CoMoM: Class-Oriented Evaluation of Multiclass Models

Giuliano Casale
College of William and Mary
Department of Computer Science
Williamsburg, VA, 23187-8795
casale@cs.wm.edu

1. INTRODUCTION

We introduce the Class-Oriented Method-of-Moments (CoMoM), a new efficient algorithm for evaluating normalizing constants and *marginal* queue-length probabilities in closed multiclass queueing networks [1] used in capacity planning of Web applications. CoMoM is motivated by the need of computing efficiently marginal queue-length probabilities required to estimate *energy consumption metrics*. For example, modern Web servers are able to dynamically re-scale their CPU frequency in order to minimize energy consumption if the local concurrency level of requests is small; this technique is known as dynamic voltage scaling (DVS). The evaluation of energy consumption with DVS technologies requires to quantify this concurrency level in terms of marginal queue-length probabilities at each Web server, but these cannot be evaluated efficiently in multiclass model using Mean Value Analysis (MVA) or Approximate MVA (AMVA) methods [2], since MVA and AMVA only return mean indexes and not probabilities. Load-dependent algorithms may be used, but their computational costs are prohibitive in multiclass networks. Thus, there is a lack of techniques that could effectively address the estimation of energy consumption metrics in a practical way.

CoMoM defines an efficient *exact* solution scheme for normalizing constants and marginal queue-length probabilities based on a *simple* recursive computation of conditional queue-length moments. It is easy to find cases where CoMoM is more than ten orders of magnitude faster and less memory consuming than MVA. Further, CoMoM is extremely efficient for solving models with several service classes, besides being the only available method for efficiently evaluating marginal probabilities.

The next sections are organized as follows. After giving required background in Section 1.1, we introduce CoMoM by example in Section 2. The general algorithm is outlined in Section 3, where we also discuss the computation of marginal queue-length probabilities. Section 4 gives conclusions and outlines future work.

1.1 Analytical Model

Queueing Networks: We focus on the closed product-form multiclass queueing networks introduced in [1]. The model has population $\vec{N} \equiv (N_1, \ldots, N_r, \ldots, N_R)$, R is the number of service classes, N_r is the number of jobs of class r. The class-r jobs visit M constant-rate queues having mean service demand $D_{k,r}$ (service time $\times$ visit ratio); Z_r is the think time of class-r jobs. The probability of state $\vec{n} = (n_{0,1}, \ldots, n_{k,r}, \ldots, n_{M,R})$, is [1]

$$P(\vec{n}|\vec{N}) = \frac{1}{G(\vec{N})} \left(\prod_{r=1}^{R} \frac{Z_r^{n_{0,r}}}{n_{0,r}!} \right) \left(\prod_{k=1}^{M} \left(\sum_{r=1}^{R} n_{k,r} \right)! \prod_{r=1}^{R} \frac{D_{k,r}^{n_{k,r}}}{n_{k,r}!} \right),$$

where the normalizing constant $G(\vec{N})$ imposes $\sum_{\vec{n}} P(\vec{n}|\vec{N}) = 1$.

Normalizing Constants: CoMoM recursively computes $G(\vec{N})$ and marginal probabilities obtained by summation of $P(\vec{n}|\vec{N})$. The method is defined upon existing recurrences for normalizing constants, i.e., the *convolution expression* (CE) [2] in the form

$$G(\vec{1}_k, \vec{N}) = G(\vec{N}) + \textstyle\sum_{r=1}^{R} D_{k,r} G(\vec{1}_k, \vec{N} - \vec{1}_r), \qquad (1)$$

for arbitrary $1 \leq k \leq M$, where, e.g., $\vec{1}_2 = (0, 1, 0, \ldots, 0)$, and $G(\vec{1}_k, \vec{N} - \vec{1}_r)$ is the normalizing constant of a model with a job of class-r removed and including an additional queue identical to queue k. It is possible to show that adding one or more replicas of queue k has the probabilistic interpretation of computing conditional (binomial) moments of station k queue-length in the initial model. CoMoM also uses the *population constraint* (PC)

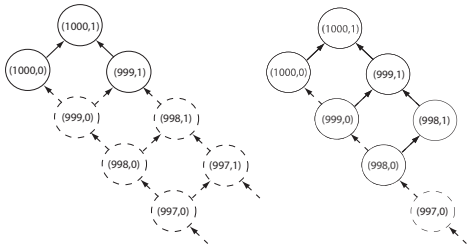
$$N_r G(\vec{N}) = Z_r G(\vec{N} - \vec{1}_r) + \textstyle\sum_{k=1}^{M} D_{k,r} G(\vec{1}_k, \vec{N} - \vec{1}_r), \quad (2)$$

for arbitrary $1 \leq r \leq R$, which can be shown to be equivalent to the RECAL recurrence equation [3]. Mean value indexes such as the mean class-r throughput $X_r(\vec{N})$ and the mean class-r queue-length $Q_{k,r}$ at queue k are obtained immediately from normalizing constants as $X_r(\vec{N}) = G(\vec{N} - \vec{1}_r)/G(\vec{N})$, and $Q_{k,r}(\vec{N}) = D_{k,r} G(\vec{1}_k, \vec{N} - \vec{1}_r)/G(\vec{N})$, respectively [2]. Response times and utilizations are easily computed from throughputs and queue-lengths using Little's Law [2].

2. ILLUSTRATIVE EXAMPLE

To build intuition on the CoMoM approach and compare it with the standard MVA recursion[1], we solve by CoMoM a small network with $R = 2$ classes and $M = 2$ queues. In particular, we focus on the computation of the normalizing constant for the population $(1000, 1000)$, and we initially discuss the critical intermediate recursive step for $\vec{N} = (1000, 1)$, which is the first population that cannot be evaluated by basic single class algorithms. The

[1]In order to perform a comparison using normalizing constants, we here implicitly refer to the LBANC algorithm, i.e., the *unnormalized* version of MVA which computes normalizing constants instead of mean indexes (see [3] for a review).



(a) MVA (b) CoMoM

Figure 1: Illustrative Example. ***Recursive structure of MVA (figure (a)) and CoMoM (figure (b)) in a neighborhood of*** $\vec{N} = (1000, 1)$. ***Dashed states indicate populations of models solved previously by the recursion and not involved in the current recursive step; solid arrows indicate relations (e.g., (1)-(2)) that are exploited by the algorithm in the current step of the recursion.***

evaluation of marginal queue-length probabilities is similar and it is discussed in Section 3.2. A graphical representation of the recursive structure of MVA in a neighborhood of $\vec{N} = (1000, 1)$ is given in Figure 1(a). MVA reaches $(1000, 1)$ in a bottom-up approach, evaluating step-by-step models with populations summing to $N = N_1 + N_2 = 1, \ldots, 999, 1000, 1001$. The recursive step for $N = 1001$ is depicted in Figure 1(a), see the caption for a description. Conversely, Figure 1(b) illustrates the approach of the CoMoM recursion. CoMoM computes $\vec{N} = (1000, 1)$ directly from the solutions of the single class models with populations $(1000, 0)$, $(999, 0)$, and $(998, 0)$, which avoids evaluating almost half of the populations, i.e., the states $(1, 1), (2, 1), \ldots, (997, 1)$ are all *skipped* by CoMoM. On larger models, while the number of states evaluated by MVA grows *combinatorially* as the set of all possible populations component-wise less than or equal to $\vec{N}$, the set of states spanned by CoMoM grows *linearly* with the total population size N, since the algorithm can recursively reapply the same step depicted in Figure 1(b) for all populations. This efficient schema is made possible by simultaneously evaluating CE's and PC's in a linear system. A comparison with the radically different queue-addition technique proposed in [3], that also employs CE's and PC's, is given in [4].

In order to derive CoMoM, let us consider the state $\vec{N} = (1000, 1)$ and assume that we want to compute the set of normalizing constants $\lambda(1000, 1) = \{G(\vec{1}_1, 1000, 1), G(\vec{1}_2, 1000, 1), G(1000, 1)\}$. Further assume to know all normalizing constants in the similarly-defined vectors $\lambda(1000, 0)$, $\lambda(999, 0)$, $\lambda(998, 0)$. These vectors are easily computed by efficient single-class convolution and thus we assume in the analysis to be known values. The CoMoM approach is explained by the following observations:

Observation 1: Using the PC (2) for $r = 2$, we can immediately compute $G(1000, 1)$ from the normalizing constants in $\lambda(1000, 0)$ which are assumed known. Thus, the remaining unknowns are

$$\{G(\vec{1}_1, 1000, 1), G(\vec{1}_2, 1000, 1)\} \subset \lambda(1000, 1). \quad (3)$$

Observation 2: The constants (3) can *only* be computed by the CE's (1). In fact, the PC's in this case would require the availability of constants with two queue replicas, e.g., $G(1_1 + 1_2, 999, 1)$, not included in the vectors λ. Therefore, to evaluate $\lambda(1000, 1)$ we need also the constants in $\lambda(999, 1)$. Thus, we have to expand the set of unknowns by including also the vector $\lambda(999, 1)$, therefore the set of unknowns becomes

$$\{G(\vec{1}_1, 1000, 1), G(\vec{1}_2, 1000, 1)\} \subset \lambda(1000, 1), \quad (4)$$

$$\{G(\vec{1}_1, 999, 1), G(\vec{1}_2, 999, 1)\} \subset \lambda(999, 1), \quad (5)$$

where $G(999, 1)$ is omitted being easily computed by the PC for $r = 2$ from the known constants in $\lambda(999, 0)$.

Observation 3: We now note that the unknowns (4) are related to the unknowns (5) by two CE's for $k = 1, 2$, where the other constants appearing in the formulas are all known. Further, the unknowns (5) are related to each other by the PC for $r = 1$. Therefore, we have moved from the initial problem (3) with $p = 2$ unrelated unknowns to a new problem with $p' = 4$ unknowns (4)-(5) which are instead related by $q' = 3$ equations (1)-(2). This reduces the degrees of freedom df of the problem from $df = p = 2$ of (3) to $df = p' - q' = 4 - 3 = 1$ of (4)-(5). That is, *the extension of the analysis to the vector* $\lambda(999, 1)$ *in (4)-(5) has reduced the degrees of freedom in determining the normalizing constants*. This is a counter-intuitive property of multiclass models, since we are stating that increasing the difficulty of the problem by adding to the set of values to be computed also the moments represented by the constants in $\lambda(999, 1)$ has the opposite effect of reducing the computational costs and thus eventually simplifies the solution.

Observation 4: Using the last observation, we can further extend the analysis to include as unknown also $\lambda(998, 1)$ in addition to $\lambda(1000, 1)$ and $\lambda(999, 1)$. Using the same arguments given above, it is possible to conclude that the new problem has $p'' = 6$ unknowns related by $q'' = 6$ linear equations (1)-(2) and therefore the degrees of freedom become $df = p'' - q'' = 0$, which allows the exact computation of all unknowns by inverting a linear system of equations (1)-(2).

Summarizing, starting from the vectors $\lambda(1000, 0)$, $\lambda(999, 0)$, and $\lambda(998, 0)$, CoMoM has computed $\lambda(1000, 1)$, $\lambda(999, 1)$, and $\lambda(998, 1)$. These vectors provide the new initial conditions for reapplying recursively the same scheme to models with $N_2 \geq 2$ jobs. Thus, we can easily reach the final model solution for the population $(1000, 1000)$ after 1000 recursive steps on class 2. Conversely, MVA, after computing the solutions for the single-class states, still requires 9.9×10^5 evaluations to reach $(1000, 1000)$. A definition of the CoMoM algorithm for general models is given in the next section.

3. GENERAL ALGORITHM

We propose two definitions of the CoMoM algorithm. The first version in Section 3.1 follows the recursive steps outlined in the previous section and is sufficient to compute normalizing constants. Whenever the marginal queue-length probabilities are of interest, a slightly more complex recursion should be adopted, which is discussed in Section 3.2. In order to distinguish between the two implementations, we refer to the two algorithms respectively as the CoMoM and the probabilistic CoMoM (pro-CoMoM) algorithms.

3.1 CoMoM: Normalizing Constants

Following the illustrative example in the previous section, we give a general definition of CoMoM. We specify the set of unknown normalizing constants to be evaluated at each step of the recursion, henceforth referred to as the *base* of CoMoM.

DEFINITION 1. *Consider a model with M queues, R classes, and population $\vec{N}$, $N_r \geq 1$, $1 \leq r \leq R$. The* base $\Lambda(\vec{N})$ *is the set*

$$\Lambda(\vec{N}) = \lambda(\vec{0}) \cup \lambda(\vec{1}_1) \cup \ldots \cup \lambda(\vec{1}_M), \quad (6)$$

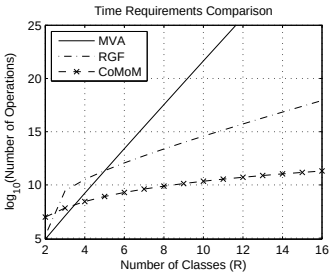


Figure 2: CoMoM Time Requirements. Comparison with MVA and RGF of time requirements for a model with $M = 3$ queues and $N_r = 100$ jobs for each class.

where $\lambda(\Delta\vec{m}) = \{G(\Delta\vec{m}, \vec{N} - \vec{n}) \mid \sum_{r=1}^{R-1} n_r \leq M \wedge n_R = 0\}$ are the normalizing constants for all possible populations having up to M jobs less than $\vec{N}$ chosen among the first $R-1$ classes.

The CoMoM recurrence equation is introduced in the next theorem.

THEOREM 1. *The base (6) satisfies the linear matrix recurrence*

$$\bar{\mathbf{A}}(\vec{N})\Lambda(\vec{N}) = \bar{\mathbf{B}}(\vec{N})\Lambda(\vec{N} - 1_R), \tag{7}$$

where $\bar{\mathbf{A}}(\vec{N})$ and $\bar{\mathbf{B}}(\vec{N})$ are square matrices of order $card(\Lambda(\vec{N})) = (M+1)\binom{M+R-1}{M}$, defined by the set of all equations (1)-(2) that relate only normalizing constants in $\Lambda(\vec{N})$ and $\Lambda(\vec{N} - 1_R)$.

A proof of the theorem can be found in the technical report [4]. The CoMoM algorithm is immediately defined by recursively solving (7) as a system of linear recurrence equations; the recursion is carried out on the right hand side terms up to the population $\vec{N} = \vec{0}$. The computational cost thus grows as $O(N)$, $N = N_1 + \ldots + N_R$, with respect to the population size, since the order of the matrices $\bar{\mathbf{A}}(\vec{N})$ and $\bar{\mathbf{B}}(\vec{N})$ is independent of $\vec{N}$. However, numerical stabilization required to deal with the scaling of the normalizing constants and possibly ill-conditioned matrices $\bar{\mathbf{A}}(\vec{N})$ raise the computational cost in general to $O(N^2 \log N)$ if exact linear algebra is used to perform this stabilization. Whenever $\bar{\mathbf{A}}(\vec{N})$ is singular, it is possible to reformulate the recursion to solve the model with increased costs, but still much more efficiently than with MVA [4].

A plot comparing the computational costs of CoMoM with the MVA algorithm and with RGF [6], which is the best-available algorithm for models with large number of classes, is given in Figure 2. Figure 2 suggests that CoMoM is much faster than MVA and RGF in the evaluation of models with several classes. The memory requirements of both CoMoM and RGF are feasible on existing architectures, with RGF being more efficient than CoMoM. On larger queueing networks, the computational requirements of CoMoM with respect to existing methods are typically the best available, both in terms of time and space, see [4].

3.2 pro-CoMoM: Marginal Probabilities

The pro-CoMoM algorithm computes the marginal probabilities

$$\widetilde{P}_k(n \mid \vec{N}) = P_k(n \mid \vec{N})G(\vec{N}) = \textstyle\sum_{\vec{n}:n_k=n} P(\vec{n} \mid \vec{N})G(\vec{N}), \tag{8}$$

$0 \leq n \leq N$, which can be normalized by the constant $G(\vec{N}) = \sum_{n=0}^{N} \widetilde{P}_k(n \mid \vec{N})$. The fundamental property of (8) is to satisfy recurrence equations that are similar to the CE's and PC's.

THEOREM 2. *The un-normalized probabilities $\widetilde{P}_k(n \mid \vec{N})$, $1 \leq n \leq N$, satisfy the probabilistic convolution expression (PCE)*

$$\widetilde{P}_k(n \mid \vec{1}_j, \vec{N}) = \widetilde{P}_k(n \mid \vec{N}) + \textstyle\sum_{r=1}^{R} D_{j,r}\widetilde{P}_k(n \mid \vec{1}_j, \vec{N} - \vec{1}_r),$$

for all $1 \leq n \leq N$, $1 \leq k \leq M$, $1 \leq j \leq M$, $j \neq k$, and the probabilistic population constraint (PPC)

$$N_r\widetilde{P}_k(n \mid \vec{N}) = Z_r\widetilde{P}_k(n \mid \vec{N} - \vec{1}_r) + nD_{k,r}\widetilde{P}_k(n-1 \mid \vec{N} - \vec{1}_r) + \textstyle\sum_{j \neq k} D_{j,r}\widetilde{P}_k(n \mid \vec{1}_j, \vec{N} - \vec{1}_r),$$

for arbitrary $1 \leq r \leq R$, where $\widetilde{P}_k(n \mid \vec{1}_j, \vec{N})$ and $\widetilde{P}_k(n-1 \mid \vec{1}_j, \vec{N} - \vec{1}_r)$ refer to models with a replica of queue j more.

From the above property, it is simple to define a linear recursion for marginal probabilities that is similar to the CoMoM recursion (7).

THEOREM 3. *The probabilistic base*

$$\Pi_k(n \mid \vec{N}) = \pi_k(n, \vec{0}) \cup_{1 \leq j \leq M \wedge j \neq k} \pi_k(n, \vec{1}_j),$$

$\pi_k(n, \Delta\vec{m}) = \{\widetilde{P}_k(n \mid \Delta\vec{m}, \vec{N} - \vec{n}) \mid \Delta\vec{m} \neq \vec{1}_k \wedge \sum_{r=1}^{R-1} n_r \leq M \wedge n_R = 0\}$ satisfies the matrix recurrence $\bar{\mathbf{A}}_k(n \mid \vec{n})\Pi_k(n \mid \vec{n}) = \bar{\mathbf{B}}_k(n \mid \vec{n})\Pi_k(n \mid \vec{n} - \vec{1}_r) + \bar{\mathbf{C}}_k(n \mid \vec{n})\Pi_k(n-1 \mid \vec{n}) + \bar{\mathbf{D}}_k(n \mid \vec{n})\Pi_k(n-1 \mid \vec{n} - \vec{1}_r)$, where the coefficient matrices are defined by all PCE's and PPC's relating only the four probabilistic bases.

The pro-CoMoM algorithm is defined by the recursive computation of the probabilistic base $\Pi_k(n \mid \vec{N})$ using the linear matrix recurrence defined above. The computational costs now grow as $O(N^2)$, and accounting for the exact algebra overheads this becomes $O(N^3 \log N)$. This is several orders of magnitude more efficient than load-dependent algorithms which grow in complexity as $O((N_1 N_2 \cdots N_R)^2)$, e.g., this grows as $O(N^4)$ for a model with only $R = 2$ classes. For instance, using pro-CoMoM we were able to solve models with up to $R = 5$ customer classes and hundreds of jobs which are infeasible to evaluate with the MVA. We point to [4] for case studies showing the applicability of pro-CoMoM to the evaluation of energy consumption.

4. CONCLUSION

We have introduced CoMoM, a new algorithm for multiclass queueing networks that is orders of magnitude more efficient than MVA. CoMoM can recursively compute marginal queue-length probabilities in multiclass models: these values are required by state-dependent cost indexes such as energy consumption metrics. Future work will focus on releasing an open source implementation of the tool and on analyzing by CoMoM joint queue-length probabilities such as in the DAC algorithm [5].

5. REFERENCES

[1] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *JACM*, 22(2):248–260, 1975.

[2] G. Bolch et al. *Queueing Networks and Markov Chains*. John Wiley & Sons, 1998.

[3] G. Casale. An efficient algorithm for the exact analysis of multiclass queueing networks with large population sizes. In *Proc. of joint ACM SIGMETRICS/IFIP Performance 2006*, pages 169–180. ACM Press, 2006.

[4] G. Casale. CoMoM: A Class-Oriented Algorithm for Probabilistic Evaluation of Multiclass Queueing Networks. Technical Report WM-CS-2008-04, Department of Computer Science, College of William and Mary, 2008.

[5] E. De Sousa e Silva and S. S. Lavenberg. Calculating joint queue-length distributions in product-form queueing networks. *JACM*, 36(1):194–207, 1989.

[6] P. G. Harrison, S. Coury. On the asymptotic behaviour of closed multiclass queueing networks. *PEVA*, 47(2):131–138, 2002.