) )</td>
<td>( dQ_2(t) )</td>
<td>( 1 - \frac{\mu_2}{\mu_1} - dS_3(T_3(t)) )</td>
</tr>
<tr>
<td>( t_3 &lt; t &lt; T )</td>
<td>( \frac{\mu_2}{\mu_1} - dS_1(T_1(t)) )</td>
<td>( dQ_2(t) )</td>
<td>( dQ_3(t) )</td>
</tr>
</tbody>
</table>

(2.6)

It can be seen that the residual for a buffer \( k \) which is non-empty in the fluid solution increases at the rate \( w^m_k \), which we call the nominal input rate of this buffer; if buffer \( k \) is empty in the fluid solution then \( \tilde{Q}_k(t) = Q_k(t) \).

The residual process is used to calculate the pressure of the network, at time \( t \). The pressure is used to determine a non-splitting non-preemptive policy for the network: Whenever machine 1 is available it can either be idled, or it can choose to work on the next in line job from buffer 1 or the next in line job from buffer 3. Whenever machine 2 is available it can either be idled or it can choose to work on the next in line job in buffer 2. The decision which is taken is to choose the available non empty buffer or idle so as to maximize the calculated pressure. The pressure for idling is 0. Buffer \( k \) is available if \( Q_k(t) > 0 \). The various values of the pressure, and the decisions are summarized in the following table:
In figure 1 we have drawn the step functions of $Q_1, Q_1 + Q_2, Q_1 + Q_2 + Q_3$, as an illustration of how the max pressure policy tracks the fluid solution.

We will show that as $Q(0), \mu$ increase, the scaled queue length process converges to the fluid solution.

**To be done:** In the following table we compare the fluid solution for given $q(0), \mu$, with the values obtained by simulating the queueing network, with initial buffers levels $Q_N(0) = Nq(0)$ and processing rates $\mu_N = N\mu$. The table lists the objective value divided by $N$, for $N = 5, 10, 20, 100, 1000$. For each value of $N$ we simulated 100 random sequences of processing times, and we report mean, variance, and various quantiles of this sample.

3 Preliminaries

3.1 Separated continuous linear programs

Continuous linear programs were introduced by Bellman [11] to analyze some economic models. Dantzig [23] and his students Perold [66] and Anstreicher [8], worked on this problem, and Anderson formulated the sub-class of separated continuous linear programs (SCLP) for a job shop scheduling application [3, 4]. Despite continuous research by Anderson, Philpott and Nash [5, 6, 7], and later by Pullan [67, 68, 69], (see also Bertsimas and Luo [54], Shapiro [72] and Barvinok [10]) much of the theory remained unexplored, and all previous suggestions for solution were based on LP approximations by discretizing time. In a recent paper [85] we introduced a finite simplex type algorithm to solve SCLP exactly in a finite number of steps. The analysis in [85] also reveals important features of the solutions.

In [85] we have focused on the following SCLP: Find a vector of control functions $u(t)$ and a vector of state functions $q(t)$, to optimize

$$\min \int_0^T ((r' + (T-t)c')u(t) + d'q(t)) \, dt$$

SCLP

s.t. $\int_0^T Gu(s)ds + Fq(t) = \alpha + at$,

$Hu(t) = b,$

$q(t), u(t) \geq 0, \quad t \in [0, T].$
In formulating the fluid approximation of our finite horizon problem we always obtain an SCLP of this form. In fact, SCLP define a much wider range of problems than those obtained directly as fluid approximations of our finite horizon discrete stochastic processing network problems.

Typically the SCLP which we obtain is feasible and bounded. The algorithm of [85] requires in addition that the problem be non-degenerate. A precise sufficient condition is that \[
\begin{bmatrix}
a \\
b
\end{bmatrix}
\]
is in general position to \[
\begin{bmatrix}
G & F \\
H & 0
\end{bmatrix}
\] and \[
\begin{bmatrix}
c \\
d
\end{bmatrix}
\]
is in general position to \[
\begin{bmatrix}
G & F \\
H & 0
\end{bmatrix}'. In fact in many of our applications the SCLP is degenerate. In that case we need to perturb the problem, by perturbing some of the values in \(F, G, H, a, b, c, d\). Such a perturbation can often be done in a systematic and meaningful way: In the example of Section 2, one can introduce slightly higher holding costs for items which are in more advanced stage of completion. As we noted, if the holding cost in buffer 2 is slightly higher than that in buffer 3 this determines the time \(t_1\) uniquely, and avoids degeneracy of the problem. Once the problem is perturbed, it has a unique solution, and can be solved in a finite number of steps by our algorithm (Weiss [85]). This solution can then be rounded off to an optimal solution of the original unperturbed problem.

For the purpose of this paper the important feature of the solution is that it consists of a partition of the time horizon \(0 = t_0 < t_1 < \cdots < t_M = T\), so that the optimal solution uses constant controls in each interval, and the states (fluid buffer levels) are continuous piecewise linear functions. Furthermore, the values of the controls, and the slopes of the states in the \(m\)th interval, which we denote by \(u_j^m, \dot{q}_k^m\), are optimal basic solutions of a linear programming problem which we call the rates-LP:

\[
\begin{align*}
\text{max} & \quad c' u + d' \dot{q} \\
\text{Rates-LP} & \quad \text{s.t.} \quad Gu + F \dot{q} = a, \\
& \quad Hu = b,
\end{align*}
\]

The rates-Lp (3.4) is solved in each interval under different sign restrictions on the variables \(u, \dot{q}\), where some of the \(\dot{q}\) are restricted to be non-negative, while others are unrestricted, and some of the \(u\) are restricted to be non-negative while others are restricted to be equal to 0.

The SCLP simplex algorithm can be extended to solve SCLP with piecewise constant data. This means that the time horizon is divided into time ranges with different values of the model parameters \(G, F, H\) and the rates \(a, b, c, d\). The optimal solution still consists of piecewise constant controls and continuous piecewise linear states, however, each of the change points in the data is now also a breakpoint in the solution.

We believe that this generalization is extremely important for the solution of practical problems in which typically the environment will change over the finite time horizon. A striking example to this is control of traffic over the rush hour period, where the rate of cars entering the system is increasing first and decreasing later.
3.2 Processing networks with infinite virtual buffers

Processing networks were introduced by Harrison [35, 37, 37]. They generalize multiclass queueing networks and serve to model a much wider class of systems and policies. A processing network consists of three elements: Items which are classified into classes, so that the state of the network is given by the number of items in each class, activities which are used to process items from the various classes, thereby changing the number of items and their classification, and resources which are consumed by the activities. Infinite virtual buffers were introduced by Weiss et al in [1, 2, 48, 83, 84] (see also Goodman and Massey [57]) to model the connection of a processing network to the outside world. We now describe processing networks with infinite virtual buffers.

Items are classified into the classes $k \in K$. Items of class $k$ will be stored in buffer $k$. We distinguish two types of buffers: Buffers $k \in K_0 \subseteq K$ are standard buffers, with a finite number of items, and we let $Q_k(t) \geq 0$, $k \in K_0$ count the number of items in class $k$. Buffers $k \in K_\infty = K \setminus K_0$ have an unlimited supply of items, and are referred to as infinite virtual buffers. Since these buffers are virtually infinite we use their level $Q_k(t)$, $k \in K_\infty$ as a relative measure, which changes by $+1$ at an arrival, by $-1$ at a departure, but in addition each of these infinite virtual buffers has a nominal inflow rate $\alpha_k$ (which can be $= 0$, $> 0$, or $< 0$), such that $\frac{dQ_k(t)}{dt} = \alpha_k$ at all times $t$ at which there is no arrival or departure. The buffer levels $Q_k(t)$, $k \in K_\infty$ are not restricted to be non-negative.

Activities $j \in J$ are used to process the various items. Each application of an activity processes a fixed number of items from one or from several buffers for a certain amount of time and at the end of the processing the items are transformed into a new set of items. The $\ell$'s application of activity $j$ will seize $B_{kj}$, an integer $\geq 0$ number of items of class $k$, out of each of the buffers $k \in K$, and will process them for a duration $\eta_j(\ell)$. At the end of the processing these items will be converted into a new set of items, consisting of integer $\geq 0$ numbers $\phi_{kj}(\ell)$ of items of class $k$, which will go into each of the buffers $k \in K$. We denote by $T_j(t)$ the total time that activity $j$ has been applied, over the time interval $(0, t)$. We let $S_j(s)$ be the counting process of activity $j$ completions, that is $S_j(s) = \max\{n : \sum_{\ell=1}^{n} \eta_j(\ell) \leq s\}$. We let $\Phi_{kj}(n)$ be the cumulative of $\phi_{kj}(\ell)$, that is $\Phi_{kj}(n) = \sum_{\ell=1}^{n} \phi_{kj}(\ell)$.

The following equations describe the dynamics of the buffer levels:

\[
Q_k(t) = Q_k(0) + \alpha_k t - \sum_j \{S_j(T_j(t))B_{kj} - \Phi_{kj}(S_j(T_j(t)))\}, \quad k \in K_\infty \tag{3.5}
\]

\[
Q_k(t) = Q_k(0) - \sum_j \{S_j(T_j(t))B_{kj} - \Phi_{kj}(S_j(T_j(t)))\} \geq 0, \quad k \in K_0. \tag{3.6}
\]

Infinite virtual buffers generalize the usual notion of the outside world of a queueing network: In a standard queueing network we have a stream of exogenous arrivals $A_k(t)$ from the outside world into buffer $k$. We now consider this outside world as an infinite virtual buffer which is the source of these arrivals, and we assign to it an activity, which generates these arrivals. This is more general in several ways: We now control the arrivals from within the system, by assigning activities which generate the arrivals, including the routing of these arrivals to various nodes. We assume that the activity which generates arrivals consume resources,
and these resources may be shared with other activities. We have no exogenous inputs, but we have nominal flow rates. The buffer level $Q_k(t)$ for an infinite virtual buffer measures the backlog of items not yet departed, relative to $\alpha_k t$.

The application of activities requires resources. Let $i \in \mathcal{I}$ be the set of resources. We let $A_{ij} = 1$ if activity $j$ is using resource $i$, and $A_{ij} = 0$ otherwise. The capacity constraints of the resources are:

$$T_j(0) = 0, \quad T_j \text{ non-decreasing}, \quad \sum_j (T_j(t) - T_j(s))A_{ij} \leq t - s, \quad 0 \leq s < t, \quad i \in \mathcal{I} \quad (3.7)$$

Note that from these constraints it follows that $T_j(t)$ are Lipschitz continuous with constant 1, and therefore have a derivative almost everywhere. We call time points $t$ at which a derivative exists regular $t$, and whenever we talk about the derivative $\dot{T}(t)$ we assume that $t$ is a regular time.

Additional restrictions relate to the way in which we apply the activities. We assume that when an activity starts, it seizes the items which it needs to process, and those items are now ‘in process’. If another activity starts while some items are in process, it will seize a new set of items, which will be ‘in process’. To apply any activity, the various buffers must contain a sufficient number of items which are not in process, to start the new activity. There are always enough items if the buffer is virtually infinite. Otherwise, if there are not enough items in the buffer the activity cannot be applied. We assume that each activity is applied only once at a time: The $\ell + 1$ application of activity $j$ cannot start until the $\ell$’s application of activity $j$ is complete. We say that an activity $j$ is available at time $t$ if it is not ‘in process’, and if all the buffers contain at least $B_{kj}$ items which are not in process. We assume that $\mathcal{K}, \mathcal{J}$ are finite. We define:

$$D = \max_{k \in \mathcal{K}_0} \sum_{j \in \mathcal{J}} B_{kj}. \quad (3.8)$$

If all the standard buffers contain at least $D$ items, then every activity which is not in process is available.

If several activities which use the same resource are in process simultaneously then for each of them $0 \leq \hat{T}_j(t) \leq 1$, and as a result processing proceeds at a $\hat{T}_j(t)$ fraction of the speed. This is called processor splitting. In most of this paper we do not allow processor splitting. This is enforced by adding the requirement that $\hat{T}_j(t) \in \{0, 1\}$. An activity which is in process and has $\hat{T}_j(t) = 0$ is being preempted. In most of this paper we also do not allow preemptions. This can be enforced by requiring that $\hat{T}_j(t)$ can only decrease when $S_j(T_j(t))$ increases.

We make the following probabilistic assumptions about the processing times and the sets of items that come out of the processing, and define quantities $R_{kj}$:

$$\lim_{t \to \infty} S_j(t, \omega)/t = 1/m_j, \quad \lim_{n \to \infty} \Phi_{kj}(n, \omega)/n = v_{kj}, \quad R_{kj} = (B_{kj} - v_{kj})/m_j \quad (3.9)$$

Here $m_j$ is the average time needed for application of activity $j$, and $v_{kj}$ is the average number of items of class $k$ produced by an application of activity $j$. The quantities $R_{kj}$ describe the long term average rate at which employing activity $j$ depletes buffer $k$. We refer to $R$ as the
input output matrix of the process. We also use \( \mu_j = 1/m_j \) to denote the long term average processing rate.

The evolution of the system is summarized by the stochastic network process which we define as \((Q(t), T(t))\). The determination of \( T(t) \) constitutes the policy.

Given the parameters of the network, \( \alpha, \mu, R, A \) the utilization \( \rho \) of the network is determined by the following linear program, which is a slight modification of the static planning problem \( \text{LP} \) introduced by Harrison [35].

\[
\begin{align*}
\min \quad & \rho \\
\text{s.t.} \quad & Ru = \alpha, \\
& Au \leq 1 \rho, \\
& u \geq 0.
\end{align*}
\] (3.10)

Unfortunately we are not yet able to derive results for general processing networks, and in this paper we will restrict attention to processing networks with special structure. We list here the definitions (see Dai and Lin [19]) of these special structures.

**Definition 3.1** A processing network is called **strictly Leontief** if each activity depletes exactly one buffer, equivalently, each column of \( R \) has a single positive entry.

A processing network is called **reversed Leontief** if each activity uses exactly one resource, equivalently, each column of \( A \) is a unit vector.

A processing network is called **unitary** if it is both strictly Leontief and reversed Leontief.

A processing network is a multi-class queueing network (MCQN) if each buffer/class has a single activity which processes items of this class, items are processed singly, and a processed item either leaves the system or is routed to another buffer. In this case we denote both buffers and activities by \( k \in K \), and \( \phi_{k'k}(\ell) = 1 \) if the processed item of class \( k \) is routed to buffer \( k' \) \((k' \neq k)\) and \( \phi_{k'k}(\ell) = 0 \) otherwise. The input output matrix is then of the form \( R = I - P' \) where \( P' \) is a substochastic matrix in which \( P_{kk'} = \lim_{n \to \infty} \Phi_{k'k}(n, \omega)/n \).

A processing network is a MCQN with alternative routing if each buffer/class has one or more activities which processes items of this class, items are processed singly, and a processed item either leaves the system or is routed to another buffer. Here each column of the input output matrix is of the form as for MCQN, but each class may have several columns.

A processing network with virtual infinite buffers is draining if no items are routed into infinite virtual buffers, that is for \( k \in K and all \( j, \ell, \phi_{kj}(\ell) = 0 \). Row \( k \) of \( R \) with \( k \in K has non-negative entries.

**A comment on probabilistic assumptions and on the concept of online policies**

In problems of deterministic scheduling one often assumes all processing times are known and one can choose any order to perform tasks. In that case ordering tasks by shortest processing time first (SPT) minimizes total waiting time. At the other extreme, one assumes that one cannot order jobs according to processing time. For example, maybe jobs are ordered in line and we are required to perform them in that order, which is the case for online scheduling.
In this case we cannot assume that the jobs are ordered favorably. Thus in online scheduling one often looks at the ‘competitive ratio’ of a scheduling rule — the performance of the rule for unfavorable orders. In many applications it may be the case that tasks are performed in an order which is neither favorable nor disfavorable. The probabilistic assumption (3.9) says that the ordering is neutral, so that in the long run we do not give preference to shorter tasks over longer tasks or otherwise differentiate tasks according to their processing times.

The optimization in this paper is within the confines of treating items in each class as indistinguishable, or of scheduling them online, so that (3.9) holds. Sometimes one can improve the control by refining the classification. Our asymptotics here assume that such a refinement has already been done and we have a fixed number of classes, with the number of items in each class becoming large.

To be more concrete, our policies determines $T_j(t)$, but given that $T_j(t) = s$, the number of completions, $S_j(s)$ is not influenced by our policy, and will satisfy $S_j(s)/s \to \mu_j$ and $\Phi_{kj}(n)/n \to v_{kj}, k \in K$. In particular, this means that if $T_j(t) \to \infty$ then $S_j(T_j(t))/T_j(t) \to \mu_j$ and $\Phi_{kj}(S_j(T_j(t)))/S_j(T_j(t)) \to v_{kj}, k \in K$.

### 3.3 Maximum pressure policies and stability

The search for policies which can keep systems with $\rho \leq 1$ stable has been going on for the last ten years. Notably Bramson [12] has shown that FIFO is stable for Kelly networks, and Head of the Line Processor Sharing is stable for general multi-class queueing networks. Recently Dai and Lin [19] introduced Maximum pressure policies. These are special for two reasons: As Dai and Lin show, they are stable for general processing networks with $\rho \leq 1$ if the networks satisfy some structural condition, and it seems that under heavy traffic conditions they are optimal on a diffusion scale. Similar policies were proposed by Tassiulas [75] and Stolyar [74]. In this section we introduce maximum pressure policies for processing networks with infinite virtual buffers and nominal flow rates, and discuss their stability when $\rho \leq 1$. As it turns out, the proofs of Dai and Lin’s results go over without any change for this more general situation.

Let the set of feasible allocations $A$ be defined as the vectors $u$ such that $\sum_{j \in J} A_{ij} u_j \leq 1$, $i \in I$, $u_j \geq 0$, $j \in J$. Let $q$ be a vector of buffer levels (queue lengths, though not necessarily integer), that is $q_k$ real $\geq 0$, $k \in K_0$, $q_k$ real, $k \in K_\infty$. We refer to $q$ as a network state. For an allocation $u \in A$ and a network state $q$ define the total network pressure to be: $p(u, q) = q'R u$ where $'$ denotes the transpose.

To calculate the maximum pressure policy at time $t$, consider the current system state, and calculate

$$u^* = \arg \max_{u \in A} p(u, Q(t)) \tag{3.11}$$

The allocation $u^*$ may not be feasible. The allocation $u$ is available at time $t$, if each of the buffers $k \in K_0$ contains enough items. Here one needs to exclude items which are in process (by activities which are progressing or activities which are preempted), and the remaining items in buffer $k$ must exceed in number $\sum_{j: u_j > 0} B_{kj}$. Note that the buffers $k \in K_\infty$ always have enough items available for any activity. Also if a buffer has at least $D$ items then it has enough items for any allocation. If an allocation $u$ is not available at time $t$, it cannot be used. Denote by $A(t)$ the set of feasible allocations which are available at time $t$. 13
The set of all the feasible allocations $\mathcal{A}$ is a non-empty (contains 0), bounded (all $u_j \leq 1$) convex polytope, and has a finite number of extreme points. Hence the maximum in (3.11) exists, and it is always possible to choose $u^*$ as an extreme point of $\mathcal{A}$. Denote by $\mathcal{E}$ the set of extreme points of $\mathcal{A}$, and call such allocations extreme allocations.

On the other hand, $\mathcal{A}(t)$ is not a convex set: It is the union of faces of $\mathcal{A}$, where each face is obtained by imposing additional constraints of the form $u_j = 0$, $j \in \bar{J}$, where $\bar{J}$ is a set of activities such the $J \setminus \bar{J}$ is available, but no additional activity is available. Note that all the extreme points of a face of $\mathcal{A}$ are also extreme points of $\mathcal{A}$. Each face is again a convex polytope, and so there exists an extreme point at which the maximum pressure on the face is obtained. Therefore there exists an extreme point of $\mathcal{A}$ at which the maxima pressure on the set $\mathcal{A}(t)$ is obtained.

Let $\mathcal{E}(t) = \mathcal{E} \cap \mathcal{A}(t)$ denote the set of the extreme allocations which are available at time $t$. As we have explained, $\max_{u \in \mathcal{E}(t)} p(u, Q(t)) = \max_{u \in \mathcal{A}(t)} p(u, Q(t))$. We now define the maximum pressure policy:

**Definition 3.2** A policy is said to be a maximum pressure policy if at each time $t$ it chooses an allocation $u^0 = \arg\max_{u \in \mathcal{E}(t)} p(u, Q(t))$.

**Note** The maximum pressure policy of Definition 3.2 allows processor splitting and preemptions, subject to the rule that items in process (including preempted items) cannot be reallocated, and each activity has to complete application $\ell$ before starting application $\ell + 1$.

For an allocation $u$ buffer $k$ is a constituent buffer if there exists $j$ such the $u_j B_{kj} > 0$.

**Assumption 3.3 (EAA assumption)** A processing network satisfies the Extreme Allocation Available assumption if for every state vector $q$ there exists an extreme allocation $u^* \in \mathcal{E}$ such that it maximizes the network pressure, i.e. $u^* = \arg\max_{u \in \mathcal{E}} p(u, z)$, and such that for each constituent buffer $k \in K_0$, $q_k > 0$.

**Definition 3.4** A stochastic processing network operating under some general policy is said to be pathwise stable if for every initial state, with probability one,\\\[ \lim_{t \to \infty} Q_k(t)/t = 0, \quad k \in K. \] (3.12)

**Theorem 3.5** Consider a processing network, with infinite virtual buffers and nominal flows $\alpha$. Assume that the optimal solution of the static planning problem LP (3.10) for this $\alpha$ has utilization $\rho \leq 1$. Assume further that the processing network satisfies the EAA assumption. Then operating this network under maximum pressure policy (with preemptions and processor splitting), is pathwise stable.

The proof of Dai and Lin carries through for this case with no change. To prove the theorem they consider fluid limits and show that every fluid limit is weakly stable: if the fluid limit buffer levels at time $t_0$ are all 0 then the fluid limit will remain 0 for all $t > t_0$. This property carries over to networks with infinite virtual buffers.

Dai and Lin [19] proceed to find processing networks which satisfy the EAA property. They show that a standard processing network which is strictly Leontief, satisfies EAA. Unfortunately this does not carry through for processing networks with infinite virtual buffers, unless one makes an additional assumption:
Theorem 3.6 Assume that a processing network with infinite virtual buffers is strictly Leontief. Assume in addition that the network is draining, i.e. no items are routed into any of the infinite virtual buffers. Then this network satisfies the EAA assumption.

Theorem 3.7 If the processing network is reversed Leontief, then every extreme allocation is integer. Hence, the maximum pressure policy does not split processors.

Furthermore, if a network is reversed Leontief, then maximum pressure over $\mathcal{A}$ is found by maximizing the pressure separately for each resource (it is separable). Finally:

Theorem 3.8 If a network satisfies EAA condition and is reversed Leontief, and if in addition there exist $m'_j < \infty, \epsilon_j > 0, j \in \mathcal{J}$ such that:

$$\lim_{n \to \infty} \frac{1}{n} \sum_{\ell=1}^{n} \eta_j(\ell)^{1+\epsilon_j} = m'_j$$

almost surely (3.13)

then maximum pressure policy with no preemptions is stable when $\rho \leq 1$.

Recall that unitary networks, which include all multiclass queueing networks and MCQN with alternative routing are both strictly Leontief and reversed Leontief. Thus if the are draining, have utilization $\rho \leq 1$, and satisfy (3.13), then they are stable under max pressure policies with no processor splitting and no preemptions.

4 Fluid control of multi-class queueing networks

In this section we describe our fluid approach as it is applied to a multiclass queueing network. This includes

- Definition of the finite horizon multi-class queueing network problem.
- Formulation of the fluid approximation and description of the fluid solution.
- Tracking of the fluid solution by a fluid tracking max pressure policy.

4.1 The multiclass queueing network finite horizon problem

We assume that our processing network is a multi-class queueing network as in Definition 3.1. For $k \in \mathcal{K}$ we let $k$ label a class/queue/buffer, as well as the activity which is processing items of that class. Partition $k \in \mathcal{K}$ into $k \in \mathcal{C}_i$, where $\mathcal{C}_i$, the constituency of machine $i$, are the activities which require machine $i \in \mathcal{I}$. The $\ell$s application of activity $k$ will process a single item of buffer $k$ for a duration $\eta_k(\ell)$, at the end of which it will be converted into an item of class $k' \neq k$, so that $\phi_{kk'}(\ell) = 1$ and $\phi_{jk}(\ell) = 0, j \neq k'$, or it will leave the system so that $\phi_{jk}(\ell) = 0, j \in \mathcal{K}$. The dynamics of the queueing network are:

$$Q_k(t) = Q_k(0) - S_k(T_k(t)) + \sum_{k'} \Phi_{kk'}(S_{k'}(T_{k'}(t))) \geq 0, \quad k \in \mathcal{K}.$$ (4.1)
The constraints on the allocation of time, including the machine capacity constraints are:

\[ T_k(0) = 0, \quad T_k \text{ non-decreasing}, \quad \sum_{k \in C_i} (T_k(t) - T_k(s)) \leq t - s, \quad 0 \leq s < t, \quad i \in I \quad (4.2) \]

In addition we do not allow processor splitting, i.e. \( T_k(t) \in \{0,1\} \) and we do not allow preemption, i.e. \( T_k(t) \) can decrease only when \( S_k(T_k(t)) \) increases.

We wish to control this system over a finite time horizon, \( 0 < t < T \). In (4.1) there is no input: We assume that all the items to be processed within the time horizon are present in the system initially. Our objective is to minimize total operation and holding costs given by:

\[ \sum_{k \in K} \gamma_k T_k(T) + \sum_{k \in K} c_k \int_0^T Q_k(t) dt. \quad (4.3) \]

We make the following probabilistic assumptions which parametrize the system. Almost surely for \( \omega \):

\[ \lim_{t \to \infty} S_k(t, \omega)/t = \mu_j, \quad \lim_{n \to \infty} \Phi_k(n, \omega)/n = P_{kk'}. \quad (4.4) \]

We assume that \( P \) has spectral radius < 1, so that \( I - P' \) has an inverse (this corresponds to a finite long term average number of processing steps per item).

We further assume that the sequences of processing times satisfy the higher moment probabilistic assumption (3.13).

The input output matrix \( R \) and the resource consumption matrix \( A \) for this multiclass queueing network are:

\[ R_{k'k} = \begin{cases} \mu_k & \text{if } k' = k \\ -P_{kk'} & \text{if } k' \neq k \end{cases}, \quad \text{equivalently } R = (I - P') \text{diag}(\mu), \quad (4.5) \]

\[ A_{ik} = \begin{cases} 1 & \text{if } k \in C_i \\ 0 & \text{else} \end{cases}. \quad (4.6) \]

### 4.2 The fluid model and the fluid solution

The fluid model for the multiclass queueing network has the dynamics:

\[ q_k(t) = q_k(0) - \mu_k \int_0^t u_k(s) ds + \sum_{k' \neq k} P_{k'k} \mu_{k'} \int_0^t u_{k'}(s) ds \quad (4.7) \]

where \( u_k(t) \) is allocation of processing rate to activity \( k \) at time \( t \), so that \( \int_0^t u_k(s) ds \) is the total fluid time allocated to activity \( k \) over \((0,t)\).

The processing rate allocations are subject to:

\[ \sum_{k \in C_i} u_k(t) \leq 1, \quad i \in I, \quad u_k(t) \geq 0 \quad k \in K \quad (4.8) \]
And the fluid objective is:

$$\min \int_0^T \sum_k (\gamma_k u_k(t) + c_k q_k(t)) \, dt \quad (4.9)$$

This is summarized as the following fluid optimization problem, which is a separated continuous linear program (SCLP):

$$V^* = \min \int_0^T (\gamma' u(t) + c' q(t)) \, dt$$

s.t. $\int_0^t R u(s) \, ds + q(t) = q(0)$$

$$Au(t) \leq 1$$

$$u(t) \geq 0, q(t) \geq 0$$

A basic optimal solution to this SCLP will be given by a partition of the time horizon into intervals $0 = t_0 < t_1 < \cdots < t_M = T$ where in each of these intervals the controls $u$ are constant. We let $u_k^m = u_k(t), t_{m-1} < t < t_m$ denote the constant controls in the $m$th interval. The fluid levels in the various buffers will be non-negative continuous piecewise linear functions of $t$, where the slopes $\dot{q}$ in each of the intervals are constant. We let $\dot{q}_k^m = \frac{d}{dt} q_k(t), t_{m-1} < t < t_m$ denote the constant slopes in the $m$th interval. The values $u_k^m, \dot{q}_k^m$ will satisfy:

$$Ru_k^m + \dot{q}_k^m = 0$$

$$Au_k^m \leq 1$$

$$u_k^m \geq 0.$$ 

In the $m$th interval of the solution we partition the buffers into the set $k \in \mathcal{K}^m_\infty$ of non-empty fluid buffers with $q_k(t) > 0, t_{m-1} < t < t_m$, and the set $k \in \mathcal{K}^m_0 = \mathcal{K} \setminus \mathcal{K}^m_\infty$ of empty fluid buffers with $q_k(t) = 0, t_{m-1} < t < t_m$. Clearly for $k \in \mathcal{K}^m_0$ we have $\dot{q}_k^m = 0$. Partition the input output matrix $R$ into submatrices: $R_{\mathcal{K}^m_0, \mathcal{K}^m_0}, R_{\mathcal{K}^m_0, \mathcal{K}^m_\infty}, R_{\mathcal{K}^m_\infty, \mathcal{K}^m_0}, R_{\mathcal{K}^m_\infty, \mathcal{K}^m_\infty}$. We can then write:

$$R_{\mathcal{K}^m_0, \mathcal{K}^m_0} u_{\mathcal{K}^m_0}^m + R_{\mathcal{K}^m_0, \mathcal{K}^m_\infty} u_{\mathcal{K}^m_\infty}^m = 0$$

$$R_{\mathcal{K}^m_\infty, \mathcal{K}^m_0} u_{\mathcal{K}^m_0}^m + R_{\mathcal{K}^m_\infty, \mathcal{K}^m_\infty} u_{\mathcal{K}^m_\infty}^m = -\dot{q}_{\mathcal{K}^m_\infty}^m$$

from which we can express $u_{\mathcal{K}^m_0}^m, \dot{q}_{\mathcal{K}^m_\infty}^m$ in terms of $u_{\mathcal{K}^m_\infty}^m$:

$$u_{\mathcal{K}^m_0}^m = -\left( R_{\mathcal{K}^m_0, \mathcal{K}^m_0} \right)^{-1} R_{\mathcal{K}^m_0, \mathcal{K}^m_\infty} u_{\mathcal{K}^m_\infty}^m$$

$$\dot{q}_{\mathcal{K}^m_\infty}^m = \left( R_{\mathcal{K}^m_\infty, \mathcal{K}^m_0} \left( R_{\mathcal{K}^m_0, \mathcal{K}^m_0} \right)^{-1} R_{\mathcal{K}^m_0, \mathcal{K}^m_\infty} - R_{\mathcal{K}^m_\infty, \mathcal{K}^m_\infty} \right)^{-1} u_{\mathcal{K}^m_\infty}^m$$

Recall that $P$ has spectral radius less than 1, hence $(I - P')^{-1}$ is well defined, and so is $R^{-1}$ and $(R_{\mathcal{K}^m_0, \mathcal{K}^m_0})^{-1}$. 

17
4.3 Tracking the fluid solution by maximum pressure policy

Tracking of a fluid solution can be done in many ways. Maglaras [55, 56] describes such methods for some quite general queueing networks. In the present paper we suggest a new way of tracking the fluid solution obtained in Section 4.2, which is based on the maximum pressure policy of Dai and Lin [19].

To track the fluid solution we use a maximum pressure policy, which is specially tailored for each of the intervals of the fluid solution. Given the values \( Q_k(t_{m-1}) \geq 0 \) at the beginning of the \( m \)th interval, we define for \( t_{m-1} < t < t_m \) a residual process \( \tilde{Q}(t) \) with the following features:

The process \( \tilde{Q}(t) \) is coupled with \( Q(t) \) in that they share the same processes \( S_k(T_k(t)) \) as well as the same time allocation \( T_k(t) \), for \( k \in \mathcal{K}, k' \in \mathcal{K}_0^m \).

The process \( \tilde{Q}(t) \) describes the queue lengths of a processing network with infinite virtual buffers

The process \( \tilde{Q}(t) \) is controlled by a maximum pressure policy.

We define:

\[
\tilde{Q}_k(t) = \begin{cases} 
Q_k(t), & k \in \mathcal{K}_0^m \\
\mu_k u^m_k(t - t_{m-1}) - (S_k(T_k(t)) - S_k(T_k(t_{m-1}))), & k \in \mathcal{K}_\infty^m, \\
0 & k \in \mathcal{K}_0^m, t_{m-1} < t < t_m.
\end{cases}
\]

(4.14)

Note the differences between the processes \( Q(t) \) and \( \tilde{Q}(t) \):

The two processes are identical on the \( k \in \mathcal{K}_0^m \) buffers.

The initial values at the time \( t_{m-1} \) are \( Q_k(t_{m-1}) \geq 0, \tilde{Q}_k(t_{m-1}) = 0, k \in \mathcal{K}_\infty^m \).

The process \( Q_k(t) \) has no exogenous inputs, while the process \( \tilde{Q}_k(t) \) has nominal inputs of rate \( \mu_k u^m_k, k \in \mathcal{K}_\infty^m \).

For the \( \ell \) application of activity \( k \), if \( \phi_{k'k}(\ell) = 1 \) for some \( k' \in \mathcal{K}_\infty^m \) then an item is moved from buffer \( k \) to \( k' \) in the process \( Q(t) \), whereas this item will leave the system in \( \tilde{Q}(t) \).

In other words, no items are routed into \( k' \in \mathcal{K}_\infty^m \) in \( \tilde{Q} \).

It is seen from its dynamics that \( \tilde{Q}(t) \) in the interval \( t_{m-1} < t < t_m \) describes a multi-class queueing network with infinite virtual buffers \( \mathcal{K}_\infty^m \). The nominal inflows for the infinite virtual buffers are \( \alpha_k = \mu_k u^m_k \). The input output matrix of \( \tilde{Q}(t) \) in the \( m \)th interval is:

\[
\mathcal{R}_{k'k}^m = \begin{cases} 
R_{kk} & k' = k, \\
R_{k'k} & k' \neq k, k' \in \mathcal{K}_0^m \\
0 & k' \neq k, k' \in \mathcal{K}_\infty^m
\end{cases} = \begin{cases} 
\mu_k & k' = k, \\
-\mu_k P_{kk'} & k' \neq k, k' \in \mathcal{K}_0^m \\
0 & k' \neq k, k' \in \mathcal{K}_\infty^m
\end{cases}.
\]

(4.15)

It is immediately seen by its definition that \( \tilde{Q}(t) \) is a draining network as in Definition 3.1.
We use \( Q(t), \tilde{Q}(t) \) to define a maximum pressure policy. The pressure is defined according to the process \( \tilde{Q}(t) \) and matrix \( \tilde{R}^m \), and for allocation \( u \) it is \( \tilde{Q}(t)' \tilde{R}^m u \). The maximum pressure allocation for \( t_{m-1} < t < t_m \) is the \( u \) which solves:

\[
\begin{align*}
\max & \quad \tilde{Q}(t)' \tilde{R}^m u \\
\text{s.t.} & \quad Au \leq 1 \\
& \quad u \geq 0.
\end{align*}
\] (4.16)

Because this is a unitary processing network, the maximization is separable, and maximum pressure for machine \( i \) is

\[
\max \{0, \mu_k \left[ \tilde{Q}_k - \sum_{k' \in K^m_0} P_{kk'} \tilde{Q}_{k'} \right] : k \in C_i \}
\] (4.17)

However, when we choose the maximizing allocation we only choose to process items for which \( Q_k(t) \) is positive.

**Definition 4.1 (Fluid Tracking Maximum Pressure Policy)**  With no processor splitting and no preemptions, whenever machine \( i \) is available at time \( t \), and the state and residual state are \( Q, \tilde{Q} \), calculate:

\[
\max \{0, \mu_k \left[ \tilde{Q}_k - \sum_{k' \in K^m_0} P_{kk'} \tilde{Q}_{k'} \right] : k \in C_i \text{ and } Q_k(t) > 0 \}
\] (4.18)

and choose activity \( k \) which is a maximizing activity. If the maximum is 0, idle machine \( i \).

### 5  Asymptotic optimality of the fluid approach

We now prove asymptotic optimality of the fluid approach described in the last section.

We consider parameters \( A, P, \mu, R = (I - P') \text{diag}(\mu), \gamma, c \), initial fluid vector of integers \( q(0) \), and time horizon \( T \). We denote by \((q(t), u(t), 0 < t < T, V^*)\) the resource allocation rates, fluid buffer levels, and objective value of the optimal fluid solution of (4.10). Recall that this solution is given by the partition of the time horizon \( 0 = t_0 < t_1 < \cdots < t_M = T \), and by the constant rates \( u^m, \tilde{q}^m \) in the intervals \((t_{m-1}, t_m), m = 1, \ldots, M\).

We consider a probability space with the processing and routing sequences \( S_k(s, \omega), 0 < s < \infty, \text{ and } \Phi_{k'(n, \omega)}, n = 1, 2, \ldots, \) which satisfy (4.4, 3.13) almost surely for \( \omega \).

We consider a sequence of systems, indexed by \( N = 1, 2, \ldots, \) which are defined as follows: The initial state of system \( N \) is \( Q^N(0) = Nq(0) \), the processing sequences are \( S_k^N(s) = S_k(Ns) \), the processing cost rates are \( \gamma^N = N\gamma \), and the systems share the same routing sequences \( \Phi_{k'(n)} \), the same resource consumption matrix \( A \), and the same holding cost rates \( c \). It follows that the remaining parameters of the \( N \)th system are \( \mu^N = N\mu, P^N = P, R^N = NR \). We denote by \((Q^N(t, \omega), T^N(t, \omega), 0 \leq t \leq T, V(\omega)^N)\) the buffer levels, time allocations and
objective value for the $N$th system. We use a superscript $N$ to denote various other quantities of the $N$th system.

For this sequence of systems we prove the following theorem. The proof follows closely the ideas and steps of Dai and Lin [19];

**Theorem 5.1** (i) For any policy, almost surely:

$$\lim_{N \to \infty} \inf_{\omega} \frac{V^N(\omega)}{N} \geq V^*$$  \hspace{1cm} (5.1)

(ii) Under the fluid tracking maximum pressure policy, almost surely for $\omega$ (uniformly on $t \in (0, T)$),

$$\lim_{N \to \infty} Q^N(t, \omega)/N = q(t), \quad \lim_{N \to \infty} \frac{V^N(\omega)}{N} = V^*$$  \hspace{1cm} (5.2)

**Proof.** (i) Consider the sequence of systems under some arbitrary fixed policy. We will drop $\omega$ from the notation when we consider a fixed sample path.

We say that $(\bar{Q}(t), \bar{T}(t), 0 \leq t \leq T, \bar{V})$ is a fluid limit if for some $\omega$ which satisfies (4.4, 3.13), and for a subsequence of indexes $r \to \infty$,

$$\bar{Q}(t), \bar{T}(t), \bar{V} = \lim_{r \to \infty} \left( \frac{Q^r(t, \omega)}{r}, T^r(t, \omega), \frac{V^r(\omega)}{r} \right), \text{ uniformly on } 0 \leq t \leq T$$

Consider now a fixed sample path $\omega$ which satisfies (4.4). Let $T^N(t), 0 \leq t \leq T$ be the sequence of time allocations for that $\omega$ under the arbitrary fixed policy. By $T^N_k(t) - T^N_k(s) < t - s$, $0 \leq s < t \leq T$ the family of functions $T^N_k$ are equicontinuous, and we can find a subsequence $r$ so that $T^r(t)$ converges uniformly for $0 \leq t \leq T$ as $r \to \infty$. Let $\bar{T}(t) = \lim_{r \to \infty} T^r(t)$. Denote by $\hat{T}(t)$ its derivative which exists almost everywhere, so that we can write $\bar{T}(t) = \int_0^t \hat{T}(s)ds$. By (4.4)

$$\lim_{r \to \infty} \frac{S_k^r(T_k^r(t))}{r} = \lim_{r \to \infty} \frac{S_k(rT_k^r(t))}{r} = \mu_k \bar{T}_k(t), \text{ uniformly on } 0 \leq t \leq T$$

$$\lim_{r \to \infty} \frac{\Phi_{kk'}(S_k^r(T_k^r(t)))}{r} = \lim_{r \to \infty} \frac{\Phi_{kk'}(S_k(rT_k^r(t)))}{r} = \mu_{kk'} \bar{T}(t), \text{ uniformly on } 0 \leq t \leq T$$

and so:

$$\lim_{r \to \infty} \frac{Q_k^r(t)}{r} = \lim_{r \to \infty} \left[ \frac{Q_k^r(0)}{r} - \frac{S_k^r(T_k^r(t))}{r} + \sum_{k'} \frac{\Phi_{kk'}(S_{k'}^r(T_{k'}^r(t)))}{r} \right]$$

$$= q_k(0) - \mu_k \bar{T}_k(t) + \sum_{k'} \mu_{kk'} P_{kk'} \bar{T}(t) \text{ uniformly on } 0 \leq t \leq T$$

Hence, almost surely for all $\omega$ fluid limits exist, and every fluid limit $\bar{Q}(t), \bar{T}(t), 0 \leq t \leq T, \bar{V}$ satisfies:

$$\bar{V} = \int_0^t \left( \gamma \hat{T}(t) + c' \bar{Q}(t) \right) dt$$
\[ \bar{Q}(t) = q(0) - \int_0^t R\dot{T}(s)ds \]  
\[ A\dot{T}(t) \leq 1, \]  
\[ \dot{T}(t), \bar{Q}(t) \geq 0, \quad 0 \leq t \leq T \]  

Note in particular that every fluid limit has \( \bar{Q}(t) \) continuous (in fact Lipschits continuous with constant obtained from \( R \)).

Comparing with (4.10) we have that for every fluid limit, \( \bar{V} \geq V^* \).

Consider then \( \bar{V} = \liminf_{N \to \infty} \frac{V^N(\omega)}{N} \) for some \( \omega \). Then for some subsequence \( r \), \( \bar{V} = \lim_{r \to \infty} \frac{V^r(\omega)}{r} \). Almost surely for the sample path \( \omega \), (4.4) holds. We can then find a subsequence of \( r, r' \), for which \( \lim_{r' \to \infty} (\bar{Q}(t), \bar{T}(t), \bar{V}) = \lim_{r \to \infty} (\frac{Q^r(t)}{r'}, T^r(t), \frac{V^r}{r'}) \), uniformly on \( 0 \leq t \leq T \). But then we must have \( \bar{V} \geq V^* \).

(ii) We now limit attention to the fluid tracking maximum pressure policy. The fluid problem for the \( N \)th system is

\[
V^N = \min \int_0^T \left( N\gamma' u(t) + c'q^N(t) \right) dt \\
\text{s.t.} \int_0^T NRu(s) ds + q^N(t) = Nq(0) \\
Au(t) \leq 1 \\
u(t) \geq 0, q^N(t) \geq 0
\]

Hence as is seen immediately, the fluid solution for the \( N \)th system is \((Nq(t), u(t), NV^*)\), and all the systems share the same partition \(0 = t_0 < t_1 < \cdots < t_M = T\), and the same sets of non-empty and empty buffers \( K^0, K^0\).

Our main task is to show that under this policy every fluid limit is in fact the solution of the fluid problem,

\[
(\bar{Q}(t), \bar{T}(t)) = (q(t), \int_0^t u(s)ds). \tag{5.4}
\]

We consider a fixed sample path \( \omega \) which satisfies (4.4, 3.13), and we let \( r \) be a subsequence of \( N \) such that \( \lim_{r \to \infty} (Q^r(t)/r, T^r(t)) = (\bar{Q}(t), \bar{T}(t)) \). We can now take a subsequence of \( r, r' \), such that \( \lim_{r' \to \infty} \bar{Q}^{r'}(t)/r' \) exists, and we denote it by \( \bar{Q}(t) \). Without loss of generality we can assume \( r \) is chosen so that \( \lim_{r \to \infty} (Q^r(t)/r, \bar{Q}^{r'}(t)/r, T^r(t)) = (\bar{Q}(t), \bar{Q}(t), \bar{T}(t)) \). We will prove (5.4) by showing also that

\[
\bar{Q}(t) = 0, \quad 0 \leq t \leq T. \tag{5.5}
\]

We prove (5.4, 5.5) by induction on \( m \). Assume as the induction hypothesis that \( (Q(t), Q(t), T(t)) = (q(t), 0, \int_0^t u(s)ds) \). This is obviously true for \( m = 0 \). We then show that \((Q(t), Q(t), T(t)) = (q(t), 0, \int_0^t u(s)ds), t_{m-1} < t < t_m \), from which by continuity \((Q(t), Q(t), T(t)) = (q(t), 0, \int_0^t u(s)ds) \).
Define
\[ \hat{t} = \min \{ t_m, \inf \{ t : t_{m-1} < t \leq t_m, \, \hat{Q}(t) = 0 \text{ for some } k \in K_\infty^m \} \}. \]

By continuity, \( \hat{t} > t_{m-1} \). Fix \( \tilde{t} \) such that \( t_{m-1} < \tilde{t} < \hat{t} \). We can find \( N_0 \) such that for all \( r > N_0 \) we have \( Q_k(t) > 0 \) for all \( t_{m-1} < t \leq \tilde{t} \) and for all \( k \in K_\infty^m \). Without loss of generality we assume that \( r > N_0 \). Because \( Q_k(t) > 0 \), \( t_{m-1} \leq t \leq \tilde{t} \), we have that the fluid tracking maximum pressure policy in the interval \( t_{m-1} \leq t \leq \tilde{t} \) is simply the maximum pressure policy of the process \( \hat{Q}^r(\cdot) \).

Let us now consider the sequence of processes \( \tilde{z}_N^N(s) = \tilde{Q}(s/t), 0 < s < NT \). In each of the periods \( Nt_{m-1} < s < Nt_m, m = 1, \ldots, N \) it describes the buffer levels of a multi class queueing network with infinite virtual buffers. In the period \( Nt_{m-1} < s < Nt_m \) the set of infinite virtual buffers is \( K_\infty^m \), the nominal inflows are \( \alpha_k^m = \mu_k u_k^m, k \in K_\infty^m, \alpha_k^m = 0, k \in K_0^m \), and the input output matrix of the network is \( \hat{R}^m \).

As a multiclass queueing network this network is unitary according to the Definition 3.1 (each column of \( R^m \) has only a single positive entry, and every column of \( A \) is a unit vector). Also this network has 0 feedback into any of the infinite virtual buffers, and is therefore strictly draining according to Definition 3.1. By Theorem 3.6 the network \( \tilde{z}_N^N(s) \) satisfies EAA assumption for each of the intervals \( Nt_{m-1} < s < Nt_m \).

We now calculate the utilization of \( \tilde{z}_N^N(\cdot) \), in the interval \( Nt_{m-1} \leq s \leq Nt_m \). The nominal inflow rate is \( \alpha^m \) and the input output matrix \( \hat{R}^m \) is according to (4.15):
\[
\hat{R}^m = \begin{bmatrix} R_{K_0^m, K_0^m} & R_{K_0^m, K_\infty^m} \\ 0 & \text{diag}(\mu_{K_\infty^m}) \end{bmatrix}
\]  
(5.6)

The fluid solution, according to (4.11,4.12), satisfies:
\[
\begin{bmatrix} R_{K_0^m, K_0^m} & R_{K_0^m, K_\infty^m} \\ R_{K_\infty^m, K_0^m} & R_{K_\infty^m, K_\infty^m} \end{bmatrix} \begin{bmatrix} u_{K_0^m}^m \\ u_{K_\infty^m}^m \end{bmatrix} + \begin{bmatrix} \tilde{u}_{K_0^m}^m \\ \tilde{u}_{K_\infty^m}^m \end{bmatrix} = 0
\]
(5.7)

where \( \tilde{u}_{K_\infty^m}^m = 0 \). Hence, using (5.6,5.7), in the \( m \)th interval we have:
\[
\hat{R}^m u^m = \begin{bmatrix} R_{K_0^m, K_0^m} & R_{K_0^m, K_\infty^m} \\ 0 & \text{diag}(\mu_{K_\infty^m}) \end{bmatrix} \begin{bmatrix} u_{K_0^m}^m \\ u_{K_\infty^m}^m \end{bmatrix} = \begin{bmatrix} 0 \\ \mu_k u_k^m \end{bmatrix} \quad k \in K_\infty^m
\]
(5.8)

The right hand side here is exactly the nominal inflow of \( \tilde{z}_N^N(\cdot) \) in the \( m \)th interval, \( \alpha^m \), so:
\[
\hat{R}^m u^m = \alpha^m
\]
(5.9)

and by comparing (5.9) with the static planning problem (3.10) we see that the utilization is \( \rho \leq 1 \).
Hence all the necessary conditions of Theorems 3.5, 3.6, 3.7, and 3.8 are satisfied. Thus a queueing network defined by $K^m, a^m, R^m$, is pathwise stable. Proof of pathwise stability in fact follows from weak statbility of the fluid model of the network, as Dai and Lin [19] show: Under the conditions of Theorems 3.5, 3.6, 3.7, and 3.8, if the queueing network is controlled by a maximum pressure policy, then the fluid model of the network is weakly stable: if the fluid limit buffer levels at some time are all 0 then they remain so thereafter.

By our induction hypothesis:

$$\lim_{r \to \infty} \frac{\bar{Q}(rt)}{r} = \lim_{r \to \infty} \frac{\bar{Q}'(t)}{r} = \frac{\bar{Q}(t)}{r} = 0, \quad 0 \leq t \leq t_{m-1}. $$

Furthermore, the process $\bar{Q}'(t)$ is controlled by maximum pressure policy over the interval $t_{m-1} < t < \tilde{t}$. Hence, the fluid limit of $\bar{Q}$ will retain the value of 0 for $t_{m-1} < t < \tilde{t}$, and we have:

$$\lim_{r \to \infty} \frac{\bar{Q}(rt)}{r} = \lim_{r \to \infty} \frac{\bar{Q}'(t)}{r} = \frac{\bar{Q}(t)}{r} = 0, \quad t_{m-1} \leq t \leq \tilde{t}. \quad (5.10)$$

It follows the for $k \in K^m$, using (4.14),

$$0 = \lim_{r \to \infty} \bar{Q}_k^r(t)/r = \lim_{r \to \infty} \frac{r \mu_k u^m_k(t - t_{m-1})/r - (S_k(rT_k(t)) - S_k(rT_k(t_{m-1})))}{r}$$

$$= \mu_k (u^m_k(t - t_{m-1}) - (T_k(t) - T_k(t_{m-1})))$$

for all $t_{m-1} < t < \tilde{t}$, hence $\hat{T}(t) = u^m_k$ for all $t_{m-1} < t < \tilde{t}$. Recall that $T^N_k(t)$ is shared by $Q^N$ and $\bar{Q}^N$.

Recall that $\bar{Q}_k^N(t) = Q^N_k(t), k \in K^m_0$. Hence we have for $t_{m-1} < t < \tilde{t}$ that $\bar{Q}_k(t) = 0, k \in K^m_0$.

As we saw in the proof of part (i), the fluid limit must satisfy (5.3). Comparing to (4.13), we find that $\hat{T}_k(t) = u^m_k, k \in K^m$ and $\bar{Q}_k(t) = 0, k \in K^m_0$ imply that for all $k \in K, \hat{T}_k = u^m_k$, and $\bar{Q}_k = \bar{Q}_k^m$. We have shown that from the induction hypothesis we get $(\bar{Q}(t), \bar{T}(t)) = (q(t), \int_0^t u(s)ds)$ for all $t_{m-1} < t < \tilde{t}$.

By continuity, $(\bar{Q}(\tilde{t}), \bar{T}(\tilde{t})) = (q(\tilde{t}), \int_0^{\tilde{t}} u(s)ds)$. However, $\tilde{t} < \tilde{t}$ was arbitrary, hence $(\bar{Q}(t), \bar{T}(t)) = (q(t), \int_0^t u(s)ds)$ for all $t_{m-1} < t < \tilde{t}$.

Assume now that $\tilde{t} < t_m$. Then $\bar{Q}_k(t) = q_k(t), k \in K^m_0$ is bounded away from zero in the interval $t_{m-1} \leq t < \tilde{t}$, and by continuity also at $\tilde{t}$ and its neighborhood. But this is in contradiction to the definition of $\tilde{t}$. Hence, $\tilde{t} = t_m$ and we have completed our induction step.

This completes the proof that all fluid limits under the fluid tracking maximum pressure policy are in fact equal to the fluid solution.

We now complete the proof, by showing that almost surely (5.2). Consider any $\omega$ which satisfies (4.4.3.13). Take the sequence $Q^N_k(t)$. Then we can find a subsequence $r \to \infty$ such that $\frac{1}{r}Q^r$ converges to a fluid limit. But in that case, $\frac{1}{r}Q^N_k(t) \to q_k(t)$. So for every sequence we can find a subsequence which converges to the desired value. Hence $\frac{1}{N}Q^N_k(t) \to q_k(t)$. Clearly then also $\frac{1}{N}V^N \to V^*$.  

**Question** Can this be used to show uniform convergence ???
6 Extension to general unitary processing networks

In this section we describe one way in which to extend the results of Sections 4 and 5 to general unitary processing networks. These include multiclass queueing networks with routing decisions.

We now assume that our processing network is a general unitary network. Recall that a unitary processing network is defined as one in which each activity processes items from a single buffer/class, and utilizes a single resource.

Using the notations and definitions of the previous sections we now have: Classes $K = 1, \ldots, K$, actions $J = 1, \ldots, J$ and resources $I = 1, \ldots, I$. Activity $j$ utilizes only one resource, so that $(A_{1,j}, \ldots, A_{I,j})$ is a unit vector for each $j$. Furthermore, at each application of activity $j$ it will process items from a single class. To avoid triviality we assume that each class has at least one activity which processes items of that class. Let $\sigma(k) \subseteq \{1, \ldots, J\}$ be the set of activities which process items of class $k$. The choice of these activities constitutes routing decisions for the items of class $k$.

In order to use the fluid approach to control general unitary processing networks, we subdivide the items of class $k$ into several classes, one for each of the activities which process items of class $k$. The division is done simply by assigning each item as it enters the buffer of class $k$ into a subqueue which constitutes the buffer of items destined to be processed by activity $j$. Doing so converts the network into a standard multiclass queueing network.

In order to allocate the items which arrive in class $k$ to the buffers of activities $j \in \sigma(k)$ we use the results of the fluid solution. This allocation is specially tailored for each of the intervals of the fluid solution. In the $m$th interval of the fluid solution, time interval $(t_{m-1}, t_m)$, consider buffer $k$, and let $u^{m}_{j}$ be the fluid allocation of resources to activity $j \in \sigma(k)$. Then the rate of processing of items of buffer $k$ by activity $j$ during the $m$th interval is $B_{kj} \mu_j u^{m}_{j}$. We define a vector of fractions:

$$\eta_j = \frac{B_{kj} \mu_j u^{m}_{j}}{\sum_{j' \in \sigma(k)} B_{kj'} \mu_{j'} u^{m}_{j'}} \quad j \in \sigma(k).$$

We then route every item which enters buffer $k$ to subqueue $j$ according to the long term fraction $\eta_j$.

In other words: on the $\ell$s application of activity $j'$, it will seize $B_{kj'}$ items from subqueue $j'$ of buffer $k'$, and process them. At the end of the processing some of these items will be routed to buffer $k$, and of those routed to buffer $k$ a certain fraction may be routed to subqueue $j$ (where $j' \in \sigma(k')$ and $j \in \sigma(k)$). Define $\Phi^{m}_{jj'}(n)$ as the number of all the items routed from $j'$ to $j$ in the first $n$ applications of activity $j'$ following $t_{m-1}$. Then we need to have

$$\lim_{n \to \infty} \frac{1}{n} \Phi^{m}_{jj'}(n) = v_{kj'} \eta_j$$

Once we have sorted the items into these subqueues, in each interval $(t_{m-1}, t_m)$ we have a standard multiclass queueing network, and Theorem 5.1 continues to hold.
References


