Comm. Korean Math. Soc. 10 (1995), No. 3, pp. 735–750

A DOMAIN DECOMPOSITION METHOD APPLIED TO QUEUING NETWORK PROBLEMS

PIL SEONG PARK

ABSTRACT. We present a domain decomposition algorithm for solving large sparse linear systems of equations arising from queuing networks. Such techniques are attractive since the problems in subdomains can be solved independently by parallel processors. Many of the methods proposed so far use some form of the preconditioned conjugate gradient method to deal with one large interface problem between subdomains. However, in this paper, we propose a "nested" domain decomposition method where the subsystems governing the interfaces are small enough so that they are easily solvable by direct methods on machines with many parallel processors. Convergence of the algorithms is also shown.

1. Introduction

Domain decomposition is a class of techniques designed to solve elliptic problems on irregular domains. Such methods are based on the idea that the domain of a certain problem may be regarded as the union of smaller subdomains of regular shape. Then the original problem may be solved by considering smaller problems in subdomains and the system governing the interfaces between subdomains.

In view of the interest in parallel computers, such methods are very attractive since the problems in subdomains can be solved independently. Even for sequential computation, a natural partition of the original domain often exists, such as in dividing a domain with irregular geometry into regular subregions for which fast solvers exists (e.g., see [4]), or in dividing a problem with discontinuous coefficients into subregions with constant coefficients.

Received September 10, 1994.

AMS subject classifications: 65M55, 65F10, 65Y05

Key words: nested domain decomposition, queuing network problems, parallel computing

Since the system governing the interface between subdomains is expensive to solve, many of the methods proposed so far use some form of the preconditioned conjugate gradient method, e.g., see [2, 3, 5, 8].

However, in this paper, instead of dealing with one large interface problem between subdomains, we propose a "nested" domain decomposition method where the subsystems governing the interfaces are small enough so that they are easily solvable by direct methods on machines with many parallel processors.

We start with a simple domain decomposition method applied to queuing network problems, and prove convergence of the resulting iterative algorithm. Then we extend the idea to construct a nested domain decomposition method for parallel processors.

2. Queuing network problems

A Markovian analysis of a queuing network based on solving the Kolmogorov equations for the steady-state probability distribution involves finding the null-vector of a large sparse structured matrix. Quantities of interest in the study of such networks include the probability of overflow from one queue to another, and the average waiting time of customers per queue etc. Most such quantities can be computed from the steady state probability distribution.

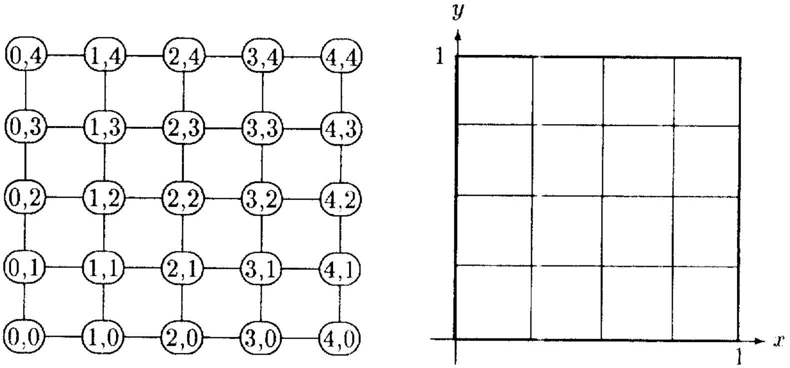
For some models, these probabilities can be obtained analytically. However, in most cases, such an analytic solution is not always available and the balance equation has to be solved explicitly.

Even for systems with a relatively small number of queues and a small number of waiting spaces and servers per queue, the resulting linear system $A\mathbf{x} = 0$ is huge, having dimension n, where n is the total number of states in the network. However the matrix is sparse, highly structured, and posesses enough algebraic structure that it is possible to solve some rather large systems.

The matrix is nonsymmetric, singular, irreducible, and has strictly positive diagonal and non-positive off-diagonal entries, with each column sum 0. It is known that such a matrix has a one dimensional null space, and the corresponding null-vector can be chosen to have positive entries([1]). Hence by adding a normalization condition, we can get a unique positive solution. For more details, see [7]. The model problem we consider is also from there.

3. Domain decomposition applied to queuing problems

In fact, there is no physical domain in queuing problems, and we cannot apply domain decomposition technique as is. However, since the graph of the matrix derived from a k-queue model is the same as that of a k-dimensional $(2k+1)$-point discrete Laplacian operator, we may regard the Kolmogorov balance equation as the finite discretization of some continuous problem on a rectangular domain. Therefore we could apply domain decomposition technique with a fictitious domain in mind.



Relation between states in 2-queue problems.

2-dimensional grid problems on the domain $[0,1] \times [0,1]$.

Figure 1: Relation between 2-queue problems and grid problems on the domain $[0,1] \times [0,1]$.

As an example, Figure 1 shows the relation between a 2-queue overflow model(with 4 waiting spaces in each queue) and a 2-dimensional grid problem. In the first figure, each ellipse represents a state of the queuing system, and the lines connecting them are possible state transitions. The ordered pair (i,j) in each ellipse represents the state where there are i customers in the first and j customers in the second queue, respectively. Note that its graph is exactly the same as that of the 2-dimensional grid problem on the domain $[0,1] \times [0,1]$ on the right.

Let the fictitious domain and the corresponding grid for a given k-queue model be as shown in Figure 2. For simplicity, we divide the k-dimensional domain into only two subregions Ω_1 and Ω_2, the boundary between the two subregions being Γ.

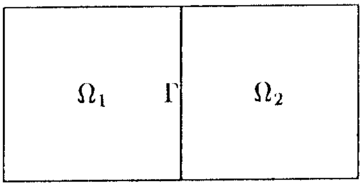


Figure 2: Domain decomposition into two subregions.

After reordering the unknowns, we can write the solution vector $\mathbf{x}$ as

$$\mathbf{x}^T = (\mathbf{x}_1^T, \mathbf{x}_2^T, \mathbf{x}_\Gamma^T)$$

where $\mathbf{x}_i$'s are the subvectors consisting of the variables in the subdomains Ω_i, $i = 1, 2$, respectively, and $\mathbf{x}_\Gamma$ is the subvector consisting of the variables on the boundary Γ. Then the system of equations $A\mathbf{x} = 0$ can be written in a block matrix notation as

$$(1) \qquad \begin{bmatrix} E_1 & 0 & F_1 \\ 0 & E_2 & F_2 \\ G_1 & G_2 & H \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_\Gamma \end{bmatrix} = 0,$$

or equivalently,

$$\begin{aligned} \mathbf{x}_i &= -E_i^{-1} F_i \mathbf{x}_\Gamma, \qquad i = 1, 2, \\ \mathbf{x}_\Gamma &= -H^{-1} \sum_{i=1}^{2} G_i \mathbf{x}_i. \end{aligned}$$

Combining the above two, we get an eigenvalue problem

$$(2) \qquad \mathbf{x}_\Gamma = H^{-1} \Big(\sum_{i=1}^{2} G_i E_i^{-1} F_i \Big) \mathbf{x}_\Gamma,$$

where $\mathbf{x}_\Gamma$ is the eigenvector corresponding to the eigenvalue of the matrix $H^{-1}(\sum_{i=1}^{2} G_i E_i^{-1} F_i)$ with the largest modulus.

By dividing into many smaller subdomains, we can make the sizes of the matrices E_i small enough to use direct methods like Gauss elimination. The computations $G_i E_i^{-1} F_i \mathbf{x}_\Gamma$, $i = 1, 2$, are all independent and can be performed in parallel.

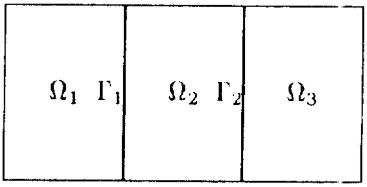


Figure 3: Domain decomposition into three subregions by two disjoint non-adjacent boundaries.

The discussion is still valid even if we divide the original domain into more than 2 pieces. For example, when the original domain is divided into 3 subdomains Ω_j, $j = 1, 2, 3$ by 2 boundaries Γ_1 and Γ_2(Figure 3), the system $A\mathbf{x} = 0$ becomes

$$(3) \qquad A\mathbf{x} = \begin{bmatrix} E_1 & 0 & 0 & F_{1,1} & 0 \\ 0 & E_2 & 0 & F_{2,1} & F_{2,2} \\ 0 & 0 & E_3 & 0 & F_{3,2} \\ G_{1,1} & G_{1,2} & 0 & H_1 & 0 \\ 0 & G_{2,2} & G_{2,3} & 0 & H_2 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_{\Gamma_1} \\ \mathbf{x}_{\Gamma_2} \end{bmatrix} = 0.$$

This is of the same form as (1), with some parts of F, G, H being zero. Note that $H = \mathrm{diag}(H_1, H_2)$ is block diagonal and separable if the two boundaries Γ_1 and Γ_2 are disjoint and non-adjacent to each other.

For notational simplicity and to write more general cases in the form of (1), define

$$F_i = [F_{i,1} \quad \cdots \quad F_{i,l-1}], \qquad G_i = \begin{bmatrix} G_{1,i} \\ \vdots \\ G_{l-1,i} \end{bmatrix},$$

$$\mathbf{x}_\Gamma^T = (\mathbf{x}_{\Gamma_1}^T, \ldots, \mathbf{x}_{\Gamma_{l-1}}^T). \qquad H = \mathrm{diag}(H_1, \ldots, H_{l-1}).$$

Then, based on domain decomposition of a given domain into l subregions by the boundaries Γ_i, $i = 1, \dots, l-1$, that are disjoint and nonadjacent to each other, we could use the following algorithm to solve our problem.

Algorithm 1

```
1. Take initial guesses $\mathbf{x}_{\Gamma_i} > 0, i = 1, \dots, l-1$.
2. Repeat until convergence
   1) Solve $E_i\mathbf{x}_i = -F_i\mathbf{x}_\Gamma$ for $\mathbf{x}_i, i = 1, \dots, l$ independently.
   2) $\mathbf{z}_i := \sum_{j=1}^{l} G_{ij}\mathbf{x}_j$, $i = 1, \dots, l-1$ independently.
   3) Solve $H_i\mathbf{x}_{\Gamma_i} = -\mathbf{z}_i$ for $\mathbf{x}_{\Gamma_i}$, $i = 1, \dots, l-1$ independently.
```

Note that each step can be computed by parallel processors independently, and even some inner computations(for example, $F_i\mathbf{x}_\Gamma$ in step 2.1), etc.) can be also done in parallel.

4. Convergence of the algorithm

We consider the convergence of the "outer" iteration when it is applied to queuing problems, assuming that the system governing the interfaces between subdomains and the smaller problems in each subdomain are solved exactly by some method(the steps 2.1) and 2.3)).

For brevity, we write the resulting matrix equation from domain decomposition as

$$\begin{bmatrix} E & F \\ G & H \end{bmatrix} \begin{bmatrix} \mathbf{x}_I \\ \mathbf{x}_\Gamma \end{bmatrix} = 0,$$

where $\mathbf{x}_I$ is the subvector consisting of all the variables in any interior subdomains, $\mathbf{x}_\Gamma$ is the subvector consisting of all the variables on the interfaces, and the matrix A is partitioned accordingly. Note that E is a diagonally dominant block diagonal M-matrix. After block elimination, we get

$$\begin{bmatrix} E & F \\ 0 & K \end{bmatrix} \begin{bmatrix} \mathbf{x}_I \\ \mathbf{x}_\Gamma \end{bmatrix} = 0,$$

where $K = H - GE^{-1}F$ is the Schur complement(e.g., see [6]). Hence the previous iterative method can be written as(see (2))

$$\mathbf{x}_\Gamma := H^{-1}GE^{-1}F\mathbf{x}_\Gamma.$$

This can be viewed as the power method using the iteration matrix $H^{-1}GE^{-1}F$, the solution being the eigenvector corresponding to the eigenvalue 1 of the iteration matrix. Hence the iterative method converges if and only if no other eigenvalue has modulus greater than or equal to 1.

LEMMA 1. *Let* $A = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$, $A \in \mathbf{C}^{n\times n}$, $E \in \mathbf{C}^{(n-p)\times(n-p)}$, *and* $H \in \mathbf{C}^{p\times p}$. *Then the resulting* $p \times p$ *matrix at the southeast corner after applying* $n - p$ *steps of Gauss elimination is the same as the Schur complement* $H - GE^{-1}F$.

PROOF. [6].

LEMMA 2. *Let the* $n \times n$ *matrix* A *have column sums 0, with positive diagonal entries and nonpositive off-diagonal entries. Then, after applying* $n - p$ *steps of Gauss elimination, the resulting* $p \times p$ *matrix at the south-east corner has column sums 0 too.*

PROOF. Let $a_{i,j}^{(k-1)}$ denote the (i,j) entry of the matrix A after application of $k-1$ steps of Gauss elimination. Assume the assertion is true, i.e., column sums of the $(n-k+1)\times(n-k+1)$ submatrix in the southeast corner are all zero. Hence

$$\sum_{i=k}^{n} a_{i,j}^{(k-1)} = 0, \quad j = k, \dots, n.$$

To annihilate the entries in the kth column $a_{i,k}^{(k-1)}$, $i = k+1, \dots, n$, multiply the kth row by $a_{i,k}^{(k-1)}/a_{k,k}^{(k)}$ and subtract from the ith row. Therefore after k-steps of Gauss elimination, the (i,j) entry of the resulting matrix will be

$$a_{i,j}^{(k)} = \begin{cases} a_{i,j}^{(k-1)} & i = 1, \dots, k, \quad j = 1, \dots, n, \\ a_{i,j}^{(k-1)} - a_{i,k}^{(k-1)} a_{k,j}^{(k-1)} / a_{k,k}^{(k-1)} & j = k, k+1, \dots, n. \end{cases}$$

Then the jth column sum, $j = k+1, \dots, n$ is

$$\sum_{i=k+1}^{n} a_{i,j}^{(k)} = \sum_{i=k+1}^{n} a_{i,j}^{(k-1)} - \sum_{i=k+1}^{n} a_{i,k}^{(k-1)} a_{k,j}^{(k-1)} / a_{k,k}^{(k-1)}$$

$$
\begin{aligned}
&= \sum_{i=k+1}^{n} a_{i,j}^{(k-1)} - a_{k,j}^{(k-1)}/a_{k,k}^{(k-1)} \sum_{i=k+1}^{n} a_{i,k}^{(k-1)} \\
&= \sum_{i=k+1}^{n} a_{i,j}^{(k-1)} - a_{k,j}^{(k-1)}/a_{k,k}^{(k-1)}(-a_{k,k}^{(k-1)}) \\
&= \sum_{i=k}^{n} a_{i,j}^{(k-1)} = 0.
\end{aligned}
$$

Lemma 3. *Let A be a matrix whose column sums are all 0, with positive diagonal entries and nonpositive off-diagonal entries. Then, for any splitting $A = M - N$, $\rho(M^{-1}N) = 1$.*

Proof. [7].

Definition 1. (Ortega [9]) A matrix $A \in \mathbf{R}^{n \times n}$ is an M-matrix if $A^{-1} \geq 0$ and $a_{ij} \leq 0$, $i \neq j$.

Lemma 4. *Let $A \in \mathbf{R}^{n\times n}$ be strictly or irreducibly diagonally dominant and assume that $a_{ij} \leq 0$, $i \neq j$, and $a_{ii} > 0$, $i = 1, 2, \ldots, n$. Then A is an M-matrix.*

Proof. [9].

Lemma 5. *The inverses of the matrices E and H are nonnegative.*

Proof. Since the matrices E and H are strictly diagonally dominant, and have positive diagonal entries and nonpositive off-diagonal entries, they are M-matrices by Lemma 4. Therefore by Definition 1, their inverses are nonnegative.

Lemma 6. $\rho(H^{-1}GE^{-1}F) = 1$.

Proof. By Lemma 1, the Schur complement is the same as that obtained by application of some steps of Gauss elimination, and hence the iteration matrix K has zero column sums. Note that E and H are M-matrices, F and G are nonpositive, and E^{-1} is nonnegative by Lemma 5. Hence all the off-diagonal entries of the Schur complement

$K = H - GE^{-1}F$ must be nonpositive. Since each column sum of K is 0 by Lemma 2, diagonal entries of K must be strictly positive. Therefore $\rho(H^{-1}GE^{-1}F) = 1$ by Lemma 3.

LEMMA 7. *The iteration matrix $T = H^{-1}GE^{-1}F$ is nonnegative.*

PROOF. By Lemma 5, H^{-1} and E^{-1} are nonnegative. In addition, F and G are nonpositive matrices. Hence the iteration matrix T must be nonnegative.

THEOREM 1. *The previous algorithm converges for any choice of initial guess $\mathbf{x}_\Gamma$, the asymptotic rate of convergence being $|\lambda_2|$, where λ_2 is the the second largest eigenvalue in modulus of the iteration matrix T.*

PROOF. By Lemma 6, $\rho(T) = 1$. Since T is nonnegative by Lemma 7, it has only 1 eigenvalue on the unit circle by the Perron-Frobenius theory(e.g., see [10] or [11]), and the iterative method converges to the correct solution. Since the above method is also a power method, the convergence rate is clear.

5. Nested domain decomposition

Algorithm 1 considers "1-level" domain decomposition, i.e., a method to divide the original domain by one or more boundaries in one step. However, we can apply the idea recursively to reduce not only the size of the interior domain problems in each subdomain but also the size of the problems governing the interfaces between them, by using the nested domain decomposition technique described below.

For example, assume that we would like to divide the original domain into 6 subdomains. Instead of dividing the domain into 6 pieces in one step, we first divide it into 3 pieces by the boundaries Γ_1 and Γ_2(Say, "the first level".). Then we further split the subdomains by the boundaries $\Gamma_{1,1}$, $\Gamma_{2,1}$, and $\Gamma_{3,1}$(Say, "the second level". $\Gamma_{i,j}$ means that it is the jth boundary in the ith region in the first level.), as shown in Figure 4. In this manner, we can recursively divide the original domain into many small subdomains, keeping each interface problem small.

Let the resulting 6 subdomains be Ω_{ij}, $i = 1, 2, 3$, $j = 1, 2$, and $\mathbf{x}_{i,j}$ be the subvector that consists of the unknowns at the grid points in Ω_{ij}.

In addition, let $\mathbf{x}_{\Gamma_i}$ be the subvector consisting of the unknowns on the boundary Γ_i, and $\mathbf{x}_{\Gamma_{i,j}}$ be the subvector for the boundary $\Gamma_{i,j}$. Using similar notations as before, the resulting matrix A in the system $A\mathbf{x} = 0$

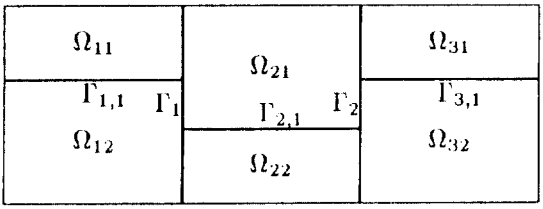


Figure 4: Nested domain decomposition.

is given by

(4)

$$\begin{bmatrix}
E_1^{(1)} & 0 & F_{1,1}^{(1)} & & & & & & & P_{1,1}^{(1)} & 0 \\
0 & E_2^{(1)} & F_{2,1}^{(1)} & & 0 & & & 0 & & P_{2,1}^{(1)} & 0 \\
G_{1,1}^{(1)} & G_{1,2}^{(1)} & H_1^{(1)} & & & & & & & R_{1,1}^{(1)} & 0 \\
 & & & E_1^{(2)} & 0 & F_{1,1}^{(2)} & & & & P_{1,1}^{(2)} & P_{1,2}^{(2)} \\
 & 0 & & 0 & E_2^{(2)} & F_{2,1}^{(2)} & & 0 & & P_{2,1}^{(2)} & P_{2,2}^{(2)} \\
 & & & G_{1,1}^{(2)} & G_{1,2}^{(2)} & H_1^{(2)} & & & & R_{1,1}^{(2)} & R_{1,2}^{(2)} \\
 & & & & & & E_1^{(3)} & 0 & F_{1,1}^{(3)} & 0 & P_{1,2}^{(3)} \\
 & 0 & & & 0 & & 0 & E_2^{(3)} & F_{2,1}^{(3)} & 0 & P_{2,2}^{(3)} \\
 & & & & & & G_{1,1}^{(3)} & G_{1,2}^{(3)} & H_1^{(3)} & 0 & R_{1,2}^{(3)} \\
Q_{1,1}^{(1)} & Q_{1,2}^{(1)} & S_{1,1}^{(1)} & Q_{1,1}^{(2)} & Q_{1,2}^{(2)} & S_{1,1}^{(2)} & 0 & 0 & 0 & H_{\Gamma_1} & 0 \\
0 & 0 & 0 & Q_{2,1}^{(2)} & Q_{2,2}^{(2)} & S_{2,1}^{(2)} & Q_{2,1}^{(3)} & Q_{2,2}^{(3)} & S_{2,1}^{(3)} & 0 & H_{\Gamma_2}
\end{bmatrix}$$

where the superscript (k) means that the matrix is related to the kth subdomain in the first level. Correspondingly, the solution vector $\mathbf{x}$ can be partitioned as

(5)

$$\mathbf{x}^T = \left[\mathbf{x}_{1,1}^T,\ \mathbf{x}_{1,2}^T,\ \mathbf{x}_{\Gamma_{1,1}}^T,\ \mathbf{x}_{2,1}^T,\ \mathbf{x}_{2,2}^T,\ \mathbf{x}_{\Gamma_{2,1}}^T,\ \mathbf{x}_{3,1}^T,\ \mathbf{x}_{3,2}^T,\ \mathbf{x}_{\Gamma_{3,1}}^T,\ \mathbf{x}_{\Gamma_1}^T,\ \mathbf{x}_{\Gamma_2}^T\right].$$

More generally, consider the case when the original domain is divided into l subregions first, and then each of them is divided again into m pieces(Let's call it (l, m)-decomposition.). We assume that the domain is decomposed by the boundaries that are disjoint from and non-adjacent to each other. For simplicity, introduce the notations

$$E_i = \begin{bmatrix} E_1^{(i)} & & & 0 & F_{1,1}^{(i)} & \cdots & F_{1,m-1}^{(i)} \\ & E_2^{(i)} & & & F_{2,1}^{(i)} & \cdots & F_{2,m-1}^{(i)} \\ & & \ddots & & \vdots & & \vdots \\ 0 & & & E_m^{(i)} & F_{m,1}^{(i)} & \cdots & F_{m,m-1}^{(i)} \\ G_{1,1}^{(i)} & G_{1,2}^{(i)} & \cdots & G_{1,m}^{(i)} & H_1^{(i)} & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ G_{m-1,1}^{(i)} & G_{m-1,2}^{(i)} & \cdots & G_{m-1,m}^{(i)} & 0 & \cdots & H_{m-1}^{(i)} \end{bmatrix}$$

and

$$F_{i,j} = \begin{bmatrix} P_{1,j}^{(i)} \\ \vdots \\ P_{m,j}^{(i)} \\ R_{1,j}^{(i)} \\ \vdots \\ R_{m-1,j}^{(i)} \end{bmatrix}, \qquad \mathbf{x}_i = \begin{bmatrix} \mathbf{x}_{i,1} \\ \vdots \\ \mathbf{x}_{i,m} \\ \mathbf{x}_{\Gamma_{i,1}} \\ \vdots \\ \mathbf{x}_{\Gamma_{i,m-1}} \end{bmatrix},$$

$$G_{i,j} = \left[Q_{i,1}^{(j)}, \cdots, Q_{i,m}^{(j)}, S_{i,1}^{(j)}, \cdots, S_{i,m-1}^{(j)}\right],$$

for $i = 1, \ldots, l$, $j = 1, \ldots, m$. Equations (4) and (5) are for the system when $l = 3$ and $m = 2$.

Hence the system for the 2-level (l, m)-decomposition can be written as

$$A\mathbf{x} = \begin{bmatrix} E_1 & & & 0 & F_{1,1} & \cdots & F_{1,l-1} \\ & E_2 & & & F_{2,1} & \cdots & F_{2,l-1} \\ & & \ddots & & \vdots & & \vdots \\ 0 & & & E_l & F_{l,1} & \cdots & F_{l,l-1} \\ G_{1,1} & G_{1,2} & \cdots & G_{1,l} & H_1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ G_{l-1,1} & G_{l-1,2} & \cdots & G_{l-1,l} & 0 & \cdots & H_{l-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_l \\ \mathbf{x}_{\Gamma_1} \\ \vdots \\ \mathbf{x}_{\Gamma_{l-1}} \end{bmatrix} = 0,$$

which is exactly in the same form as (3). For k-queue models, all the H_i's and $H_i^{(j)}$'s are $(2k-1)$-diagonal matrices, and all the $G_{i,j}$ and $F_{i,j}$ are sparse and some of them are zero matrices depending on the values of i and j.

Now we show how the two-level nested domain decomposition can be computed, and prove the convergence. Since general algorithms for arbitrary decomposition can be very complicated, we rather show an algorithm to solve the system for 2-level (3,2)-decomposition as an example so that readers can understand basic ideas easily.

Using (4) and (5), the system $A\mathbf{x} = 0$ for 2-level (3, 2)-decomposition can be written as

(6)

$$\begin{bmatrix} E_1^{(i)} & 0 & F_{1,1}^{(i)} \\ 0 & E_2^{(i)} & F_{2,1}^{(i)} \\ G_{1,1}^{(i)} & G_{1,2}^{(i)} & H_1^{(i)} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{i,1} \\ \mathbf{x}_{i,2} \\ \mathbf{x}_{\Gamma_{i,1}} \end{bmatrix} = - \begin{bmatrix} P_{1,1}^{(i)} & P_{1,2}^{(i)} \\ P_{2,1}^{(i)} & P_{2,2}^{(i)} \\ R_{1,1}^{(i)} & R_{1,2}^{(i)} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\Gamma_1} \\ \mathbf{x}_{\Gamma_2} \end{bmatrix}, \quad i = 1, 2, 3,$$

$$H_i \mathbf{x}_{\Gamma_i} = - \sum_{j=1}^{3} \Big(\sum_{k=1}^{2} Q_{i,k}^{(j)} \mathbf{x}_{j,k} + S_{i,1}^{(j)} \mathbf{x}_{\Gamma_{j,1}} \Big), \quad i = 1, 2.$$

The inner loop (6) can be written as

$$E_j^{(i)} \mathbf{x}_{i,j} = - \Big(\sum_{k=1}^{2} P_{j,k}^{(i)} \mathbf{x}_{\Gamma_k} \Big) - F_{j,1}^{(i)} \mathbf{x}_{\Gamma_{i,1}}, \quad i = 1, 2, 3, \; j = 1, 2,$$

$$H_1^{(i)} \mathbf{x}_{\Gamma_{i,1}} = - \sum_{j=1}^{2} \Big(G_{1,j}^{(i)} \mathbf{x}_{i,j} + R_{1,j}^{(i)} \mathbf{x}_{\Gamma_j} \Big), \quad i = 1, 2.$$

Hence we can use the following algorithm to solve the system $A\mathbf{x} = 0$ for 2-level (3,2)-decomposition.

Algorithm 2 [2-level (3,2)-decomposition algorithm]

```
1. Take initial guesses $\mathbf{x}_{\Gamma_i} > 0$ and $\mathbf{x}_{\Gamma_{i,1}} > 0$.
2. Repeat until convergence of $\mathbf{x}_{\Gamma_i}$'s,
    1) Repeat until convergence of $\mathbf{x}_{\Gamma_{i,1}}$'s,
        1. Solve $E_j^{(i)} \mathbf{x}_{i,j} = -(\sum_{k=1}^{2} P_{j,k}^{(i)} \mathbf{x}_{\Gamma_k}) - F_{j,1}^{(i)} \mathbf{x}_{\Gamma_{i,1}}$ for $\mathbf{x}_{i,j}$, $i =$
           $1, 2, 3$, $j = 1, 2$ indep.
        2. $\mathbf{v} := \sum_{j=1}^{2}(G_{1,j}^{(i)}\mathbf{x}_{i,j} + R_{1,j}^{(i)}\mathbf{x}_{\Gamma_j})$, $i = 1,2,3$ indep.
        3. Solve $H_1^{(i)}\mathbf{x}_{\Gamma_{i,1}} = -\mathbf{v}$ for $\mathbf{x}_{\Gamma_{i,1}}$ $i = 1,2,3$ indep.
    2) $\mathbf{z}_i := \sum_{j=1}^{3}(\sum_{k=1}^{2} Q_{i,k}^{(j)}\mathbf{x}_{j,k} + S_{i,1}^{(j)}\mathbf{x}_{\Gamma_{j,1}})$, $i = 1,2$ indep.
    3) Solve $H_i\mathbf{x}_{\Gamma_i} = -\mathbf{z}_i$ for $\mathbf{x}_{\Gamma_i}$, $i = 1,2$ indep.
```

Now we prove the convergence of such an algorithm for general (l, m)-decomposition. Convergence proof of the outer iteration is already given in the previous section. Hence we only have to consider the convergence of the inner loop like the steps 2.1.1 - 2.1.3. Since the matrix for the inner loop is small compared to the original problem, we also assume that the steps like 2.1.1 and 2.1.3 are solved exactly by some direct method.

For brevity, we write the inner loop of (l, m)-decomposition (like (6)) as

$$\begin{bmatrix} E & F \\ G & H \end{bmatrix} \begin{bmatrix} \mathbf{x}_I \\ \mathbf{x}_\Gamma \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix},$$

where $\mathbf{x}_I^T = [\mathbf{x}_{i,1}^T, \ \ldots \ , \mathbf{x}_{i,m}^T]$, $\mathbf{x}_\Gamma^T = [\mathbf{x}_{\Gamma_{i,1}}^T, \ \ldots \ , \mathbf{x}_{\Gamma_{i,m-1}}^T]$, and the matrix and the righthand side are partitioned accordingly. After block elimination, we get

$$\begin{bmatrix} E & F \\ 0 & K \end{bmatrix} \begin{bmatrix} \mathbf{x}_I \\ \mathbf{x}_\Gamma \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2' \end{bmatrix},$$

where $K = H - GE^{-1}F$ is the Schur complement as before, and $\mathbf{r}_2' = \mathbf{r}_2 - GE^{-1}\mathbf{r}_1$. Hence the previous iterative method, which is equivalent to $K\mathbf{x}_\Gamma = \mathbf{r}_2'$, can be written as

$$\mathbf{x}_\Gamma := H^{-1}GE^{-1}F\mathbf{x}_\Gamma + H^{-1}\mathbf{r}_2'.$$

The following is a lemma about the necessary and sufficient condition for such an iterative method to converge to the desired solution. The superscript (m) means that its the mth iterate(in this lemma only).

LEMMA 8. *Let $I-T$ be a nonsingular matrix. Then a general iterative method*

$$\mathbf{x}^{(m+1)} = T\mathbf{x}^{(m)} + \mathbf{b}, \quad m \geq 0$$

converges to the unique solution of $(I-T)\mathbf{x} = \mathbf{b}$ if and only if $\rho(T) < 1$.

PROOF. [11]

Hence the iterate of the inner loop is guaranteed to converge to the solution of $K\mathbf{x}_\Gamma = \mathbf{r}_2'$ if and only if $\rho(H^{-1}GE^{-1}F) < 1$. The proof is similar to that in the previous section. Note that each matrix for the inner loops is a diagonal block of the original matrix, and its column sums are all strictly positive.

LEMMA 9. *Let the $n \times n$ matrix A have positive column sums, with positive diagonal entries and nonpositive off-diagonal entries. Then, after applying $n-p$ steps of Gauss elimination, the resulting $p \times p$ matrix at the south-east corner has positive column sums too.*

PROOF. The proof is exactly the same as that of Lemma 2, except that $\sum_{i=k}^{n} a_{i,j}^{(k-1)} > 0$ at the end.

LEMMA 10. *Let A be an $n \times n$ matrix whose column sums are all positive, with positive diagonal entries and nonpositive off-diagonal entries. Then, for any splitting $A = M - N$ where M is an M-matrix, $\rho(M^{-1}N) < 1$.*

PROOF. Since $\sum_{i=1}^{n} a_{ij} > 0$ for $j = 1, 2, \ldots, n$,

$$\mathbf{e}^T A = \mathbf{e}^T(M - N) > 0 \quad \text{columnwise},$$

where $\mathbf{e}$ is the vector of all 1's. This implies

$$(M - N)^T \mathbf{e} > 0 \quad \text{rowwise}.$$

Since M is an M-matrix, $M^{-1} \geq 0$ and so is its transpose. Therefore premultiplying by M^{-T} and simplifying,

$$\mathbf{e} > M^{-T}N^T\mathbf{e} \quad \text{rowwise}.$$

This means that the maximum row sum of $M^{-T}N^T$ is less than 1 or $||M^{-T}N^T||_\infty < 1$. But

$$1 > ||M^{-T}N^T||_\infty \geq \rho(M^{-T}N^T) = \rho(NM^{-1}) = \rho(M^{-1}N).$$

LEMMA 11. *The inverses of the matrices E and H are nonnegative.*

PROOF. Same as that of Lemma 5.

LEMMA 12. $\rho(H^{-1}GE^{-1}F) < 1$.

PROOF. By Lemmas 1 and 9, the matrix $K = H - GE^{-1}F$ has positive column sums. Note that E and H are M-matrices, F and G are nonpositive, and E^{-1} is nonnegative by Lemma 11. Hence all the off-diagonal entries of the Schur complement K must be nonpositive. Since each column sum of K is positive by Lemma 9, diagonal entries of K must be strictly positive. Therefore $\rho(H^{-1}GE^{-1}F) < 1$ by Lemma 10.

Hence we can state

THEOREM 2. *The inner loop of a general 2-level algorithm like Algorithm 2 converges for any choice of initial guesses $\mathbf{x}_{\Gamma_i}$ and $\mathbf{x}_{\Gamma_{i,j}}$, the asymptotic rate of convergence being $\rho(H^{-1}GE^{-1}F)$.*

We can easily extend the idea to arbitrary number of levels. The convergence proof is still valid for arbitrary number level domain decomposition. For the speed of convergence, the inner loop is dependent on the spectral radius of the inner iteration matrix, However the outer loop is dependent on the modulus of the second largest eigenvalue of the outer iteration matrix, the estimation of its size is still open.

One of the advantages of the domain decomposition technique applied to queuing problems is that, unlike normal grid problems where domain decomposition is subject to the properties of the problem, we can divide the fictitious domain as we like without any restrictions since the queuing problem does not have a physical domain.

Moreover, the domain decomposition technique can be applied to any queuing problems in purely algebraic sense, even if there is no corresponding continuous equation whose finite discretization is similar to the queuing problem.

References

1. A. Berman and R. J. Plemmons, *Nonnegative matrices in the mathematical sciences*, Academic Press, New York, 1979.

2. P. E. Bjϕrstad and O. B. Widlund, *Iterative methods for the solution of elliptic problems on regions partitioned into substructures*, SIAM J. Numer. Anal. **23** (1986), 1097-1120.
3. T. F. Chan, *Analysis of preconditioners for domain decomposition*, SIAM J. Num. Anal. **24** (1987), 382-390.
4. T. F. Chan and D. C. Resasco, *A domain-decomposed fast Poisson solver on a rectangle*, SIAM J. Sci. Stat. Comput. **8** (1987), s14-s26.
5. M. Dryja and W. Proskurowski, *Domain decomposition seminar notes*, Technical Report CRI-86-12, Univ. of Southern California, 1986.
6. G. H. Golub and C. F. van Loan, *Matrix computations*, Johns Hopkins Univ. Press, Baltimore, Maryland, 1983.
7. L. Kaufman, *Matrix methods for queuing problems*, SIAM J. Sci. Comput. **4** (1983), 525-552.
8. D. E. Keyes and W. D. Gropp, *A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation*, SIAM J. Sci. Stat. Comput. **8** (1987), s166-s202.
9. J. M. Ortega, *Numerical analysis: A second course*, Academic Press, New York, 1972.
10. E. Seneta, *Non-negative matrices and Markov chains, 2nd ed.*, Springer-Verlag, New York, 1981.
11. R. S. Varga, *Matrix iterative analysis*, Prentice-Hall, New Jersey, 1962.

Computer Science Department
University of Suwon
Kyungki-Do 445-743, Korea