# Bottlenecks Identification in Multiclass Queueing Networks using Convex Polytopes

Giuliano Casale Giuseppe Serazzi
Politecnico di Milano
Dipartimento di Elettronica e Informazione (DEI)
via Ponzio 34/5, 20133 Milano, Italy
[casale, serazzi]@elet.polimi.it

## Abstract

*It is known that the resources that limit the overall performance of the system are the congested ones, referred to as bottlenecks. From the knowledge of the bottleneck stations with a limited computational effort it is possible to derive asymptotic values of several performance indices. While identifying the bottleneck stations under a single-class workload is a well-established practice, no simple methodology for multiclass models exists. In this paper we present new algorithms for identifying the bottlenecks in multiclass queueing networks with constant-rate servers. We show how the application of assessed techniques, such as the ones related to the convex polytopes, can provide insights on the performance of a queueing network. The application of our techniques to the asymptotic analysis of closed product-form networks is also investigated.*

## 1. Introduction

The complexity of modern computer infrastructures makes the application of known solution techniques infeasible for these new environments. On one side, exact solution techniques, e.g., the convolution algorithm [4] and the MVA [16], are prohibitively expensive in term of computational resource requirements. On the other side, approximate solution techniques (see e.g., [6, 8]) may suffer an uncontrolled decrease in accuracy as the complexity of the model grows [18].

Thus, with computer systems that comprise hundreds of servers, LANs and WANs and workloads consisting of large populations of several heterogeneous customer classes an interesting alternative (in some cases the only one feasible) to both exact and approximate solutions is represented by asymptotic techniques. With a limited effort it is possible to determine asymptotic values of several performance indices such as system response time and throughput, resources utilizations and queue lengths.

The key to determine the asymptotic performances is the knowledge of the resources that saturate at the lowest load, i.e. the bottleneck resources. The identification of the bottlenecks is important also in tuning studies in order to evaluate the performance gains of different tuning alternatives.

While identifying the bottleneck stations under a single-class workload is a well-established practice, no simple methodology for multiclass models exists. In [9] it is shown that the bottleneck for single class models is the station $i$ with the largest service demand $S_i V_i$, under the assumptions of invariance of service times $S_k$, visit ratios $V_k$ and routing frequencies.

Several works [17, 2, 13, 9] have studied the identification of bottlenecks in multiclass closed product-form networks as the population grows to infinity. In [2] it is shown that multiclass models can exhibit *multiple* simultaneous bottlenecks and the dependency of the bottlenecks set on the population mix is derived. Approximate methods to analyze the asymptotic behavior of the network taking into account the existence of bottleneck shifting points are presented in [13, 17]. Bounds on utilizations for finite populations can be computed using the results in [9, 14, 12, 10].

In this paper we present a technique, based on the theory of convex polyhedra, for identifying the bottlenecks of multiclass queueing networks with constant-rate servers. It is interesting to point out that our method relies on fewer assumptions than those of product-form networks. For the specific case of product-form networks, we study the implications of our results on the asymptotic analysis of such models. Preliminary results of this work have been presented in [5].

The paper is organized as follows. Section 2 introduces the notation and a taxonomy of queueing networks stations. Section 3 describes the bottleneck identification methodology. In Section 4 the relationships between the proposed technique and the asymptotic analysis of closed product-

form networks is examined. Finally, Section 5 illustrates experimental results.

## 2. Background

### 2.1. Notation

We consider only queueing networks with constant-rate single-server stations. Let $\mathbf{R} = \{1, 2, \ldots, R\}$ be the set of customer classes indices and $\mathbf{M} = \{1, 2, \ldots, M\}$ the set of station indices. Stations will be indexed with $i \in \mathbf{M}$; customer classes with $r \in \mathbf{R}$. Let $V_{ir}$ be the total number of visits of a class-$r$ customer to the station $i$ and $S_{ir}$ be its service requirement at each visit to the station $i$. The average loadings $L_{ir} = V_{ir}S_{ir}$ imposed by class-$r$ customers on station $i$ are collected in the loading matrix $\mathbf{L} = \{L_{ir}\}$. Let us consider the case of the following matrix with five stations and two customer classes:

$$\mathbf{L} = \begin{bmatrix} 20 & 80 \\ 65 & 60 \\ 90 & 30 \\ 25 & 65 \\ 40 & 45 \end{bmatrix} \tag{1}$$

$L_{31} = 90$ and $L_{32} = 30$ are the loadings imposed on station 3 by class 1 and class 2 customers, respectively. In the rest of paper matrix (1) will be used as a running case to illustrate our methodology. We also assume the $L_{ir}$ to be positive quantities, and $\mathbf{L}$ to be non-singular.

The per-class arrival rates $\boldsymbol{\lambda} = \{\lambda_r\}$ and the population vector $\boldsymbol{N} = \{N_r\}$ specify the input workload for open and closed models, respectively. The notation used for closed models can be easily extended to open models by replacing $\boldsymbol{N}$ with $\boldsymbol{\lambda}$. Class-$r$ throughputs and utilizations are denoted with $X_r(\boldsymbol{N})$ and $U_{ir}(\boldsymbol{N})$, respectively. In some of the proofs of this paper we shall refer to the utilization law

$$U_i(\boldsymbol{N}) = \sum_{r \in \mathbf{R}} U_{ir}(\boldsymbol{N}) = \sum_{r \in \mathbf{R}} X_r(\boldsymbol{N}) L_{ir}, \quad i \in \mathbf{M} \tag{2}$$

and the related relation

$$L_{kr} U_{ir}(\mathbf{N}) = L_{ir} U_{kr}(\mathbf{N}), \quad \forall \mathbf{N} \tag{3}$$

which hold for load-independent general multiclass queueing networks.

### 2.2. Taxonomy of Stations

We now introduce a taxonomy of stations, which extends the one in [2] (see Fig. 1). We call *bottleneck* station the queueing center with the highest utilization. Since the same network can exhibit different bottlenecks depending on the population vector, we call *actual bottlenecks set* $B_{\boldsymbol{N}}$ the set

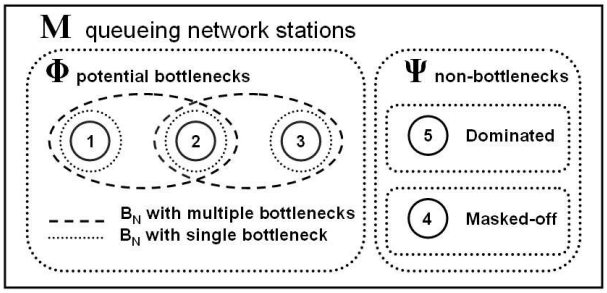


**Figure 1. Taxonomy of the set $\mathrm{M}$ of network stations specified by the loading matrix (1).**

of bottleneck stations when a population $\boldsymbol{N}$ is present in the network. Thus

$$B_{\boldsymbol{N}} = \{i \in \mathbf{M} \mid U_i(\boldsymbol{N}) = U^b_{\boldsymbol{N}}\}, \tag{4}$$

where $U^b_{\boldsymbol{N}} = \max_{j \in \mathbf{M}} U_j(\boldsymbol{N})$ is the *actual bottlenecks utilization*. We define *potential bottleneck set* $\boldsymbol{\Phi}$ the set of stations that can become bottlenecks for some feasible population $\boldsymbol{N}_k$, i.e.

$$\boldsymbol{\Phi} = B_{\boldsymbol{N}_0} \cup B_{\boldsymbol{N}_1} \cup \ldots \cup B_{\boldsymbol{N}_k}, \ \forall \text{ feasible } \boldsymbol{N}_k.$$

We denote with $|\boldsymbol{\Phi}|$ the cardinality of $\boldsymbol{\Phi}$.

Furthermore, we say that a station is a *natural bottleneck* for class-$r$ if it is an actual bottleneck for the special case where the network contains only class-$r$ customers. In the workload described by matrix (1), station 1 and station 3 are natural bottlenecks for class 2 and 1, respectively. Indeed, $\boldsymbol{\Phi}$ contains at least all natural bottlenecks of the network. Potential bottlenecks that are not natural bottlenecks for any of the customer classes are referred to as *network bottlenecks*.

The stations that cannot become bottlenecks for any feasible population are grouped in the set of *non-bottleneck stations* $\boldsymbol{\Psi} = \mathbf{M} - \boldsymbol{\Phi}$. Among these, we distinguish two kinds of centers:

- a station $j \in \boldsymbol{\Psi}$ is said to be *dominated* by $i \in \mathbf{M}$ iff

$$L_{jr} < L_{ir}, \quad \forall r \in \mathbf{R}, \tag{5}$$

- all non-dominated stations $k \in \boldsymbol{\Psi}$ are called *masked-off stations*.

Different reasons may prevent a masked-off station from becoming a bottleneck, like *compound domination* that will be addressed in Section 3.1. Station 5 of (1) is instead an example of dominated station. Note that according to (2) it is always $U_2(\mathbf{N}) > U_5(\mathbf{N})$. Also, as we shall prove in the following section, station 4 is a masked-off station, while station 2 is a network bottleneck.

## 3. Bottlenecks Identification Methodology

### 3.1. Characteristic polytope

Let $\mathbf{L}_i = \{L_{i1}, L_{i2}, \ldots, L_{iR}\}$ be the $i$-th row of $\mathbf{L}$, corresponding to the set of per-class loadings of station $i$. We can map $\mathbf{L}_i$ to a point in the *loadings space* $\mathbb{R}^R$, thus viewing the loading matrix $\mathbf{L}$ as a set of $M$ points in a $R$ dimensional space.

Using standard terminology, we call *$k$-dimensional face* a non degenerate intersection of a $k+1$-dimensional hyperplane with the surface of a bounded polyhedron (a *polytope*). Let $conv(\mathbf{A})$ denote the convex hull of a set of points $\mathbf{A} = \{\mathbf{a_0} \in \mathbb{R}^n, \mathbf{a_1} \in \mathbb{R}^n, \ldots, \mathbf{a_M} \in \mathbb{R}^n\}$, that is the smallest convex set containing the points of $\mathbf{A}$. A classical result is that the convex hull of a set of point is a polytope. Also let $proj(\mathbf{A})$ be the set of all possible projections of the points $\mathbf{a} \in \mathbf{A}$ that are not members of $\mathbf{A}$, that is the set of all vectors derived by $\mathbf{a}$ by setting to zero $k$ of its components, for all possible $k = 1, ..., n$ and for all possible positions. Figure 2 illustrates the projection $proj(\mathbf{L})$ of matrix (1). Referring to station 5, its projection is the set $proj(\mathbf{L_5}) = \{(0, 45), (40, 0), (0, 0)\}$.

We now give the following definition:

**Definition 1. Characteristic polytope.** *The characteristic polytope of a queueing network with loading matrix $\mathbf{L}$ is the convex polytope $T_\mathbf{L} = conv(\mathbf{L} \cup proj(\mathbf{L}))$.*

The characteristic polytope of (1) is shown in Figure 3. Figure 4 illustrates $T_{\mathbf{L}'}$ for the five stations and three customer classes matrix

$$\mathbf{L}' = \begin{bmatrix} 90 & 30 & 70 \\ 65 & 60 & 10 \\ 20 & 80 & 10 \\ 25 & 65 & 10 \\ 40 & 45 & 10 \end{bmatrix}. \quad (6)$$

Let us now denote with $\partial T_\mathbf{L}$ the boundary of $T_\mathbf{L}$. We can prove the following result:

**Theorem 1. Potential bottlenecks.** *All potential bottlenecks lie on $\partial T_\mathbf{L}$.*

*Proof.* To prove the statement it is sufficient to show that every station in the interior of $T_\mathbf{L}$ cannot become a bottleneck. Let $\mathbf{L}_k$ be an interior point of $T_\mathbf{L}$. Therefore we can always find a $\mathbf{v} \in T_\mathbf{L}$ such that $\mathbf{v} > \mathbf{L}_k$. Since $T_\mathbf{L}$ is a convex polytope, we can express $\mathbf{v}$ as the convex combination

$$\mathbf{v} = \sum_{i \in \mathcal{V}(\mathbf{v})} \alpha_i \mathbf{V}_i, \quad \sum_{i \in \mathcal{V}(\mathbf{v})} \alpha_i = 1, \quad \alpha_i \geq 0,$$

where $\mathcal{V}(\mathbf{v}) \in \partial T_\mathbf{L}$ is a set of indices of $d+1$ independent vertices $\mathbf{V}_i \in \partial T_\mathbf{L}$ that depends on $\mathbf{v}$, and $d$ is the (affine) dimension of $T_\mathbf{L}$. For each $\mathbf{V}_i \in proj(\mathbf{L})$ we have

$$\mathbf{V}_i \leq \mathbf{L}_i, \quad \forall i \in \mathcal{V}(\mathbf{v}),$$

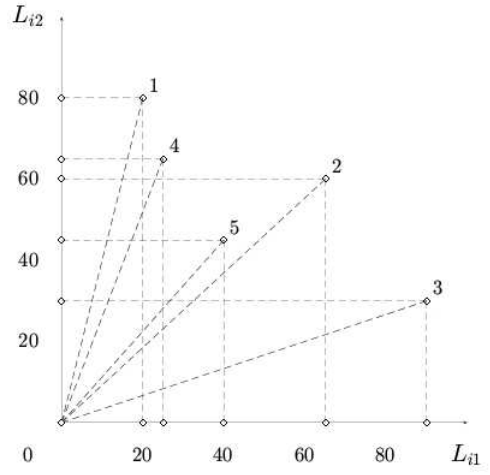


**Figure 2. Service demands of matrix $\mathbf{L}$ (1) and the set of its projections $proj(\mathbf{L})$ in the loadings space $\mathbb{R}^2$.**

where $\mathbf{L}_i$ is the point from which $\mathbf{V}_i$ is projected. This yields

$$\mathbf{L}_k < \mathbf{v} \leq \sum_{i \in \mathcal{V}(\mathbf{v})} \alpha_i \mathbf{L}_i,$$

and inserting (2) and (3) we get

$$U_k(\mathbf{N}) < \sum_{i \in \mathcal{V}(\mathbf{v})} \alpha_i U_i(\mathbf{N}), \quad \forall \mathbf{N}.$$

It then follows that

$$U_k(\mathbf{N}) < \sum_{i \in \mathcal{V}(\mathbf{v})} \alpha_i U_i(\mathbf{N}) \leq \sum_{i \in \mathcal{V}(\mathbf{v})} \alpha_i U_{\mathbf{N}}^b = U_{\mathbf{N}}^b, \quad \forall \mathbf{N}.$$

□

We now turn our attention to the non-bottleneck stations. From (5) it is easy to see that every station $j$ is dominated by a station $i$ iff it is in the interior of a $R$-dimensional box, with a lower corner placed at the origin and the opposite upper corner corresponding to $i$. Therefore we can state the following proposition:

**Proposition 1.1. Dominated stations.** *All dominated stations are interior points of $T_\mathbf{L}$.*

Figure 2 shows that 5 is indeed dominated by 2, since it is contained in a box with right upper corner in 2. It is also straightforward that a station placed close to the origin would be dominated by multiple stations.

Let us consider now the masked-off stations. We introduce the following definition:

**Definition 2. Compound domination.** *A station $j \in \mathbf{M}$ is masked-off due to a compound domination of a group $\mathbf{C}$ of*

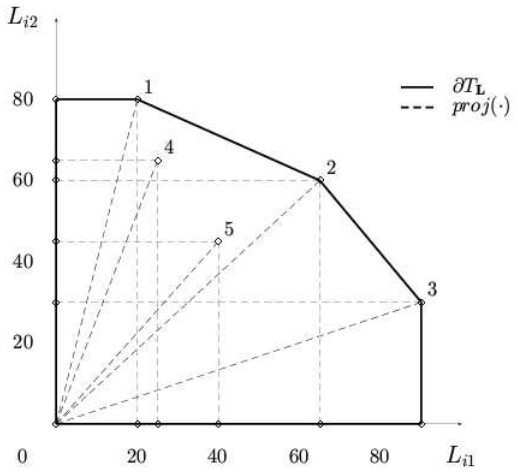


**Figure 3. Characteristic polytope $T_{\mathrm{L}}$ of the two-class loading matrix $\mathrm{L}$ (1).**

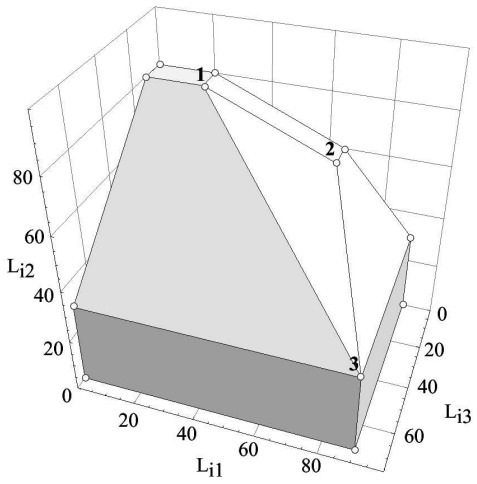


**Figure 4. Characteristic polytope of the five stations, three customer classes matrix (6).**

*stations if, for each $m \in \mathbf{C}$, there exists a $s \in \mathbf{R}$ such that*

$$U_{js}(\mathbf{N}) > U_{ms}(\mathbf{N}) \tag{7}$$

*but*

$$U_j(\mathbf{N}) < \sum_{i \in \mathbf{C}} \alpha_i U_i(\mathbf{N}), \quad \alpha_i = 1, \quad \alpha_i \geq 0 \tag{8}$$

*for all populations* $\mathbf{N}$.

This actually means that a station masked-off due to a compound domination is prevented from becoming a bottleneck by the mutual interaction of a group of stations. The following corollary is a direct consequence of Theorem 1:

**Corollary 1.1. Non-dominated interior points.** *All non-dominated stations in the interior of $T_{\mathbf{L}}$ are masked-off due to a compound domination.*

The most striking consequence of these results is that we can identify a large number of non-bottleneck stations by looking at the interior of $\partial T_{\mathbf{L}}$, which is computed using the loading matrix $\mathbf{L}$ only. Therefore, we can remove much of the complexity of a large-scale network, which may be composed by thousands of stations, focusing only on the stations lying on the boundary $\partial T_{\mathbf{L}}$. Moreover, assuming that the stations on $\partial T_{\mathbf{L}}$ are members of $\mathbf{\Phi} \cup proj(\mathbf{\Phi})$, we can fully identify the potential bottleneck set as the set of vertices of $\partial T_{\mathbf{L}}$ which are not projections of any other vertex. Using continuity arguments, we can prove this conjecture for at least all product-form networks. Please note that it would be computationally infeasible for the analyst to identify the set of all potential bottlenecks by running approximate algorithms on all possible populations. This holds true also for product-form open models: although the knowledge of the input workload $\boldsymbol{\lambda}$ and of the matrix $\mathbf{L}$ may suffice for computing all the utilizations using well-known non-iterative expressions, it turns out to be difficult to enumerate all possible bottlenecks with several customer classes without resorting to convex polytopes. Moreover, using the concept of domination and compound domination an ordering of station utilizations is possible using an approach similar to the proof technique of Theorem 1. In Section 3.3 we discuss existing algorithmic techniques to compute $\partial T_{\mathbf{L}}$.

### 3.2. Analysis of actual behavior using $\partial T_{\mathbf{L}}$

We now focus on the actual bottleneck stations. Theorem 1 states that all bottlenecks must lie on the boundary faces of $T_{\mathbf{L}}$. The following proposition gives additional informations concerning the actual bottlenecks.

**Proposition 2.1.** *If $d+1$ independent actual bottlenecks lie on the same face $\mathcal{F} \in \partial \mathcal{T}_{\mathbf{L}}$ with $\dim{}^1(\mathcal{F}) = d$, then all stations $\mathbf{L}_{\mathbf{w}} \in \mathcal{F}$ are actual bottlenecks.*

This raises the question whether, for a given $\mathbf{N}$, all actual bottlenecks lie on a same face[2] $\mathcal{F} \in \partial T_{\mathbf{L}}$ or not. This observation is confirmed by the following theorem:

**Theorem 2. Actual Bottlenecks.** *For a given population $\boldsymbol{N}$, all actual bottlenecks in $B_{\boldsymbol{N}}$ lie on the same face of $\partial T_{\mathbf{L}}$.*

1 The dimension $\dim(\mathcal{F})$ of a face $\mathcal{F}$ is the maximum number of independent vectors in the smallest affine space $S$ that contains $\mathcal{F}$.

2 We shall refer to faces that comprise only stations on their face, which can be easily recognized as the faces of the *upper convex hull* of $\mathbf{L} \cup proj(\mathbf{L})$ with strictly positive outward-pointing normals

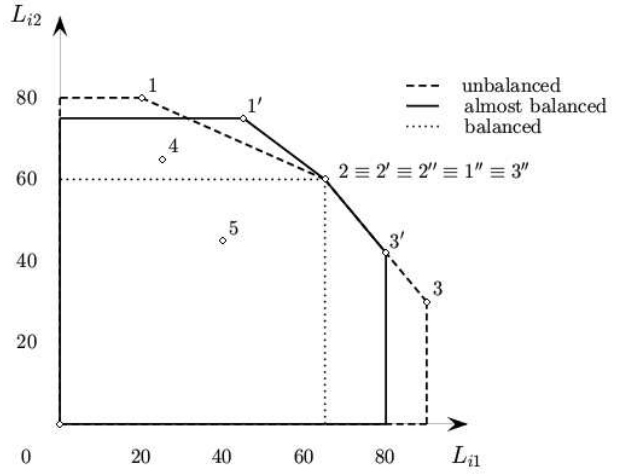


**Figure 5. Characteristic polytopes for different levels of load balance of the potential bottlenecks loadings.**

*Proof.* We prove the theorem by contradiction. Consider two faces $\mathcal{F}, \mathcal{G} \in \partial T_{\mathbf{L}}$ such that there is no face $\mathcal{H} \in \partial T_{\mathbf{L}}$, with $\mathcal{H} \neq \partial T_{\mathbf{L}}$, that connects them. We assume that two station $\mathbf{L}_1 \in \mathcal{F}$ and $\mathbf{L}_2 \in \mathcal{G}$ are both actual bottlenecks. Then we can always find a segment $\mathbf{L}_1 - \mathbf{L}_2$ whose internal points lie in the interior of $T_{\mathbf{L}}$. Let $\mathbf{v} \in \mathbf{L}_1 - \mathbf{L}_2$ be such a point and let $\mathbf{w}$ be a point of $\partial T_{\mathbf{L}}$ such that

$$\mathbf{w} > \mathbf{v} = \alpha_1 \mathbf{L}_1 + \alpha_2 \mathbf{L}_2, \quad \sum_{i=1}^{2} \alpha_i = 1, \quad \alpha_i \geq 0$$

Using equations (2) and (3) we get

$$\sum_{j \in \mathcal{V}(\mathbf{w})} \beta_j U_j(\mathbf{N}) > \sum_{i=1}^{2} \alpha_i U_i(\mathbf{N}) \qquad \sum_{j \in \mathcal{V}(\mathbf{w})} \beta_j = 1, \quad \beta_j \geq 0$$

and since

$$\alpha_1 U_1(\mathbf{N}) + \alpha_2 U_2(\mathbf{N}) = U_{\boldsymbol{N}}^b$$

we have

$$\sum_{j \in \mathcal{V}(\mathbf{w})} \beta_j U_j(\mathbf{N}) > U_{\boldsymbol{N}}^b$$

that is clearly a contradiction because $U_j(\mathbf{N}) \leq U_{\boldsymbol{N}}^b$ and $\beta_j \leq 1$. □

Thus, using Theorem 2 we can enumerate all possible bottleneck sets $B_{\boldsymbol{N}}$ by looking at the faces of $\partial T_{\mathbf{L}}$. This is quite interesting for a number of reasons. First, the number of sets $B_{\boldsymbol{N}}$ and the extension of the faces on which they lie measure the impact of high workload variabilities on the system under exam. To show this, let us consider a loading matrix $\mathbf{L}''$ derived from (1)

$$\mathbf{L}'' = \begin{bmatrix} 45 & 75 \\ 65 & 60 \\ 80 & 40 \\ 25 & 65 \\ 40 & 40 \end{bmatrix}$$

where the first three stations are more balanced that the corresponding of (1). Figure 5 compares the characteristic polytope of $\mathbf{L}''$ with that of (1). As the potential bottlenecks tend to be balanced, their relative distances decrease up to the ideal point where their loadings are perfectly balanced and the three stations collapse into a single degenerate face. This indicates that the extension of the faces without projected points can be an effective measure of the level of balance of the system. Moreover, since the cardinality $|\mathbf{\Phi}|$ depends on the number of faces $|\partial T_{\mathbf{L}}|$ of the characteristic polytope, $|\partial T_{\mathbf{L}}|$ estimates the number of possible bottleneck migrations. This can help, for instance, in a preliminary analysis of the QoS controllability of a complex Web application.

### 3.3. Computational complexity

Computing the boundary $\partial T_{\mathbf{L}}$ is a non-trivial task, addressed by the *convex hull problem* a fundamental problem in computational geometry with applications in several research fields. Unfortunately, an exact evaluation of its computational complexity is still an open (difficult) problem. It has been proved that the worst-case complexity, given the input size $M$ and the number of dimensions $R$, is $O(M^{\lfloor R/2 \rfloor})$ for fixed $R$ [7]. However, the average complexity depends on the number of computed faces, and thus depends on the choice of the algorithm. In [1] a comprehensive comparison of the most popular convex hull algorithms is presented. Two of them, namely `qhull` [3] and `cddf+` [11], are reported to achieve the best performances on non-degenerate polytopes: `qhull` has to be preferred in low dimensions ($R \leq 9$), while `cddf+` requires less time and memory in higher dimensions. Nevertheless, the complexity depends on the nature of the input data. In the experimental results section new benchmarking for the current versions of `qhull` and `cddf+` are presented.

Up to now we have denoted with $M$ the cardinality of the input set of the convex hull algorithm. This suggests that the set of input points corresponds to the set of stations $\mathbf{M}$. However, since the characteristic polytope is built using also the set of projections $proj(\mathbf{M})$, one may argue that the number of points is exponential in the number of dimensions $R$. However, such difficulty can be avoided by computing the convex hull of $\mathbf{M}$ only, and then keeping the faces of the upper convex hull with strictly positive outward-pointing normal vector, which correspond to the region of $\partial T_{\mathbf{L}}$ without projections (the only part of interest). Moreover, to reduce the complexity of computing

$\partial T_{\mathbf{L}}$, we propose several alternative options. First, dominated stations can be efficiently removed by means of specialized data structures or using sorting algorithms. Additionally, many interior points can be removed solving a collection of fast low-dimensional convex hull problems on a partition of $\mathbf{M}$. These tasks can significantly reduce the cardinality of the input set at a low computational cost. However, in high dimensional spaces ($R \geq 9$) an exact computation of $\partial T_{\mathbf{L}}$ is still very expensive. A fundamental observation here is that the complexity of the convex hull problem is mainly determined by the exact computation of the equations of the hyperplanes that define $\partial T_{\mathbf{L}}$, and therefore by the computation of the sets $B_{\boldsymbol{N}}$. In fact, if we wish to determine only the extreme points of $T_{\mathbf{L}}$, a superset of the potential bottlenecks set $\boldsymbol{\Phi}$, the computational requirements reduce approximately to $O(M \cdot |\Phi|)$ time and $O(M)$ space [15]. More precisely, what we need here is again to determine the extreme points lying in the upper convex hull of $\mathbf{L}$ with strictly positive outward-pointing normal vector. To do so, we introduce the following efficient algorithm:

**Algorithm 1.** *Redundancy Elimination Problem (REP)*
$\boldsymbol{\Phi} = \{\}$;
$\mathbf{M}' = \mathbf{M}$;
$\mathbf{x} =$ a point of $\mathbf{M}'$;
**let** REP($\mathbf{x}$) be the LP feasibility problem:
$\mathbf{x} \leq \sum_i \alpha_i \mathbf{L}_i; \quad \sum_i \alpha_i = 1, \quad \alpha_i \geq 0 \quad \forall i \in \mathbf{M}'$;
**for** each point $\mathbf{p} \in \mathbf{M}'$
  **if** REP($\mathbf{p}$) is feasible **then**
    $\mathbf{M}' = \mathbf{M}' \setminus \mathbf{p}$;
  **else**
    $\boldsymbol{\Phi} = \boldsymbol{\Phi} \cup \mathbf{p}$;
  **end if**
**end for**
**return** $\boldsymbol{\Phi}$;
**end algorithm**

which has a worst-case time complexity of $O(M \cdot lp(M, R))$, where $lp(M, R)$ is the (polynomial) complexity of solving a linear program with $M$ rows and $R$ columns. However, since we require only to check the feasibility of $M$ redundancy elimination problems, this can be done efficiently using the first-phase of a two-phase implementation of the simplex algorithm. In the experimental results we show numerical evidence that the complexity is polynomial in $R$ for fixed $M$.

## 4. Application to asymptotic analysis of closed product-form networks

Let $\boldsymbol{N} = \{N_r\}$ be the actual population in the network. We can always express such vector as the scalar product $\boldsymbol{N} = N \cdot \boldsymbol{\beta}$, where $N = \sum_{r \in \mathbf{R}} N_r$ is total population in the network, and $\boldsymbol{\beta} = \{\beta_r = N_r/N\}$ is the *population mix*. Using the approach described in [2], it is possible to derive $B_{\boldsymbol{N}}$ as a function of the population mix $\boldsymbol{\beta}$ assuming that the population $N$ tends to infinity. In particular, using the following algorithm, it is possible to map a connected set of $\boldsymbol{\beta}$s (*saturation sector*) to a unique bottleneck set $B_{\boldsymbol{N}}$:

**Algorithm 2.** *Saturation Sectors Identification*
**let** $\mathbf{S} = \{\}$ be the set of saturation sectors;
**let** $s = R$ be the rank of the loading matrix $L$
  **let** $\mathbf{m} = (m_1, \ldots, m_s)$ be a set of $s$ stations of $\mathbf{M}$;
  **let** $LS_2(\mathbf{m})$ be the linear system:

$$\sum_{r \in \mathbf{R}} U_{ir}(\boldsymbol{\beta}) = 1 \qquad i \in \mathbf{m} \tag{9}$$

$$L_{jr} U_{ir}(\boldsymbol{\beta}) = L_{ir} U_{jr}(\boldsymbol{\beta}) \quad i, j \in \mathbf{m} : j \neq i \tag{10}$$

  with fixed $j$ for each $r$.

  **for** any feasible $\mathbf{m} \subseteq \mathbf{M}$ w.r.t. $S$
  $\mathbf{U_m} = LS_2(\mathbf{m})$
    **let** $\boldsymbol{\beta}*$ be the set of $s$ vertices of a saturation sector;
    **if** $\boldsymbol{\beta}*$=$\mathbf{U_m}$ is feasible w.r.t. $\mathbf{S}$
      $\mathbf{S} = \mathbf{S} \cup \boldsymbol{\beta}*$;
    **else**
      solve the conflict and update $\mathbf{S}$;
    **end if**
  **end for**
Add to $\mathbf{S}$ the remaining sectors with $s < R$ saturating stations
**return** $\mathbf{S}$;
**end algorithm**

Algorithm 2 works efficiently only for low values of $R$ and $M$, since the number of possible sets $\mathbf{m}$ grows combinatorially as $R$ increases. However, using the result of Theorem 2 we introduce the following modification that allows the algorithm to scale as $R$ and $M$ increase:

**Algorithm 3.** *Scalable Saturation Sectors Identification*
**let** $\mathbf{T_L}$ be the characteristic polytope of $\mathbf{L}$;
**let** $\mathbf{S} = \{\}$ be the set of saturation sectors;
**let** $\mathbf{m}(\mathcal{F}) = (m_1, \ldots, m_R)$ be the set of $R$ stations of the facet $\mathcal{F}$;
**let** $LS_3(\mathcal{F})$ be the linear system:

$$\sum_{r \in \mathbf{R}} U_{ir}(\boldsymbol{\beta}) = 1 \qquad i \in \mathbf{m}(\mathcal{F}) \tag{11}$$

$$L_{jr} U_{ir}(\boldsymbol{\beta}) = L_{ir} U_{jr}(\boldsymbol{\beta}) \quad i, j \in \mathbf{m}(\mathcal{F}) : j \neq i \tag{12}$$

and $j$ is fixed for each $r$.

  **for** any facet $\mathcal{F} \in \partial \mathbf{T}$
  $\mathbf{U}_{\mathcal{F}} = LS_3(\mathcal{F})$
    **let** $\boldsymbol{\beta}*$ be the set of $R$ vertices of a saturation sector;
    $\boldsymbol{\beta}*$=$\mathbf{U}_{\mathcal{F}}$ $\mathbf{S} = \mathbf{S} \cup \boldsymbol{\beta}*$;
  **end for**
  **return** $\mathbf{S}$;
  Add to $\mathbf{S}$ the remaining sectors with $s < R$ saturating stations
**end algorithm**

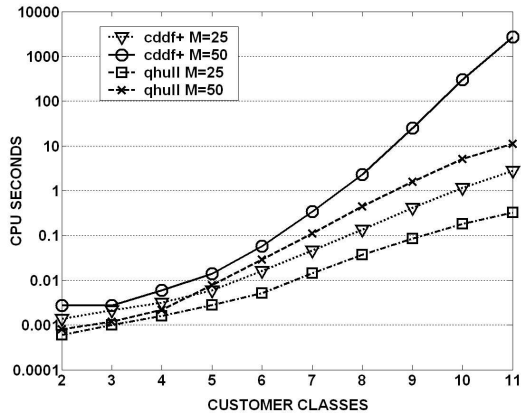


**Figure 6. Timing results for computing the convex hull for different cardinalities of $\mathrm{M}$ using different algorithms.**

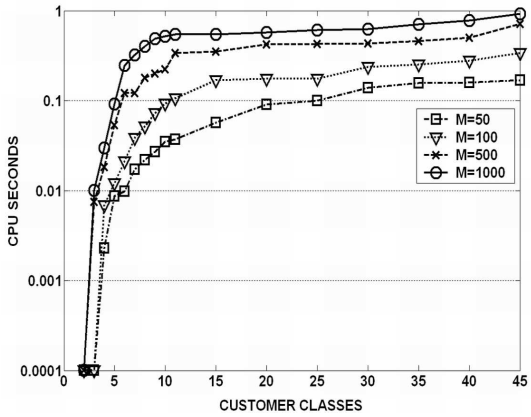


**Figure 7. Timing results for Algorithm 1 for different cardinalities of $\mathrm{M}$.**

The rationale behind our modification is the following: since Theorem 2 states that each bottleneck set $B_N$ must lie on a same face of $\mathbf{T_L}$, we limit the enumeration of the sets $\mathbf{m}$ by looking at $\partial\mathbf{T_L}$. As we shall see in the following section, the effect of our modification on the scalability of the saturation sectors identification algorithm is remarkable.

## 5. Experimental Results

Let us now focus on the numerical results of the proposed algorithms. First, let us consider the choice of the convex hull algorithm. As reported in Section 3.3, previous work have shown that `cddf+` (with the `lexmin` option) has to be preferred to `qhull` for high dimensional data. Since the experiments in literature were conducted on old versions of the two applications, we checked the validity of the results for the current versions using the following experimental setup: we generated a testbed of $50$ random models for each value of the number of stations ($25$ stations and $50$ stations), and for a number of customer classes ranging from 2 to 11. All the points were generated with non-negative coordinates using a uniform distribution $U(0, 1000)$. The number of instances was limited to fifty since the variance in computation times was negligible. We tested on a Athlon XP 1800 (1400 Mhz - 512KB cache) with 1GB of RAM and 80Gb HD, running Linux Fedora Core release 1 (kernel 2.4.22). The versions of the two programs were `qhull2003.1` and `cddf+0.77`. Both applications are available for free on the World Wide Web. Times were measured in average CPU seconds. Experimental results are plotted in Figure 6 in logarithmic scale. Surprisingly, `qhull` scales better than `cddf+` as the number of dimensions grows. However, the validity of our benchmarks is limited to the specific nature of our dataset that comprised also degenerate faces. As expected, instead, the measures show that both `qhull` and `cddf+` execution times grow exponentially as the number of classes grows. It is important to remark that also `qhull` perfomance quickly degenerate in higher dimensions: for instance, running `qhull` on a model with fifty stations and fifteen classes may require more than `1GB` of memory usage. When the memory limit yields poor performances, `cddf+` has in general to be preferred. In fact, for the same instance with fifty stations and fifteen classes, `cddf+` used only few tens of megabytes. The interested reader should refer to [1, 3, 11] for additional informations.

Let us now consider the numerical results of Algorithm 3. Let $LS_2$ and $LS_3$ be the mean number of linear systems solved (on the same input set) by Algorithm 2 and Algorithm 3, respectively. Let

$$\Delta_{23} = \frac{LS_3 - LS_2}{LS_2}$$

be the normalized reduction, in terms of number of linear systems to be solved, when Algorithm 3 is used instead of Algorithm 2. In this case, the testbed was composed by $100$ random models generated using a uniform $U(0, 1000)$.

Table 1 shows the mean values of $LS_2$ and $LS_3$ measured from the experimental testbed. The values of $\Delta_{23}$ show that Algorithm 3, compared to Algorithm 2, allows a sensible reduction on the number of linear system to be solved for the determination of the edges of the saturation sectors. More precisely, Algorithm 3 is optimal since it requires only to compute the linear system which yield a feasible saturation sector. However, it requires the computation of $T_{\mathbf{L}}$, and thus it cannot be applied in high dimensional spaces. Finally, Figure 7 presents the experimental

| $R = 3$ | $LS_2$ | $LS_3$ | $\Delta_{23}$ |
|---|---|---|---|
| $M = 10$ | $1.08 \cdot 10^2$ | $4.25 \cdot 10^1$ | $-60.65\%$ |
| $M = 100$ | $1.91 \cdot 10^3$ | $1.73 \cdot 10^2$ | $-90.94\%$ |
| $M = 1000$ | $9.71 \cdot 10^4$ | $3.90 \cdot 10^2$ | $-99.60\%$ |
| $R = 4$ | $LS_2$ | $LS_3$ | $\Delta_{23}$ |
| $M = 10$ | $2.97 \cdot 10^2$ | $1.14 \cdot 10^2$ | $-61.62\%$ |
| $M = 100$ | $3.32 \cdot 10^4$ | $1.12 \cdot 10^3$ | $-96.63\%$ |
| $M = 1000$ | $3.84 \cdot 10^5$ | $3.95 \cdot 10^3$ | $-98.97\%$ |

**Table 1. Comparison of the mean number of linear systems solved by Algorithms 2 and 3**

results of the REP algorithm. Compared to the results of the convex hull algorithms shown in Figure 6, the algorithm scales easily as the number of classes grows. The slope of the curves show that the complexity is clearly polynomial for a fixed number of stations $M$. Moreover, since the average CPU time requires less than one second in all cases, Algorithm 1 could be successfully employed in real-time applications. An open issue is whether the principle behind the REP can be applied for simplifying the computation of the convex hull problem.

## 6. Conclusions

In this paper we have shown how the theory of convex polytopes can be applied to the bottleneck analysis of multiclass queueing networks. We defined efficient algorithms to study large-scale models comprising thousands of servers and several tens of customer classes, all serving an arbitrary number of customers. Our algorithms allow to compute the set of potential bottlenecks in a network with one thousand servers and fifty customer classes in few seconds. They also allow to understand graphically the behavior of models comprising up to three distinct customer classes. The experimental results show that additional informations concerning the actual bottlenecks of product-form networks can be obtained efficiently if the parametrization of the model does not exceed $10 - 11$ customer classes.

## 7. Acknowledgements

This work has been partially supported by the Italian FIRB-Perf project (http://www.perf.it).

## References

[1] Avis, D., Bremner, D., & Seidel, R. (1997). How good are convex hull algorithms. *Computational Geometry: Theory and Applications*,7,265-302.

[2] Balbo, G., & Serazzi, G. (1997). Asymptotic analysis of multiclass closed queueing networks: Multiple bottlenecks. *Perform. Eval. 30(3)*, 115–152.

[3] Barber, C. B., Dobkin, D. P., & Huhdanpaa, H. T. (1996). The Quickhull algorithm for convex hulls. *ACM Trans. on Mathematical Software*, *22(4)*,469–483.

[4] Buzen, J. P. (1973). Computational algorithms for closed queueing networks with exponential servers. *Commun. ACM*,*16(9)*, 527–531.

[5] Casale, G. & Serazzi, G. (2003). Estimating Bottlenecks of Very Large Models. *Performance Evaluation : Stories and Perspectives*, (pp. 89–104) . Austrian Computing Group (OGC) Press, December 2003.

[6] Chandy, K. M., Neuse, D. (1982). Linearizer: A heuristic algorithm for queuing network models of computing systems. *Commun. ACM*, *25(2)*, 126–134.

[7] Chazelle, B. (1993). An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10,377-409.

[8] Cremonesi, P., Schweitzer P. J. & Serazzi, G. (2002). A unifying framework for the approximate solution of closed multiclass queueing networks. *IEEE Trans. Comp.*,*16(9)*, 527–531.

[9] Denning, P. J., & Buzen, J. P. (1978). The Operational Analysis of Queueing Network Models. *ACM Computing Surveys*,*10(3)*, 225–261.

[10] Eager, D. L., Sevcik, K. C. (1986). Bound hierarchies for multiple-class queuing networks.*Journal of the ACM*. *33(1)*, 179–206.

[11] Fukuda, K. (1999). `cdd/cdd+` reference manual, version 0.61/0.76. EPF Lausanne, Switzerland. `http://www.cs.mcgill.ca/~fukuda/`.

[12] Kerola, T. The Composite Bound Method for Computing Throughput Bounds in Multiple Class Environments. *Perform. Eval. 6(1)*, 1–9.

[13] Knessl, C. & Tier, C. (1998). Asymptotic approximations and bottleneck analysis in product form queueing networks with large populations. *Perform. Eval. 33(4)*, 219–248.

[14] Lazowska, J., Zahorjan, J., Eager, D. L. & Galler, B. (1982). Balanced Job Bound Analysis of Queueing Networks. *Commun. ACM*. *25(2)*, 134–141.

[15] Ottmann, T., Schuierer, S. & Soundaralakshmi, S. (1995). Enumerating extreme points in higher dimensions. *Proc. of the 12th Annual Symp. on Theoretical Aspects of Comp. Science - LNCS 900* (pp. 562-570). New York: Springer-Verlag.

[16] Reiser, M., & Lavenberg, S.S. (1980). Mean-value analysis of closed multichain queueing networks. *Journal of the ACM*, 27(2),312–322.

[17] Schweitzer, P. J. (1992). A fixed-point approximation to product-form networks with large populations. *Presented at 2nd ORSA Telecomm. Conf., Boca Ration, FL.*

[18] Zahorjan, J., Eager, D. L. & Sweillam, H.M. (1988). Accuracy, Speed, and Convergence of Approximate Mean Value Analysis. *Perform. Eval. 8(4)*, 255–270.