6th IFAC Conference on Management and Control of Production and Logistics
The International Federation of Automatic Control
September 11-13, 2013. Fortaleza, Brazil

Optimal periodical behavior of a multiclass fluid flow network

D.A.J. van Zwieten * **E. Lefeber** * **I.J.B.F. Adan** *

* *Eindhoven University of Technology, P.O. Box 513,5600 MB Eindhoven, The Netherlands*
(e-mail: {D.A.J.v.Zwieten, A.A.J.Lefeber, I.Adan}@tue.nl)

Abstract: We consider the optimal scheduling problem of a single server multiclass fluid flow network with non-negligible setup times. The server is able to serve several queues simultaneously, each at its own rate, independent of the number of queues being served. As performance criteria we consider the weighted average amount of fluid in the queues, also known as work in progress, or we consider the cycle time. Restrictions on cycle time and service times are taken into account. A 4-queue server is presented to illustrate the process of deriving the optimal schedule.

Keywords: Optimal trajectory; Manufacturing networks; Quadratic programming .

1. INTRODUCTION

Queueing networks constitute a large family of models in a variety of settings, involving resources, such as 'jobs' or 'customers', that wait in queues until being served. Once its service is completed, a resource moves to the next prescribed queue, where it remains until being served. This procedure continues until the resource leaves the network. In a lot of applications, capacity over competing resources is shared by servers. One can think of communication via the Internet, signalized traffic intersections, or manufacturing networks. For all these applications, switching between resources might take time.

We consider the optimal scheduling problem multiclass fluid flow networks that can be described by a single switching server that competes over multiple queues. The problem consists of determining *optimal periodical behavior* or service timing plan, i.e., scheduling the service and idle times of each queue during a single cycle. Analytical derivation of optimal periodical behavior, e.g., as presented in van Eekelen et al. [2006], Ioslovich et al. [2011], is only possible for networks with two queues or flows. In practice, however, it is likely for servers to serve more than two queues. Therefore, we aim at deriving optimal periodical behavior of single server networks with multiple queues, by using optimization techniques.

For a network with a number of competing unlimited queues, negligible setup-times and cost of operation per unit linear in the queue content, it is well-known that the optimal policy is a μc-rule, see Baras et al. [1985] or Buyukkoc et al. [1985]. This policy allocates service to the non-empty queue with the largest rate of cost decrease. In this paper we assume that the server is able to serve several queues *simultaneously*, unlike the above mentioned papers. Moreover, each queue is served at a queue-dependent rate, which is independent of the number of queues being served. These kind of models may arise when studying, e.g., multiclass queueing tandems, multi-server polling systems with overtaking constraints and signalized traffic intersections. In addition to the network studied in Lefeber et al. [2011], where optimal control of a multiclass fluid flow network for which several queues can be served simultaneously is presented, we include external arrival rates in each queue and consider non-negligible setup times.

Examples of multiclass queuing networks considered are presented in Figure 1. The network on the left is a 4-queue two-server polling system with physical constraints such that the servers can not serve consecutive queues and the servers can not overtake each other, e.g., think of quay cranes at a container terminal serving vessels. In the middle, a traffic intersection with four flows is presented, for which conflicting flows can not be served simultaneously. The network on the right presents a a four queue tandem network where each of the three servers can serve two queues and only one queue can be served at a time. Each of these networks can be modeled as a single server with four queues that can serve several queues simultaneously, with a fixed service rate for each queue, i.e., capacity is not shared over multiple queues.

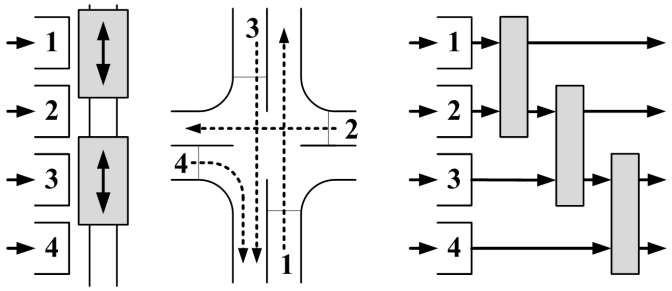

Fig. 1. Three examples of multiclass queueing networks: Two-server polling network (left), Traffic intersection (middle) and multiclass queueing tandem network (right).

We decompose the optimal scheduling problem into four subproblems that can be considered consecutively. *Grouping of queues* renders the first subproblem, i.e., which queues can be served simultaneously? The specification of the number and composition of groups of queues that can be served simultaneously is a nontrivial task that can have a major impact on the network performance. For scheduling of signalized intersections, work exists on grouping of non-conflicting traffic queues on an intersection, see for instance Improta and Cantarella [1984], Mohsen Hosseini and Orooji [2009], Stoffers [1968] or Tully [1977]. We introduce a similar approach. However, we take *all* possible groups into account, i.e., not only the *maximal* groups, but also all subgroups of these groups. In Gallivan and Heydecker [1988] and Irani and Leung [1996] also subgroups are taken into account, but the authors restrict themselves to sequences with a *single* green (service) period for each flow, whereas we allow *multiple* service periods for a flow in a cycle.
The second subproblem consists of combining the groups. For each combination, all queues are served at least once in a sequence. A maximal number of groups in a sequence is taken into account for computational reasons. However, all possible combinations are evaluated under this restriction. This includes considering combinations where some queues have multiple service times and some queues are served more often than others. Solving subproblems one and two is computationally expensive. To diminish the load, we use a recursive approach that eliminates unfeasible groups or combinations of groups as soon as possible by employing the constraints, instead of an exhaustive approach, i.e., deriving all possibilities.
The groups in each feasible combination of groups can be ordered in various ways, resulting in multiple sequences (or cycles) with possible different results. This renders the third subproblem. Sequences with similar behavior, based on the constitution of groups, are removed.
Finally, for each feasible sequence, optimal periodical behavior can be derived using linear or quadratic programming techniques. This is the fourth subproblem. As performance criteria, we consider the costs, which are a linear function of the queue contents, or the cycle time.

The remainder of this paper is organized as follows. Section 2 introduces the model description and notation. The performance criteria are presented in Section 3. Derivation of the optimal periodical behavior and network constraints are presented in Section 4. In Section 5 an example of 4-queue switching server is presented to illustrate this method. Conclusions are provided in Section 6.

2. MODEL

We consider multiclass fluid flow networks which can be described by a single switching server that competes over $N > 1$ queues. Each queue $n \in \mathcal{N} = \{1, 2, ..., N\}$ has fluid input at constant arrival rate λ_n. The content of queue n is denoted by x_n. If the server serves queue n, the service rate is given by $r_n \in [0, \mu_n]$.

Before focussing on the model, we reconsider the networks presented Figure 1. Each of these networks can be modeled as a single server. This server can serve queues 1 and 3, 1 and 4, and 2 and 4 simultaneously. Also, each queue can be served separately,

In van Eekelen [2008] it is shown that for optimal policies, a server, once serving a queue, does not idle and serves at maximal rate. Therefore, if queue n is served, the service rate is given by

$$r_n = \begin{cases} \mu_n & \text{if } x_n > 0, \\ \lambda_n & \text{if } x_n = 0. \end{cases} \tag{1}$$

In other words, if the queue is nonempty, service is at maximal rate, otherwise at arrival rate. Define the load of queue n by $\rho_n = \frac{\lambda_n}{\mu_n}$. The queues are divided into groups. A *group* $m \in \mathcal{M}$ is a set of queues, e.g., $\{n_1, n_2, n_3\}$, that can be served simultaneously. The set of all groups is denoted by $\mathcal{M}$. A *sequence* $s \in \mathcal{S}$ consists of multiple groups, e.g., (m_1, m_2, m_3), and is a single repetition of the service operations, sometimes referred to as signal cycle or period. Note that a sequence specifies the order in which the groups are served. The set of feasible sequences is denoted by $\mathcal{S}$. For computational reasons, we restrict ourselves to a reasonable maximal number of groups in a sequence, given by

$$|s| \leq S, \qquad \forall s \in \mathcal{S}. \tag{2}$$

The time it takes to serve all groups in a sequence is called the *cycle time*, denoted by T. A service (idle) time is defined as the uninterrupted interval during which the queue is (not) served. Note that during a setup no queue content is processed, hence a setup is part of the idle time. The duration of a service (idle) time for queue n in group m is nonnegative, is labeled service (idle) time and is denoted by $\tau_{m,n}$ ($t_{m,n}$). Furthermore, the maximal number of service times of a queue during one cycle for sequence s is denoted by γ_s. A service time indicates the duration of serving a queue without interruption, it starts after an idle time, can be spread over multiple consecutive groups and ends before another idle time. Receiving a service time at the first and last group of a sequence also counts as a single service time, as the sequence is repeated. Let Γ denote the maximal number of service times for a queue in a sequence, i.e.,

$$\gamma_s \leq \Gamma, \qquad \forall s \in \mathcal{S}. \tag{3}$$

The maximal number of service times for a queue in one cycle is bounded by

$$\Gamma \leq \left\lfloor \frac{S}{2} \right\rfloor, \tag{4}$$

since the maximal number of service times, which are interrupted by idle times, is at most half of the number of groups in the sequence. For operational purposes, it is desired to keep Γ low.

Switching between serving queue $n_i \in \mathcal{N}$ to serving queue $n_j \in \mathcal{N}$ with $n_i \neq n_j$ requires a *setup-time* $\sigma_{n_i,n_j} \geq 0$. The setup time for switching from a group of queues $m \in \mathcal{M}$ to a single queue $n \in \mathcal{N}$ is the *maximal* setup time between all these queues:

$$\sigma_{m,n} = \max_{n_i \in m} \sigma_{n_i,n}.$$

Furthermore, switching between non-conflicting queues, i.e., queues that can be served simultaneously, does not require a setup time. For sequence s, define the set of groups in which queue n is served by

$$\mathcal{G}_{s,n} = \{i \in \{1,2,...,|s|\} | n \in m_i\}.$$

For $s = (m_1, m_2, m_3)$ and $\mathcal{G}_{s,n} = \{m_2, m_3\}$, queue n is served in group m_2 after setup time $\sigma_{m_1,n}$ and service ends at the end of serving group m_3. Moreover, switching between groups m_2 and m_3 does not interrupt the service time of queue n, i.e., $\sigma_{m_2,n} = 0$.

Assumption 1.

$$\sum_{i=1}^{|s|} \sum_{n} \sigma_{m,n} > 0, \qquad \forall s \in \mathcal{S},$$

with $n \in m_{(i \bmod |s|)+1}$.

This assumption states that during a sequence at least one setup of non-zero duration is present.

For operational reasons, some restrictions can be imposed on the network. Minimal and maximal cycle times, respectively $T^{\min}$ and $T^{\max}$, can be taken into account:

$$T^{\min} \leq T \leq T^{\max}. \tag{5}$$

Also, minimal service times can be required. As the service time of a queue is not restricted by a single group, let us define by $G_{s,n}$ the set of sets of consecutive groups in which queue n is served in sequence s. Formally, for a sequence $s = \left(m_1, m_2, ..., m_{|s|}\right)$, we define

$$G_{s,n} = \{g_{s,n} \in P(\mathcal{G}_{s,n}) \backslash \emptyset | \exists! i \in g_{s,n} : i \bmod |s|+1 \notin \mathcal{G}_{s,n} \\ \wedge \exists! j \in g_{s,n} : (j-2) \bmod |s|+1 \notin \mathcal{G}_{s,n}\},$$

where $P(q)$ denotes the *powerset* of any set q, which is the set of all subsets of q, including q and $\emptyset$, e.g., $P(\{1,2\}) = \{\{1,2\},\{2\},\{1\},\emptyset\}$.

The maximal number of service times that a queue receives during one cycle, γ_s, can be derived from $G_{s,n}$ as follows:

$$\gamma_s = \max_{n \in \mathcal{N}} |G_{s,n}|.$$

The minimal service times are imposed via

$$\sum_{i \in g_{s,n}} \tau_{m_i,n} \geq \tau_n^{\min} \qquad \forall g_{s,n} \in G_{s,n}, \quad \forall n \in \mathcal{N}. \tag{6}$$

For example, if $s = (m_1, m_2, ..., m_5)$ and queue n is contained in groups m_1, m_3 and m_5, we have

$$\mathcal{G}_{s,n} = \{1,3,5\},$$
$$G_{s,n} = \{\{1,5\},\{3\}\},$$

which yields the following service time restrictions:

$$\tau_{1,n} + \tau_{5,n} \geq \tau_n^{\min},$$
$$\tau_{3,n} \geq \tau_n^{\min}.$$

Below we formulate the problem of determining the optimal periodical schedule.

Problem formulation
Derive the optimal schedule of service and idle times for a single switching server that competes over N queues, given arrival rates, service rates and setup times and satisfying $|s| \leq S$ and $\gamma_s \leq \Gamma$. In other words, find the periodical service and idle times for each queue that minimizes a performance criterium.

3. PERFORMANCE CRITERIA

The optimal periodical behavior depends on the performance criteria. There exists a plethora of performance indicators. We consider the cycle time or the costs, which are a linear function of the queue contents. This is also referred to as minimizing the weighted average work in process (wip) level. This term will be used throughout the paper. These criteria are discussed below. Note that we do not take setup costs into account. However, setups are penalized by setup times which lead to larger queue contents, and therefore higher costs.

3.1 Cycle time

The first performance criterium we consider is cycle time. Moreover, the minimal cycle time is also used to speed up the optimization of the weighted average wip level, as shown below. Note that minimal cycle time does not necessarily imply full capacity for the queues with the highest load in each group. Due to both constraints (5) and (6), the network might have spare capacity at the minimal cycle time. For ease of exposition, we introduce α_m as the total time required by group m, i.e., setting up to and serving group m:

$$\alpha_m = \sigma_{m-1,n} + \tau_{m,n} \qquad \forall n \in m, \tag{7}$$

where $m-1$ indicates the predecessor of group m. Then, the cycle time for sequence s can be described by:

$$T = \sum_{m=1}^{|s|} \alpha_m. \tag{8}$$

The problem is to find the service and idle times for each queue for a given sequence. Note that the idle times of queue n are a combination of setup times and times required by groups not serving queue n. Using the service times as optimization variables, the cycle time (8) can be written as a linear combination of the service times. These linear objective functions are subject to linear equality and linear inequality constraints, which are presented below. Therefore, minimizing the cycle time results in a LP problem.

3.2 Weighted average wip level

A well-known performance criterium is the work in progress (wip) level. The time averaged wip of queue n is denoted by

$$w_n = \frac{1}{T}\int_0^T x_n(\tau)d\tau. \tag{9}$$

Weights on the queue content n, which can, for example, represent costs, are denoted by c_n. The weighted time average wip level of the network is given by:

$$W = \sum_{n=1}^{N} c_n w_n. \tag{10}$$

The average wip level (9) of a single queue can be described by the duration of the service and idle times during a cycle, discussed below. By means of an example we illustrate the derivation of the average wip level. Consider a sequence consisting of five groups and a queue n that is served twice during a sequence, e.g., $\mathcal{G}_{s,n} = \{1, 2, 4\}$. Note that the queue receives two separate service periods although it is contained in three groups. The queue content during a single cycle is presented in Figure 2.

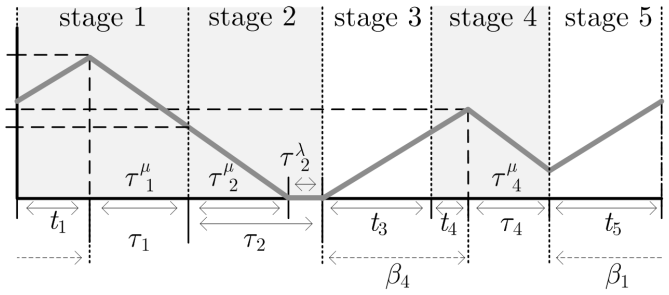

Fig. 2. Queue content evolution, queue served twice in a cycle.

For ease of exposition, the subscript $_{,n}$ is omitted in this example and we denote by β_m the idle time duration before serving queue n in group m. The figure presents also all service and idle times for each group and the maximal queue contents at the group in which the flow is served, i.e., x_1 and x_3. Note that if queue n is served multiple times in a sequence, the queue is not necessarily emptied each time during service. The total queue content during this cycle equals the area beneath the graph.

Calculation of the total queue content, or wip level, can be split into $|\mathcal{G}_{s,n}|$ parts, three for this example, where each part consists of a possible idle time and successive service time.

Let us consider the first part of queue content n in Figure 2, i.e., the evolution during $\beta_1+\tau_1$. The area can be easily computed, since the part consists of a linear increase with rate λ_n during β_1, decrease with rate $\lambda_n - \mu_n$ during τ_1^μ and constant level during τ_1^λ, which is zero in this case. Therefore, the average queue content during $\beta_1 + \tau_1$, denoted by w_1, is given by

$$\begin{aligned} w_1 =& \frac{1}{T}[x_1(\tau_1+\beta_1) - (\mu_n - \lambda_n)\,(\tfrac{1}{2}\tau_1^\mu + \tau_1^\lambda)\tau_1^\mu \\ & - \tfrac{1}{2}\lambda_n\beta_1^2], \end{aligned} \tag{11}$$

with

$$x_1 = \beta_1\lambda_n + x_3 - \tau_4^\mu(\mu_n - \lambda_n)$$

which can be generalized for each queue. With $\beta_{i,f}$ the sum of setup times and green times $\tau_{m,\not n}$, it can be seen that equation (11) is quadratic in the optimization variables $\tau_{m,n}^\mu$ and $\tau_{m,n}^\lambda$, provided that cycle time T is a constant.

4. OPTIMAL PERIODICAL BEHAVIOR

Optimal periodical behavior is derived in four consecutive subproblems. These subproblems, i.e., grouping of queues, combining groups, deriving feasible sequences and optimizing a sequence, are presented below.

4.1 Group generation

First, all possible combinations of queues which can be served simultaneously are grouped. This equals finding all independent sets in a graph. More specifically, considering graphs, this amounts to finding all sets of vertices in a graph in such a way that in a set no edges are connected. A multiclass queueing network can be considered as an undirected graph $G = (N, E)$ consisting of a set of vertices N together with a set of edges E. Vertices represent queues and edges denote conflicting queues, i.e., these queues can not be served simultaneously (for instance queues 1 and 2 for the system in Figure 1). The graph corresponding to the networks presented in Figure 1 is depicted in Figure 3. For this network, servers 1 and 2, 2 and 3, and 3 and 4 are unable to operate simultaneously. Hence, these vertices are connected.

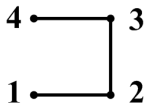

Fig. 3. Graph corresponding to the 4-queue switching server.

Let us define allowed groups:

Definition 2. A set $m \subseteq V$ is labeled *allowed group*, when m is an independent set for graph G, i.e., for every two vertices in m there is no edge connecting the two. Let $\mathcal{M}$ denote the set of all allowed groups in G.

In other words, the set of allowed groups $\mathcal{M}$ is the set of all independent sets of graph G. Generating all groups is equivalent to finding all allowed groups. The groups are generated by starting from an empty set and recursively adding vertices to this set. The connection between the vertices in the set is checked after every addition. If none of the vertices in the set are connected, the set is stored and the addition of vertices continues. If an edge between vertices in the set exists, i.e., conflicting queues, the set is dismissed and also all possible sets resulting from this set are disregarded. Dismissing a set and thereby disregarding

all possible sets resulting from this set speeds up the process drastically compared to an exhaustive search. The set of allowed groups for the 4-queue switching server is given by:

$$\mathcal{M} = \{\{1,3\},\{1,4\},\{2,4\},\{1\},\{2\},\{3\},\{4\}\}.$$

A maximal group is defined by:

Definition 3. A group of P is a *maximal* group of P if it is not properly contained in another group of P.

As mentioned before, unlike some approaches used in literature, we do not only take the maximal groups into account, but also *all subsets* of the maximal groups. It might seem counterintuitive to take subsets of maximal groups into account, as the maximal groups contain the most queues and therefore achieve the highest decrease in load if the queues in the group are served. However, conflicting queues and topology-dependent setup times can result in using proper subsets of maximal groups in the optimal sequence. The set of maximal groups for our illustrative example is given by

$$\{\{1,3\},\{1,4\},\{2,4\}\}. \tag{12}$$

All allowed groups are the powersets of these maximal groups.

4.2 Combining groups

Having generated the groups, the second subproblem consists of combining these groups. More specifically, finding all feasible combinations of groups. A combination of groups is feasible if it contains all queues, i.e., all queues are served at least once in a sequence and the number of groups satisfies (2). The sequences are generated using a similar approach as in Section 4.1, by recursively enumerating all possible combinations of groups, regarding the constraints. The resulting sets are contained in the family of feasible sets of powerset $P(\mathcal{M})$, and is denoted by $P^{\text{feas}}(\mathcal{M})$. In addition, a lower bound on the number of groups in a sequence can be imposed if desired.

4.3 Sequence generation

Third, for each set of combined groups in $P^{\text{feas}}(\mathcal{M})$, the groups can be ordered in different ways, resulting in different sequences. If a combination of groups consists of i groups, $(i-1)!$ different sequences arise, since w.l.o.g. it can be assumed that the first group is fixed in the sequence. From all generated sequences, infeasible sequences are discarded to minimize optimization time. A sequence is infeasible if (3) is violated or if it contains two or more identical groups in consecutive order, as these groups can be lumped together resulting in a sequence which is already considered or not allowed. This yields a set of feasible sequences $\mathcal{S}$.

4.4 Optimization

Having generated all feasible sequences $s \in \mathcal{S}$ in the first three subproblems, we derive for each of these sequences the service and idle times for best performance. From these results, the best solution is selected, which renders the optimal periodical behavior. An individual sequence is optimized using linear programming (LP) or quadratic programming (QP), depending on the performance criteria.

In order to use QP, the objective function of deriving the wip must be quadratic with respect to the optimization variables, i.e., service times. In (11) it can be seen that the objective function is *quadratic* in the optimization variables, provided that the cycle time T, which is also a function of the service times, is a constant. Hence, to derive the optimum for a given sequence using QP, the solution of the QP must be derived for all cycle times in the range (5). However, by deriving the minimal required cycle time T^*, using a LP as shown above, the range can be reduced if $T^{\min} < T^*$ or the sequence can be discarded if $T^{\max} < T^*$. Next, the optimum is found by minimizing the objective function over the range $\min(T^{\min}, T^*) \le T \le T^{\max}$. Most of the times this optimum is located at the lower bound of T and can be determined by solving the QP twice, i.e., if $J(\min(T^{\min}, T^*)) < J(\min(T^{\min}, T^*) + \epsilon)$ where $J(t)$ is the minimal objective value as function of cycle time. Otherwise, an efficient algorithm, e.g., the bisection method, can be used to locate the optimum.

Moreover, in addition to the introduced constraints for the cycle time (5) and the service times (6), a stability constraint is required for periodical behavior. Total service time for a queue during a sequence must be large enough to serve all arrivals during a sequence, i.e.,

$$\rho_n T \le \sum_{m \in \mathcal{G}_{s,n}} \tau_{m,n}, \qquad \forall n \in \mathcal{N}.$$

Furthermore, service at maximal rate is bounded, since the queue contents are non-negative:

$$\tau^{\mu}_{m,n} \le \frac{x_{m,n} + \lambda_n \sigma_{m-1,n}}{\mu_n - \lambda_n}, \qquad \forall n \in \mathcal{N} \quad \forall m \in \mathcal{G}_{s,n},$$

where $x_{m,n}$ denotes the queue content of x_n at the start of serving group m, and $\tau^{\mu}_{m,n}$ denotes the service time at maximal rate for queue n in group m. Note that this time, together with the service time at arrival rate, denoted by $\tau^{\lambda}_{m,n}$, equals the service time, i.e.,

$$\tau_{m,n} = \tau^{\mu}_{m,n} + \tau^{\lambda}_{m,n}, \qquad \forall n \in \mathcal{N} \quad \forall m \in \mathcal{G}_{s,n}.$$

Also, all queue contents and service times are nonnegative, which are the final constraints for this problem.

5. EXAMPLE

To illustrate the derivation of the optimal schedule, we consider the 4-queue switching server from Figure 1 again. For this network we consider the parameters

$$\begin{aligned} \lambda &= [6\ 1\ 2\ 3]\,, \\ \mu &= [12\ 12\ 12\ 8]\,, \\ c &= [3\ 1\ 1\ 1]\,. \end{aligned}$$

Moreover, all setup times are equal to 5 time units. Restrictions on the cycle time and minimal service times are given by:

$$80 \leq T \leq 100,$$
$$\tau^{\min} = [5\ 5\ 5\ 6]\,.$$

All weights are assumed 1. For this network a simple schedule is desired, therefore a sequence is limited to contain maximally six groups, i.e., $S = 6$. The objective is to minimize the average weighted wip level. Using the method described in this paper, the sequences with lowest weighted average wip levels for given upper bound on the number of groups S and maximal number of service times given by (4) are presented in the following table. For each of these sequences the constitution of groups are given, labeled by m_j with $j = 1, 2, ..., S$.

S	m_1	m_2	m_3	m_4	m_5	m_6
3	1, 3	1, 4	2, 4	-	-	-
4	1, 3	1, 4	1, 3	2, 4	-	-
5	1, 3	1, 4	1, 3	2, 4	1, 4	-
6	1, 3	1, 4	1, 3	2, 4	1, 3	1, 4

The average weighted wip level W, corresponding cycle time T and group times α_j for these sequences are presented in the table below.

S	W	T	α_1	α_2	α_3	α_4	α_5	α_6
3	180.2	80	25.0	43.3	11.7	-	-	-
4	164.8	80	12.3	41.7	14.3	11.7	-	-
5	149.2	80	11.7	28.3	11.7	11.7	16.7	-
6	144.6	80	10.0	19.2	10.0	11.7	10.0	19.2

For $S = 2$, (sequence $\{\{1, 3\}, \{2, 4\}\}$) no periodical solution exists, since not all arriving products can be served. For $S = 3$ all arriving products can be served and all queues are served once. The results from $S > 3$ clearly show that performance improves by allowing multiple service times to queues 3 and 4. Moreover, queue 1, which has the highest load, is served in as many groups as possible. Note that the solution for $S = 5$ with groups m_4 and m_5 interchanged yields the same results, as the setup times are constant. If a single service period is allowed for each queue ($\Gamma = 1$), an optimal periodical solution is given by the solution for $S = 3$.

6. CONCLUSIONS

This paper presents a method to derive an optimal periodical schedule for a multiclass queueing network with deterministic arrival and service rates. For each feasible sequence, including sequences where some queues are served more than others, the service and idle times for each queue are derived to obtain optimal periodical behavior. With minimal cycle time as performance criterium, the problem is solved using LP. For the weighted average wip level as criterium, the problem can be written as a QP problem, provided that the cycle time is constant. In this case, optimal periodical behavior for a sequence is obtained by solving the QP problem considering a range of feasible cycle times.

Sequences are derived in three consecutive steps. First, all possible combinations of queues that can be served simultaneously are grouped. Second, these groups are combined into all feasible sets of groups, i.e., each set complies to restrictions on number of groups in a set and number of service times. Third, from these sets all feasible sequences are constructed by regarding the permutations in groups.

So far, we considered a single server that can serve multiple queues simultaneously. We are currently extending our method to multiclass queueing networks which can not be represented by a single server. Moreover, the arrival rates for these queues can be piecewise constant if two consecutive queues are served simultaneously.

ACKNOWLEDGEMENTS

This work is supported by the Netherlands Organization for Scientific Research (NWO-VIDI grant 639.072.072).

REFERENCES

J.S. Baras, D.-J. Ma, and A.M. Makowski. K competing queues with geometric service requirements and linear costs: The μc-rule is always optimal. *Systems & Control Letters*, 6(3):173 – 180, 1985.

C. Buyukkoc, P. Varaiya, and J. Walrand. The μc-rule revisited. *Advances in Applied Probability*, 17:237–238, 1985.

S. Gallivan and B. Heydecker. Optimising the control performance of traffic signals at a single junction. *Transportation Research Part B: Methodological*, 22(5):357–370, 1988.

G. Improta and G.E. Cantarella. Control system design for an individual signalized junction. *Transportation Research Part B: Methodological*, 18(2):147–167, 1984.

I. Ioslovich, J. Haddad, P. Gutman, and D. Mahalel. Optimal traffic control synthesis for an isolated intersection. *Control Engineering Practice*, 19(8):900 – 911, 2011. ISSN 0967-0661.

S. Irani and V. Leung. Scheduling with conflicts, and applications to traffic signal control. In *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, SODA '96, pages 85–94, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.

E. Lefeber, S. Lämmer, and J.E. Rooda. Optimal control of a deterministic multiclass queuing system for which several queues can be served simultaneously. *Systems & Control Letters*, 60(7):524 – 529, 2011.

S. Mohsen Hosseini and H. Orooji. Phasing of traffic lights at a road junction. *Applied Mathematical Sciences*, 3 (29-32):1487–1492, 2009.

Karl E. Stoffers. Scheduling of traffic lights. a new approach. *Transportation Research*, 2(3):199–234, 1968.

I.M.S.N.Z Tully. *Synthesis of sequences for traffic signal controllers using techniques of the theory of graphs*. PhD thesis, University of Oxford, 1977.

J.A.W.M. van Eekelen. *Modelling and control of discrete event manufacturing flow lines*. PhD thesis, Eindhoven University of Technology, 2008. URL `alexandria.tue.nl/extra2/200712107.pdf`.

J.A.W.M. van Eekelen, E. Lefeber, and J.E. Rooda. Feedback control of 2-product server with setups and bounded buffers. In *Proceed of the American Control Conference, 2006*, pages 544–549, 2006.