A BLOCK-PARALLEL CONJUGATE GRADIENT METHOD FOR SEPARABLE QUADRATIC PROGRAMMING PROBLEMS

Eiki Yamakawa
ATR Human Information Processing Research Laboratories

Masao Fukushima
Nara Institute of Science and Technology

(November 14, 1994)

Abstract For a large-scale quadratic programming problem with separable objective function, a variant of the conjugate gradient method can effectively be applied to the dual problem. In this paper, we consider a block-parallel modification of the conjugate gradient method, which is suitable for implementation on a parallel computer. More precisely, the method proceeds in a block Jacobi manner and executes the conjugate gradient iteration to solve quadratic programming subproblems associated with respective blocks. We implement the method on a Connection Machine Model CM-5 in the Single-Program Multiple-Data model of computation. We report some numerical results, which show that the proposed method is effective particularly for problems with some block structure.

1. Introduction

In recent years, much effort has been made in the design of parallel algorithms for solving various optimization problems [2–4, 6–14, 22–24, 28, 29]. Many of those algorithms are of the data parallel type, which execute identical or similar operations on numerous data concurrently using many processors. The two concepts that play an important role in designing such parallel algorithms are the duality in mathematical programming and the splitting of operators or matrices involved. For example, Fukushima et al. [14] have developed a descent type algorithm for solving general convex programming problems, using a dual optimization approach incorporating the idea of operator splitting. Another useful idea from the duality theory is the alternating direction method of multipliers [7–9, 12], which is considered an ingenious dual method combined with operator splitting. On the other hand, Mangasarian and De Leone [22, 23] propose parallel successive over-relaxation (SOR) methods for solving the symmetric linear complementarity problem. Those methods exploit a matrix splitting that uses block diagonal parts of the matrix involved. More recently, multisplitting has turned out to be a potential idea in developing effective parallel algorithms [11, 20]. Practical large-scale problems often possess some special structure, which can be exploited in designing a variety of parallel algorithms. Zenios [28] discusses basic issues in the design and implementation of data level parallelism for problems with network structure. For problems with certain block structure, parallelizable decomposition algorithms are proposed in [3, 10], which may effectively be realized in the Single-Program Multiple-Data (SPMD) model of computation.

In this paper, we propose an algorithm, particularly suited for parallel computation, to solve the following separable quadratic programming problem:

$$\begin{array}{llll} \mathrm{P}: & \text{minimize} & \frac{1}{2}x^\top Dx + c^\top x & \\ & \text{subject to} & a_i^\top x = b_i, & i \in E, \\ & & a_i^\top x \le b_i, & i \in I, \end{array}$$

where E and I are finite index sets, D is an $n \times n$ positive diagonal matrix, c and a_i are vectors in R^n, b_i are real numbers and $\top$ denotes transposition. Let A and b denote the matrix whose i-th row is $a_i^\top$, $i \in E \cup I$, and the vector whose i-th element is b_i, $i \in E \cup I$, respectively. Note that, if problem P is feasible, it has a unique solution because of the strong convexity of the objective function. The dual of problem P can be defined as

$$\begin{array}{llll} \mathrm{D}: & \text{minimize} & \frac{1}{2}z^\top Mz + q^\top z & \\ & \text{subject to} & z_i \ge 0, & i \in I, \end{array}$$

where

$$\begin{aligned} M &= AD^{-1}A^\top, \\ q &= AD^{-1}c + b. \end{aligned} \tag{1.1}$$

From the standard duality theory [21], we can easily verify that the optimal solution x^* of problem P is related to an optimal solution z^* of problem D by

$$x^* = -D^{-1}(A^\top z^* + c). \tag{1.2}$$

Let r denote the negative gradient of the dual objective function at z, that is,

$$r = -Mz - q.$$

Then the Karush-Kuhn-Tucker conditions for problem D may be written as

$$\begin{array}{ll} r_i = 0, & i \in E, \\ r_i \le 0, \quad z_i \ge 0, \quad r_i z_i = 0, & i \in I. \end{array} \tag{1.3}$$

Since M is positive semidefinite, conditions (1.3) are not only necessary but also sufficient for optimality of problem D. Finding a z satisfying (1.3) is in general a mixed linear complementarity problem. In particular, when the index set E is empty, (1.3) becomes the symmetric linear complementarity problem

$$Mz + q \ge 0, \quad z \ge 0, \quad z^\top(Mz + q) = 0,$$

while, in the case where the index set I is empty, (1.3) reduces to the system of linear equations

$$Mz + q = 0.$$

The algorithm proposed in this paper is a block-parallel modification of the conjugate gradient method applied to the mixed linear complementarity problem (1.3). More precisely, the algorithm exploits a splitting of matrix M such that $M = G + H$ with G being a block diagonal matrix. Thus the algorithm may be regarded as a block Jacobi method. The resulting subproblems involving matrix G are decomposed into small mixed linear complementarity problems, which can be solved independently from each other. Those mixed linear complementarity problems are solved using a conjugate gradient method, which is a modification of the one given in [16] and is extended to deal with simple bounds. This approach is particularly effective when the problem is sparse, because it only requires simple operations on the vectors a_i. In the special case where the block diagonal matrix G is actually a diagonal matrix, the proposed algorithm reduces to the classical point Jacobi method. Although the point Jacobi method is suited for massively parallel computations, but often suffers slow convergence because of small steps. The block Jacobi modification usually leads to larger steps and hence accelerated convergence is expected. To examine the practical efficiency of the proposed algorithm, it is implemented on a Connection Machine Model CM-5 in the SPMD model of computation. Particular attention will be paid on the efficiency of the algorithm on problems with block angular structure.

This paper is organized as follows. A splitting method for problem D is presented and its convergence properties are discussed in Section 2. A block-parallel algorithm for problem P is then developed in Section 3. Implementation strategies for the proposed algorithm in the SPMD model of computation are described in Section 4. Some computational results on the Connection Machine Model CM-5 are reported in Section 5. Conclusions are given in Section 6.

Throughout the paper we shall adopt the following notation. Let J be an index set. For a vector u, u_J denotes the subvector of u with components u_i, $i \in J$. For matrix A, A_J denotes the submatrix of A consisting of the rows $a_i^\top$, $i \in J$.

2. Basic Splitting Method

In this section, we describe a splitting method for solving problem D, which is a natural modification of that for the symmetric linear complementarity problem [5]. A pair (G, H) of matrices is called a *splitting* of matrix M if

$$M = G + H. \tag{2.1}$$

A splitting (G, H) is said to be *regular* if $G - H$ is positive definite [5, p. 400]. Since M is positive semi-definite (see (1.1)), $G = (M + G - H) / 2$ is positive definite for any regular splitting (G, H).

Using a regular splitting (G, H) of M, a splitting method for the mixed linear complementarity problem (1.3) generates a sequence $\{z^{(k)}\}$ by the following iterative process.

Splitting Method (Prototype) :

Given $z^{(k)}$, find a solution $z^{(k+1)}$ of the mixed linear complementarity problem

$$\begin{aligned} &\left(Gz + v^{(k)}\right)_i = 0, && i \in E, \\ &\left(Gz + v^{(k)}\right)_i \geq 0, \quad z_i \geq 0, \quad \left(Gz + v^{(k)}\right)_i z_i = 0, && i \in I, \end{aligned} \tag{2.2}$$

where

$$v^{(k)} = Hz^{(k)} + q. \tag{2.3}$$ □

If G is symmetric, then subproblem (2.2) is equivalent to the following quadratic programming problem:

$$\begin{array}{llll} \mathrm{DSP}^{(k)}: & \text{minimize} & \frac{1}{2} z^\top G z + \left(v^{(k)}\right)^\top z & \\ & \text{subject to} & z_i \geq 0, & i \in I. \end{array}$$

Since G is positive definite as noted above, subproblem $\mathrm{DSP}^{(k)}$ has a unique solution for each k.

General convergence results of the splitting method for symmetric linear complementarity problems have recently been obtained by Luo and Tseng [18] and Iusem [17]. Moreover, Luo and Tseng [19] have established the convergence of the splitting method for symmetric affine variational inequality problems, which contain as special cases various optimization problems. In particular, the mixed linear complementarity problem (1.3) can be represented as the affine variational inequality problem of finding a $z^* \in Z$ such that

$$(z - z^*)^\top (Mz^* + q) \geq 0, \qquad \forall z \in Z,$$

where Z is the polyhedral convex set in $R^{|E|+|I|}$ defined by

$$Z = \{ z \mid z_i \in R, i \in E, \quad \text{and} \quad z_i \geq 0, i \in I \}.$$

Moreover, the solution $z^{(k+1)}$ of the mixed linear complementarity problem (2.2) is the vector in Z that satisfies the variational inequality

$$\left(z - z^{(k+1)}\right)^\top \left(Gz^{(k+1)} + v^{(k)}\right) \geq 0, \qquad \forall z \in Z.$$

Therefore, the convergence of the splitting method for solving the mixed linear complementarity problem (1.3) can be deduced from the general results established in [19].

Theorem 1. If problem P is feasible, then the sequence $\{z^{(k)}\}$ generated by the splitting method converges to an optimal solution of problem D.

Proof : Under our blanket assumptions, if problem P is feasible, it has a unique solution. Then, from the standard duality theory [21], problem D also has a solution and hence the objective function of problem D is bounded from below on Z. Consequently the assumptions of Theorem 3.2 in [19] are satisfied and the desired results follow. □

3. Block-Parallel Algorithm

By appropriately choosing matrix G involved in subproblem $\mathrm{DSP}^{(k)}$, a block-parallel algorithm for problem P is derived from the splitting method described in the previous section. To see this, let $J_\ell, \ell = 1, \cdots, L$, be index sets such that

$$\bigcup_{\ell=1}^{L} J_\ell = E \cup I \qquad \text{and} \qquad J_\ell \cap J_{\ell'} = \emptyset, \quad \ell \neq \ell'.$$

Suppose that we have a regular splitting (G, H) of matrix M such that, by reordering the rows and the columns if necessary, G is a block diagonal matrix of the form

$$G = \begin{pmatrix} G_1 & 0 & \cdots & 0 \\ 0 & G_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & G_L \end{pmatrix},$$

where G_ℓ are $|J_\ell| \times |J_\ell|$ symmetric matrices whose rows and columns correspond to the indices in J_ℓ. Then, subproblem $\mathrm{DSP}^{(k)}$ can be decomposed into L independent subproblems $\mathrm{DSP}_\ell^{(k)}$, $\ell = 1, \cdots, L$, as follows:

$$\begin{array}{llll} \mathrm{DSP}_\ell^{(k)}: & \text{minimize} & \frac{1}{2} z_{J_\ell}^\top G_\ell \, z_{J_\ell} + \left(v_{J_\ell}^{(k)} \right)^\top z_{J_\ell} & \\ & \text{subject to} & z_i \geq 0, & i \in I \cap J_\ell, \end{array}$$

where $v_{J_\ell}^{(k)}$ is the subvector of $v^{(k)}$ given by (2.3). Since G is positive definite, matrices G_ℓ are also positive definite. Thus, each subproblem $\mathrm{DSP}_\ell^{(k)}$ always has a unique solution.

Note that the vector $v_{J_\ell}^{(k)}$ involved in each subproblem $\mathrm{DSP}_\ell^{(k)}$ is calculated in the following way. For the current dual estimate $z^{(k)}$, let

$$x^{(k)} := -D^{-1} \left(A^\top z^{(k)} + c \right). \tag{3.1}$$

Then, $x^{(k)}$ can be regarded as an estimate of the optimal solution of problem P (see (1.2)). The residual vector $r^{(k)}$ at $x^{(k)}$ for the constraints of problem P is defined by

$$r^{(k)} := A x^{(k)} - b. \tag{3.2}$$

Since (3.1) and (3.2) together with (1.1) imply

$$r^{(k)} = -M z^{(k)} - q,$$

it follows from (2.1) that

$$\begin{aligned} v^{(k)} &:= H z^{(k)} + q \\ &= (M - G) \, z^{(k)} + q \\ &= -G z^{(k)} - r^{(k)}. \end{aligned}$$

Since G is block-diagonal, we have

$$v_{J_\ell}^{(k)} = -G_\ell \, z_{J_\ell}^{(k)} - r_{J_\ell}^{(k)}, \qquad \ell = 1, \cdots, L, \tag{3.3}$$

which means that each subproblem $\mathrm{DSP}_\ell^{(k)}$ is completely defined using the data available within block ℓ.

Let $I^{(k)}$ denote the index set of active constrains of problem D at $z^{(k)}$ such that

$$I^{(k)} := \left\{ i \in I \;\middle|\; z_i^{(k)} = 0 \quad \text{and} \quad r_i^{(k)} \leq 0 \right\}.$$

Then, it is easy to see that $z^{(k)}$ and $r^{(k)}$ satisfy conditions (1.3) if and only if

$$r_i^{(k)} = 0, \qquad i \in E \cup \overline{I}^{(k)}, \tag{3.4}$$

where

$$\overline{I}^{(k)} = I \setminus I^{(k)}.$$

Thus, we may terminate the iteration of the splitting method when conditions (3.4) hold.

To summarize, we obtain the following block-parallel algorithm for problem P.

Algorithm BP :

Step 1. Choose an initial dual estimate $z^{(1)}$ with components $z_i^{(1)} \in R$, $i \in E$, and $z_i^{(1)} \geq 0$, $i \in I$. Set $k := 1$.

Step 2. Compute

$$\begin{aligned} x^{(k)} &:= -D^{-1}\left(A^\top z^{(k)} + c\right), \\ r^{(k)} &:= Ax^{(k)} - b, \end{aligned}$$

and let

$$\begin{aligned} I^{(k)} &:= \left\{ i \in I \,\middle|\, z_i^{(k)} = 0 \quad \text{and} \quad r_i^{(k)} \leq 0 \right\}, \\ \overline{I}^{(k)} &:= I \setminus I^{(k)}. \end{aligned}$$

If $r_i^{(k)} = 0$, $i \in E \cup \overline{I}^{(k)}$, then stop. (**comment:** $z^{(k)}$ and $x^{(k)}$ are optimal solutions of problem D and problem P, respectively.)

Step 3. For $\ell = 1, \cdots, L$, find the solutions $z_{J_\ell}^{(k+1)}$ of subproblems $\mathrm{DSP}_\ell^{(k)}$.

Step 4. Set $k := k + 1$ and go to Step 2. □

Step 3 of Algorithm BP can be executed in parallel for J_ℓ, $\ell = 1, \cdots, L$. We solve each subproblem $\mathrm{DSP}_\ell^{(k)}$ by applying a slight modification of the conjugate gradient (CG) method [16], which uses an active set strategy to deal with simple bounds.

Let the objective function of subproblem $\mathrm{DSP}_\ell^{(k)}$ be denoted by

$$\psi_\ell^{(k)}(z_{J_\ell}) = \frac{1}{2} z_{J_\ell}^\top G_\ell z_{J_\ell} + \left(v_{J_\ell}^{(k)}\right)^\top z_{J_\ell}.$$

The steepest descent direction of $\psi_\ell^{(k)}$ at $z_{J_\ell}^{(k)}$ is then calculated as

$$\begin{aligned} -\nabla\psi_\ell^{(k)}(z_{J_\ell}^{(k)}) &= -G_\ell z_{J_\ell}^{(k)} - v_{J_\ell}^{(k)} \\ &= r_{J_\ell}^{(k)}, \end{aligned}$$

where the last equality follows from (3.3). Thus, we can execute the CG iterations using $z_{J_\ell}^{(k)}$ and $r_{J_\ell}^{(k)}$ as an initial estimate of a solution and an initial search direction, respectively. In particular, there is no need to calculate the vector $v_{J_\ell}^{(k)}$ explicitly.

To improve the speed of convergence, we precondition G_ℓ as

$$\hat{G}_\ell = W_\ell G_\ell W_\ell^\top,$$

where W_ℓ is a nonsingular matrix chosen so that the condition number of $\hat{G}_\ell$ is smaller than that of G_ℓ [15, §9.2]. Assuming that the diagonal elements of G are all positive, we adopt here the simple preconditioning called *diagonal scaling* that uses

$$W_\ell = \operatorname{diag}\left\{\frac{1}{\sqrt{g_{ii}}}\right\}_{i\in J_\ell},$$

where g_{ii} are the diagonal elements of G. Then, since W_ℓ is positive and diagonal, subproblem $\mathrm{DSP}_\ell^{(k)}$ is equivalent to the problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\hat{z}_{J_\ell}^\top \hat{G}_\ell\, \hat{z}_{J_\ell} + \left(\hat{v}_{J_\ell}^{(k)}\right)^\top \hat{z}_{J_\ell} \\ \text{subject to} & \hat{z}_i \geq 0, \qquad\qquad i \in I \cap J_\ell, \end{array}$$

where $\hat{z}_{J_\ell} = W_\ell^{-1} z_{J_\ell}$ and $\hat{v}_{J_\ell}^{(k)} = W_\ell\, v_{J_\ell}^{(k)}$. The preconditioned conjugate gradient (PCG) method stated below is in essence a CG method applied to this problem. Now let $\hat{\psi}_\ell^{(k)}$ denote the objective function of the latter problem. Then, since $\nabla\hat{\psi}_\ell^{(k)}(\hat{z}_{J_\ell}) = W_\ell \nabla\psi_\ell^{(k)}(z_{J_\ell})$, the initial search direction $r_{J_\ell}^{(k)}$ is in particular modified as $Y_\ell\, r_{J_\ell}^{(k)}$, where $Y_\ell = W_\ell^2 = \operatorname{diag}\{1/g_{ii}\}_{i\in J_\ell}$.

Now we state the preconditioned CG procedure to find a solution of subproblem $\mathrm{DSP}_\ell^{(k)}$.

Procedure PCG :

Step 1. Let Y_ℓ ($= \operatorname{diag}\{y_i\}_{i\in J_\ell}$) $:= \operatorname{diag}\{1/g_{ii}\}_{i\in J_\ell}$. Set

$$\begin{aligned} z_{J_\ell} &:= z_{J_\ell}^{(k)}, \\ r_{J_\ell} &:= r_{J_\ell}^{(k)}, \\ E_\ell &:= E \cap J_\ell\,. \end{aligned}$$

Step 2. Set

$$\begin{aligned} I_\ell &:= \{\, i \in I \cap J_\ell \mid z_i = 0 \quad \text{and} \quad r_i \leq 0 \,\}, \\ \overline{I}_\ell &:= (I \cap J_\ell) \setminus I_\ell\,. \end{aligned}$$

If $r_i = 0$, $i \in E_\ell \cup \overline{I}_\ell$, then stop. (**comment:** z_{J_ℓ} is an optimal solution of subproblem $\mathrm{DSP}_\ell^{(k)}$.)

Step 3. Let p_{J_ℓ} be the vector such that

$$p_i := \begin{cases} 0, & i \in I_\ell\,, \\ y_i\, r_i, & i \in E_\ell \cup \overline{I}_\ell\,. \end{cases}$$

Step 4. Compute

$$\begin{aligned} s_{J_\ell} &:= G_\ell\, p_{J_\ell}\,, \\ \zeta_\ell &:= p_{J_\ell}^\top s_{J_\ell}\,, \\ \xi_\ell &:= p_{J_\ell}^\top r_{J_\ell}\,, \\ \alpha_\ell &:= \xi_\ell \,/\, \zeta_\ell. \end{aligned}$$

If $z_{J_\ell} + \alpha_\ell\, p_{J_\ell} \geq 0$, then go to Step 5. Otherwise, go to Step 6.

Step 5. Set

$$\begin{aligned} z_{J_\ell} &:= z_{J_\ell} + \alpha_\ell p_{J_\ell}, \\ r_{J_\ell} &:= r_{J_\ell} - \alpha_\ell s_{J_\ell}. \end{aligned}$$

If $r_i = 0$, $i \in E_\ell \cup \overline{I}_\ell$, then go to Step 2. Otherwise, let

$$\beta_\ell := \frac{1}{\xi_\ell} \sum_{i \in E_\ell \cup \overline{I}_\ell} y_i r_i^2,$$

$$p_i := \begin{cases} 0, & i \in I_\ell, \\ y_i r_i + \beta_\ell p_i, & i \in E_\ell \cup \overline{I}_\ell, \end{cases}$$

and go to Step 4.

Step 6. Calculate

$$\overline{\alpha}_\ell := \min \left\{ \frac{-z_i}{p_i} \,\middle|\, p_i < 0,\, i \in \overline{I}_\ell \right\}.$$

Set

$$\begin{aligned} z_{J_\ell} &:= z_{J_\ell} + \overline{\alpha}_\ell p_{J_\ell}, \\ r_{J_\ell} &:= r_{J_\ell} - \overline{\alpha}_\ell s_{J_\ell}, \\ I_\ell &:= \{ i \in I \cap J_\ell \mid z_i = 0 \}, \\ \overline{I}_\ell &:= (I \cap J_\ell) \setminus I_\ell. \end{aligned}$$

If $r_i = 0$, $i \in E_\ell \cup \overline{I}_\ell$, then go to Step 2. Otherwise, go to Step 3. □

In the remainder of this section, some examples of splitting (G, H) are described. The simplest choice is to let G_ℓ be a scalar multiple of the block diagonal part of matrix M, that is,

$$G_\ell := \omega^{-1} M_\ell, \tag{3.5}$$

where $\omega > 0$ is a relaxation parameter and M_ℓ denotes the submatrix of M consisting of the elements $a_i^\top D^{-1} a_j$, $(i,j) \in J_\ell \times J_\ell$. Clearly G is positive semi-definite. In addition, if the constraint matrix A of problem P has some block structure, the off-block-diagonal parts of matrix M are supposed to be highly sparse. Then, a relatively large value of the relaxation parameter ω can be chosen and hence fast convergence is expected. Moreover, since each element of G_ℓ is of the form $\omega^{-1} a_i^\top D^{-1} a_j$, the matrix-vector multiplications involving G_ℓ in Procedure PCG can effectively be executed using only simple operations on the rows of A.

If matrices M_ℓ are positive definite (they are positive semi-definite by construction), then the splitting (G, H) given by (3.5) is regular, provided that parameter ω is chosen sufficiently small. On the other hand, when M_ℓ are not positive definite, the splitting (G, H) is not necessarily regular. In such a case, by choosing G_ℓ as

$$G_\ell := M_\ell + \omega^{-1} \mathcal{I}_\ell \qquad \text{or} \qquad G_\ell := \omega^{-1} (M_\ell + \mathcal{I}_\ell),$$

where $\mathcal{I}_\ell \in R^{|J_\ell| \times |J_\ell|}$ is the identity matrix, the splitting (G, H) becomes regular for sufficiently small $\omega > 0$.

The above-mentioned choices of G yield sparsity-preserving CG iterations, and hence enable us to deal effectively with large-scale problems.

4. Implementation Strategy

Algorithm BP is of the control parallel type in the sense that multiple quadratic programming subproblems may be solved concurrently. This type of parallel algorithms can be implemented on Multiple-Instruction Multiple-Data (MIMD) systems, where individual processors run under the control of their own program. For the proposed algorithm, however, the programs in the individual processors are nearly identical because we solve each subproblem $\mathrm{DSP}_\ell^{(k)}$ by applying Procedure PCG. In such a case, the SPMD model of computation is convenient, in which a single program is written and all processors execute this program on their own data independently. It is also appropriate that the proposed algorithm is implemented on a distributed memory system, in which processors communicate by exchanging messages through a interconnecting network.

Now we describe the implementation strategy for the proposed algorithm. Under the multi-processor environment, one processor, which we refer to as the *master* node, controls major iterations, while the remaining processors called *worker* nodes concentrate on inner iterations. More precisely, subproblems $\mathrm{DSP}_\ell^{(k)}$ are solved using Procedure PCG on worker nodes in parallel. Then, making use of massage passing primitives, their solutions $z_{J_\ell}^{(k+1)}$ are transmitted to the master node, which executes Step 2 of Algorithm BP. If conditions (3.4) are satisfied, the master node broadcasts the termination of the algorithm. Otherwise, the master node transmits the subvectors $r_{J_\ell}^{(k+1)}$ to the corresponding worker nodes. Then, using $r_{J_\ell}^{(k+1)}$ together with the previous solution $z_{J_\ell}^{(k+1)}$, each worker node again executes Procedure PCG to solve subproblem $\mathrm{DSP}_\ell^{(k+1)}$, and so forth.

The above mentioned strategy, which naturally follows from the construction of Algorithm BP, is slightly inefficient because each worker node remains idle while the master node executes Step 2 of Algorithm BP. In fact, the most time consuming part of the algorithm is the matrix-vector multiplications $A^\top z^{(k)}$ and $Ax^{(k)}$. To reduce idle time, we modify the implementation strategy in the following way. Since Procedure PCG consists of simple operations on the rows of submatrix A_{J_ℓ}, we replace the message $z_{J_\ell}^{(k+1)}$ from each worker node to the master node by $D^{-1}A_{J_\ell}^\top z_{J_\ell}^{(k+1)}$. Then, the master node calculates an estimate $x^{(k+1)}$ of the optimal solution of problem P simply as

$$x^{(k+1)} = -\sum_{\ell=1}^{L} D^{-1}A_{J_\ell}^\top z_{J_\ell}^{(k+1)} - D^{-1}c,$$

and immediately broadcasts it to all worker nodes. Moreover, before solving subproblem $\mathrm{DSP}_\ell^{(k+1)}$, each worker node independently calculates the subvector of the residual $r^{(k+1)}$ as

$$r_{J_\ell}^{(k+1)} = A_{J_\ell}x^{(k+1)} - b_{J_\ell}.$$

On the other hand, the residual $r^{(k+1)}$ is also calculated on the master node and the termination criterion (3.4) is checked while the worker nodes are solving subproblems $\mathrm{DSP}_\ell^{(k+1)}$. An example of the execution sequence for the proposed algorithm is illustrated in Figure 1.

Finally, we briefly describe how matrix A, which is decomposable into smaller sparse blocks A_{J_ℓ}, is represented. On the master node and each worker node, all working variables used in Algorithm BP and Procedure PCG are stored in one-dimensional arrays whose length are equal to the non-zero elements of A and A_{J_ℓ}, respectively. Moreover, we store the non-zero elements of a_i in a contiguous segment for each index $i \in J_\ell$. Then, the segmented scan operation [29, §3.2] enables us to perform matrix-vector multiplications associated with matrix A or its submatrices A_{J_ℓ} effectively on their segments. If data level parallelism is available on the master node and each worker node, these multiplications are executed, in fact, in parallel.

5. Numerical Results

In this section, we report some numerical results with the proposed algorithm on the Connection Machine Model CM-5 at ATR Human Information Processing Research Laboratories. The particular configuration of the CM-5 is 32 processing nodes (PN's). Each PN has 4 vector floating point units and 4 banks of 8 Mbytes local memories. The performance of each PN is estimated as 128 MFLOPS (peak) [26]. On the CM-5, the SPMD model of computation is available, in which various algorithms of control parallel type can effectively be implemented as well as those of data parallel type. The procedure to be executed on each PN was coded in the data parallel language CM Fortran [25] together with the software library CMMD [27], which provides various routines for realizing control parallel primitives such as global synchronization and message passing between PN's. Under the mechanism of virtual processors [29, §3.1], each PN on the Connection Machine system operates in a serial fashion on multiple copies of data as if multiple PN's operate on their own copy of data. Since CM Fortran is an extension of FORTRAN specially designed for computations on large arrays, inner products of vectors and sparse matrix-vector multiplications involved in Algorithm BP and Procedure PCG may effectively be performed on each PN.

We tested the proposed algorithm on randomly generated sparse quadratic programming problems of the following form:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^\top Dx + c^\top x \\ \text{subject to} & Ax \leq b, \end{array} \tag{5.1}$$

where D is an $n \times n$ positive diagonal matrix, c is a vector in R^n, A is an $m \times n$ matrix with some block structure and b is a vector in R^m. The non-zero elements of A and all elements of b, c and D were chosen uniformly from the intervals $[-5, 5]$, $[1, 10]$, $[-100, 100]$ and $[1, 10]$, respectively. Note that any problem thus generated is feasible, because $b > 0$ implies that the origin $x = 0$ satisfies the constraints. We used two sets of test problems; one with staircase structure, the other with block angular structure. Each set consists of problems with various size. For each problem size, we generated five problem instances using different random number seeds. The results shown in the tables and figures below are the averages over the five test problems generated.

Algorithm BP combined with Procedure PCG can be applied to problem (5.1) by setting $E = \emptyset$. In Step 2 of Algorithm BP, we stop the iteration if the conditions

$$|\, r_i^{(k)}| \leq 10^{-7} \parallel b \parallel_\infty$$

or

$$z_i^{(k)} = 0, \quad r_i^{(k)} \leq 10^{-7} \parallel b \parallel_\infty$$

hold for all $i \in \{1, \cdots, m\}$. On the other hand, we terminate Procedure PCG for solving subproblem $\mathrm{DSP}_\ell^{(k)}$ when the current iterate satisfies either

$$|\, r_i \,| \le \varepsilon_\ell^{(k)} \parallel b_{J_\ell} \parallel_\infty$$

or

$$z_i = 0, \quad r_i \le \varepsilon_\ell^{(k)} \parallel b_{J_\ell} \parallel_\infty$$

for all $i \in J_\ell$, where $\varepsilon_\ell^{(k)}$ are sufficiently small positive numbers. It should be noted that the tolerances $\varepsilon_\ell^{(k)}$ have to be 10^{-7} or less at the final stage of the major iterations. From a practical viewpoint, however, it is often effective to truncate inner iterations using relatively loose convergence criteria when the current (major) iterate is far from the optimal solution. Thus, we introduce the following strategy:

$$\begin{aligned} \varepsilon_\ell^{(1)} &:= \max \left\{ \frac{r_i^{(1)}}{\parallel b_{J_\ell} \parallel_\infty} \,\middle|\, i \in J_\ell \right\}, \\ \varepsilon_\ell^{(k)} &:= \max \left\{ 10^{-7}, \frac{\varepsilon_\ell^{(k-1)}}{10} \right\}, \qquad k > 1. \end{aligned} \tag{5.2}$$

Throughout the computational experiments, the initial point for Algorithm BP was always chosen as $z^{(1)} = 0$, and the matrices G_ℓ involved in subproblem $\mathrm{DSP}_\ell^{(k)}$ were set to be the block diagonal part of matrix M as in (3.5). Moreover, we assume that each worker node can be assigned to solve its own subproblem $\mathrm{DSP}_\ell^{(k)}$ exclusively. All computations were done in double precision arithmetic.

5.1. Staircase constraints

We first tested the proposed algorithm on problems of the form (5.1), in which matrix A has the staircase structure

$$A = \begin{pmatrix} A_{11} & A_{12} & & & & & & \\ & A_{21} & A_{22} & A_{23} & & & & \\ & & & A_{32} & A_{33} & A_{34} & & \\ & & & & & \ddots & & \\ & & & & & A_{Q-1\,Q-2} & A_{Q-1\,Q-1} & A_{Q-1\,Q} \\ & & & & & & A_{Q\,Q-1} & A_{Q\,Q} \end{pmatrix}.$$

This type of problems often arise in multi-period planning and discrete-time dynamic control. Let the diagonal matrix D be symmetrically partitioned to conform with the column partition of A, i.e.,

$$D = \mathrm{diag}\,\{D_{11}, D_{12}, D_{22}, D_{23}, D_{33}, D_{34}, \cdots, D_{Q-1\,Q-1}, D_{Q-1\,Q}, D_{Q\,Q}\}.$$

Then, matrix M given by (1.1) is written as the block tri-diagonal matrix

$$M = \begin{pmatrix} M_{11} & M_{12} & & & \\ M_{21} & M_{22} & M_{23} & & \\ & M_{32} & M_{33} & \ddots & \\ & & \ddots & \ddots & M_{Q-1\,Q} \\ & & & M_{Q\,Q-1} & M_{Q\,Q} \end{pmatrix}, \tag{5.3}$$

where

$$\begin{array}{lcll} M_{q\,q} & = & A_{q\,q-1}D_{q-1\,q}A_{q\,q-1}^{\top} + A_{q\,q}D_{q\,q}A_{q\,q}^{\top} + A_{q\,q+1}D_{q\,q+1}A_{q\,q+1}^{\top}\,, & q=1,\cdots,Q, \\ M_{q\,q+1} & = & A_{q\,q+1}D_{q\,q+1}A_{q+1\,q}^{\top} & \\ & = & M_{q+1\,q}^{\top}\,, & q=1,\cdots,Q-1, \end{array}$$

with $A_{1\,0} = A_{Q\,Q+1} = 0$. When we can use as many worker nodes as the number Q of blocks, we simply determine G_ℓ using the block diagonal part $M_{\ell\,\ell}$ of matrix M. Otherwise, we set, for example,

$$M_\ell = \begin{pmatrix} M_{q_\ell\,q_\ell} & M_{q_\ell\,q_\ell+1} & 0 & 0 \\ M_{q_\ell+1\,q_\ell} & M_{q_\ell+1\,q_\ell+1} & M_{q_\ell+1\,q_\ell+2} & 0 \\ 0 & M_{q_\ell+2\,q_\ell+1} & M_{q_\ell+2\,q_\ell+2} & M_{q_\ell+2\,q_\ell+3} \\ 0 & 0 & M_{q_\ell+3\,q_\ell+2} & M_{q_\ell+3\,q_\ell+3} \end{pmatrix}.$$

Since the block diagonal matrix G given by (3.5) does not differ so much from the block tri-diagonal matrix M of (5.3), we may expect that the proposed algorithm can effectively be applied to problems with staircase structure.

Now let $A_q \in R^{m_q \times n_q}$ denote the row block of matrix A such that

$$A_q = \begin{cases} (A_{1\,1}, A_{1\,2})\,, & q=1, \\ (A_{q\,q-1}, A_{q\,q}, A_{q\,q+1})\,, & q=2,\cdots,Q-1, \\ (A_{Q\,Q-1}, A_{Q\,Q})\,, & q=Q. \end{cases}$$

Then a problem with staircase structure can be characterized by the number Q of blocks, the size (m_q, n_q) of each block and the size (m, n) of the whole matrix A. We generated test problems with various size, in which the ratio of m to n is fixed at 1/4, the size of row blocks A_q are identical and the number of non-zero elements in each matrix A_q is fixed at 8192. The overlap of two consecutive blocks is either about 10% or about 20% in terms of the number of columns. The characteristics of the test problems are shown in Table 1.

First of all, we made a preliminary experiment to verify the effects of the strategy based on (5.2). Test problem QP12 was used in this experiment. For comparison purposes, we also solved this test problem under the same termination criterion for inner iterations with the fixed tolerance $\varepsilon_\ell^{(k)} = 10^{-7}$ for all ℓ and k. Table 2 shows the performance of the two strategies when the number L of worker nodes is 2 and 8. Observe that the truncation strategy (5.2) for inner iterations increases the number of major iterations very little and, more importantly, reduces the total computation time significantly.

We next examined the performance of the proposed algorithm under various choices of relaxation parameter ω. In this experiment, test problems QP11–QP13 were solved using L worker nodes, where L varies from 2 to Q. The results are shown in Figure 2, in which both axes represent logarithmic scale. The symbol $\odot$ in the figure indicates the total computation time in the case of $L = 1$ and $\omega = 1.0$, which means that problem D is not split into subproblems and the whole problem is solved by the preconditioned conjugate gradient method (Procedure PCG) on a single PN. It is interesting to observe that the algorithm initially tends to converge faster as ω increases, but it begins to slow down when ω exceeds 1.0, and eventually fails to converge when ω becomes 1.4.

Then some additional experiments were made to observe the behavior of the proposed algorithm for problems with various size. We solved problems QP14–QP17 by setting relaxation parameter $\omega = 1.0$. Figure 3 shows the total computation time for several choices of the number L of worker nodes. Note that, for problem QP17 with 16 blocks, the case $L = 16$ required more computation time than the case $L = 8$. This is probably because, when 16 worker nodes are used, the communication overhead between the master and worker nodes becomes relatively large in comparison with the computation time spent at each worker node executing Procedure PCG.

5.2. Block angular constraints

We also tested the proposed algorithm on problems of the form (5.1), in which matrix A has the block angular structure

$$A = \begin{pmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_Q \\ \overline{A}_1 & \overline{A}_2 & \cdots & \overline{A}_Q \end{pmatrix},$$

where A_q and $\overline{A}_q$ are matrices in $R^{m_q \times n_q}$ and $R^{\overline{n} \times n_q}$, respectively. A typical example of problems with block angular structure is a multi-commodity flow problem. Corresponding to the column partition of A, we may rewrite the diagonal matrix D as

$$D = \mathrm{diag}\left\{D_1, D_2, \cdots, D_Q\right\},$$

where D_q are $n_q \times n_q$ positive diagonal matrices. Then, the matrix M given by (1.1) is written as follows:

$$M = \begin{pmatrix} M_1 & & & & \overline{M}_1 \\ & M_2 & & & \overline{M}_2 \\ & & \ddots & & \vdots \\ & & & M_Q & \overline{M}_Q \\ \overline{M}_1^\top & \overline{M}_2^\top & \cdots & \overline{M}_Q^\top & M_{Q+1} \end{pmatrix},$$

where

$$\begin{array}{lcll} M_q & = & A_q D_q A_q^\top, & q = 1, \cdots, Q, \\ \overline{M}_q & = & A_q D_q \overline{A}_q^\top, & q = 1, \cdots, Q, \\ M_{Q+1} & = & \sum_{q=1}^{Q} \overline{A}_q D_q \overline{A}_q^\top. & \end{array}$$

When $Q+1$ worker nodes are available, (3.5) naturally suggests the simple choice of matrix G such that

$$G_\ell = \omega^{-1} M_\ell, \qquad \ell = 1, \cdots, L,$$

where $L = Q + 1$. Otherwise, we set, for example,

$$G_\ell = \omega^{-1}\,\mathrm{diag}\left\{M_{q_\ell}, M_{q_\ell+1}, \cdots, M_{q_{\ell+1}-1}\right\}, \qquad \ell = 1, \cdots, L-1,$$

$$G_L = \omega^{-1} \begin{pmatrix} M_{q_L} & & & & \overline{M}_{q_L} \\ & M_{q_L+1} & & & \overline{M}_{q_L+1} \\ & & \ddots & & \vdots \\ & & & M_Q & \overline{M}_Q \\ \overline{M}_{q_L}^\top & \overline{M}_{q_L+1}^\top & \cdots & \overline{M}_Q^\top & M_{Q+1} \end{pmatrix},$$

where $1 = q_1 < q_2 < \cdots < q_L \leq Q+1$.

Now let $\overline{A} \in R^{\overline{n} \times n}$ denote the coefficient matrix of the coupling constraints, i.e.,

$$\overline{A} = \left(\overline{A}_1, \overline{A}_2, \cdots, \overline{A}_Q\right).$$

We generated test problems with various size, in which $m_q = \overline{m}$ and $n_q = \overline{n}$ for all $q = 1, \cdots, Q$, the ratio of $\overline{m}$ to $\overline{n}$ is fixed at $1/4$ and the number of non-zero elements in each matrix A_q is fixed at 8192. The characteristics of the test problems are shown in Table 3.

The performance of the proposed algorithm under various choices of relaxation parameter ω is examined. Figure 4 shows the results for problems QP21–QP23. The general feature of the results is similar to that for problems with staircase structure. Note that, when $\omega \geq 1.4$, the algorithm fails to solve all the test problems generated for any choice of L.

We then solved problems QP24–QP26 to ascertain the effectiveness of the proposed algorithm for problems with various size. In this experiment, the relaxation parameter ω is fixed at 1.0. Figure 5 shows the relationship between the computation time and the problem size measured in term of the number $Q+1$ of blocks, for several choices of the number L of worker nodes.

It is well known that the concept of *efficiency* is useful to measure what is gained by parallel computation (see, for example, [1, p. 14] and [15, p. 65]). We define the efficiency e_p of the proposed algorithm as follows:

$$e_p := \frac{\overline{T}}{p\, T_p},$$

where $\overline{T}$ is the total computation time in the case of single PN, while T_p is the total computation time using p PN's. Since the algorithm is executed using L worker nodes and one master node, p is related to L by $p = L+1$. Note that e_p may exceed 1, because $\overline{T}$ represents the computation time of the serial algorithm that does not utilize splitting but essentially applies Procedure PCG to problem D directly. Table 4 shows the efficiency of the proposed algorithm for problems QP24–QP26. It can be seen from the table that high efficiency is achieved, especially when the number of blocks is large.

6. Conclusion

We have presented a block-parallel modification of the conjugate gradient method for solving separable quadratic programming problems. As shown in the preceding sections, the proposed algorithm can effectively be implemented on a parallel computer on which each processor can execute its own operations. The numerical results reported in Section 5 indicate that the present approach is particularly effective when the problems to be solved have some block structure.

The proposed algorithm is synchronous in the sense that each processor has to complete its conjugate gradient iterations before the algorithm proceeds to the next major iteration. Thus the performance of the algorithm could be seriously affected if the amounts of conjugate gradient computations per major iteration significantly vary from one processor to another. De Leone and Mangasarian [6] have presented an asynchronous parallel SOR method for the linear complementarity problem. It is one of the interesting future topics to develop an asynchronous version of the algorithm proposed in this paper.

Acknowledgments

The authors thank Mr. Tsukasa Kimezawa of Advanced Systems Company Limited for valuable discussions on the implementation strategy and the computer programs on the CM-5.

References

[1] Bertsekas, D.P. and Tsitsiklis, J.N.: *Parallel and Distributed Computation: Numerical Methods.* Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[2] Censor, Y.: Parallel Application of Block-Iterative Methods in Medical Imaging and Radiation Therapy. *Mathematical Programming*, Vol. 42 (1988), 307-325.

[3] Chen, R.J. and Meyer, R.R.: Parallel Optimization for Traffic Assignment. *Mathematical Programming*, Vol. 42 (1988), 327-345.

[4] Chen, G. and Teboulle, M.: A Proximal-Based Decomposition Method for Convex Minimization Problems. *Mathematical Programming*, Vol. 64 (1994), 81-101.

[5] Cottle, R.W., Pang, J.-S. and Stone, R.E.: *The Linear Complementarity Problem.* Academic Press, San Diego, California, 1992.

[6] De Leone, R. and Mangasarian, O.L.: Asynchronous Parallel Successive Overrelaxation for the Symmetric Linear Complementarity Problem. *Mathematical Programming*, Vol. 42 (1988), 347-361.

[7] Eckstein, J.: The Alternating Step Method for Monotropic Programming on the Connection Machine CM-2. *ORSA Journal on Computing*, Vol. 5 (1993), 84-96.

[8] Eckstein, J. and Bertsekas, D.P.: On the Douglas-Rachford Splitting Method and the Proximal Point Algorithm for Maximal Monotone Operators. *Mathematical Programming*, Vol. 55 (1992), 293-318.

[9] Eckstein, J. and Fukushima, M.: Some Reformulations and Applications of the Alternating Direction Method of Multipliers. *Large Scale Optimization: State of the Art* (eds. W.W. Hager, D.W. Hearn and P.M. Pardalos). Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994, 115-134.

[10] Ferris, M.C.: Parallel Constraint Distribution in Convex Quadratic Programming. *Mathematics of Operations Research*, Vol. 19 (1994). 645-658.

[11] Frommer, A. and Mayer, G.: On the Theory and Practice of Multisplitting Methods in Parallel Computation. *Computing*, Vol. 49 (1992), 63-74.

[12] Fukushima, M.: Application of the Alternating Direction Method of Multipliers to Separable Convex Programming Problems. *Computational Optimization and Applications*, Vol. 1 (1992), 93-111.

[13] Fukushima, M.: The Primal Douglas-Rachford Splitting Algorithm for a Class of Monotone Mappings with Application to the Traffic Equilibrium Problem. *Mathematical Programming*, submitted for publication.

[14] Fukushima, M., Haddou, M., Nguyen, V.H., Strodiot, J.-J., Sugimoto, T. and Yamakawa, E.: A Parallel Descent Algorithm for Convex Programming. *Computational Optimization and Applications*, to appear.

[15] Golub, G. and Ortega, J.M.: *Scientific Computing: An Introduction with Parallel Computing.* Academic Press, San Diego, California, 1993.

[16] Hestenes, M.R.: *Conjugate Direction Methods in Optimization.* Springer-Verlag, New York, 1980.

[17] Iusem, A.N.: On the Convergence of Iterative Methods for Symmetric Linear Complementarity Problems. *Mathematical Programming*, Vol. 59 (1993), 33-48.

[18] Luo, Z.-Q. and Tseng, P.: On the Convergence of a Matrix Splitting Algorithm for the Symmetric Monotone Linear Complementarity Problem. *SIAM Journal on Control and Optimization*, Vol. 29 (1991), 1037-1060.

[19] Luo, Z.-Q. and Tseng, P.: Error Bound and Convergence Analysis of Matrix Splitting Algorithms for the Affine Variational Inequality Problem. *SIAM Journal on Optimization*, Vol. 2 (1992), 43-54.

[20] Machida, N., Fukushima, M. and Ibaraki, T.: A Multisplitting Method for Symmetric Complementarity Problems. *Journal of Computational and Applied Mathematics*, to appear.

[21] Mangasarian, O.L.: *Nonlinear Programming.* McGraw-Hill, New York, 1969.

[22] Mangasarian, O.L. and De Leone, R.: Parallel Successive Overrelaxation Methods for Symmetric Linear Complementarity Problems and Linear Programs. *Journal of the Optimization Theory and Applications*, Vol. 54 (1987), 437-446.

[23] Mangasarian, O.L. and De Leone, R.: Parallel Gradient Projection Successive Overrelaxation for Symmetric Linear Complementarity Problems. *Annals of Operations Research*, Vol. 14 (1988), 41-59.

[24] Nielsen, S.S. and Zenios, S.A.: Proximal Minimizations with D-Functions and the Massively Parallel Solution of Linear Network Programs. *Computational Optimization and Applications*, Vol. 1 (1993), 375-398.

[25] Thinking Machines Corporation: *CM Fortran Programming Guide, Version 2.1.* Cambridge, Massachusetts, 1994.

[26] Thinking Machines Corporation: *CM-5 CM Fortran Performance Guide, Version 2.1.* Cambridge, Massachusetts, 1994.

[27] Thinking Machines Corporation: *CMMD Reference Manual, Version 3.0.* Cambridge, Massachusetts, 1993.

[28] Zenios, S.A.: Data Parallel Computing for Network-Structured Optimization Problems. *Computational Optimization and Applications*, Vol. 3 (1994), 199-242.

[29] Zenios, S.A. and Censor, Y.: Massively Parallel Row-Action Algorithms for Some Nonlinear Transportation Problems. *SIAM Journal on Optimization*, Vol. 1 (1991), 373-400.

Eiki YAMAKAWA :
Department 6, ATR Human Information
Processing Research Laboratories,
Seika, Soraku, Kyoto 619-02, Japan.

Masao FUKUSHIMA :
Graduate School of Information Science,
Nara Institute of Science and Technology,
Ikoma, Nara 630-01, Japan.

Table 1: Characteristics of the test problems with staircase structure.

Problem	Q	Each Block†			Total	
		m_q	n_q	Density of A_q (%)	m	n
QP11	4	128	554	11.55	512	2048
QP12	8	128	561	11.41	1024	4096
QP13	16	128	564	11.35	2048	8192
QP14	2	256	1138	2.812	512	2048
QP15	4	256	1204	2.658	1024	4096
QP16	8	256	1241	2.579	2048	8192
QP17	16	256	1264	2.538	4096	16384

† The overlap of two consecutive blocks is about 10% in terms of the number of columns for problems QP11–QP13, and about 20% for problems QP14–QP17.

Table 2: Comparison of the truncation strategies for inner iterations.

$\varepsilon_\ell^{(k)}$	$L = 2$		$L = 8$	
	Number of Iterations Major / Inner	CPU (sec)	Number of Iterations Major / Inner	CPU (sec)
Strategy (5.2)	19.2 / 124.0	30.74	27.0 / 510.6	13.88
Fixed at 10^{-7}	18.6 / 569.2	107.11	27.2 / 2349.6	41.21

Table 3: Characteristics of the test problems with block angular structure.

		Each Block†			Total	
Problem	Q	$\overline{m}$	$\overline{n}$	Density of A_q (%)	m	n
QP21	3	128	512	12.5	896	1536
QP22	7	128	512	12.5	1408	3584
QP23	15	128	512	12.5	2432	7680
QP24	3	256	1024	3.125	1792	3072
QP25	7	256	1024	3.125	2816	7168
QP26	15	256	1024	3.125	4684	15360

† The number of non-zero elements in $\overline{A}$ is 4096 for problems QP21 and QP24, 8192 for problems QP22 and QP25, and 16384 for problems QP23 and QP26.

Table 4: Efficiency e_p of the proposed algorithm.†

	Number of PN's			
Problem	$p = 3$	$p = 5$	$p = 9$	$p = 17$
QP24	0.5645	0.5383	–	–
QP25	0.7990	0.8853	0.7625	–
QP26	1.381	1.465	1.299	0.6777

† For each problem, the number L of worker nodes varies from 2 to $Q + 1$. Note that p equals $L + 1$, because one master node is used besides L worker nodes.