

Some Applications via a Parallel Interior-Point Method

C. Durazzi, V. Ruggiero, G. Zanghirati

Department of Mathematics, University of Ferrara

Abstract

Si riportano i risultati ottenuti utilizzando un metodo del punto interno combinato con l'algoritmo dei gradienti coniugati preconditionati per la risoluzione di alcune applicazioni di grandi dimensioni su Cray T3E ed SGI Origin 2000. La prima applicazione riguarda problemi di programmazione stocastica e di ottimizzazione robusta, mentre la seconda deriva dalla riformulazione di una speciale classe di problemi di controllo ottimo nel discreto.

We report the results obtained by a parallel Interior-Point method combined with the Preconditioned Conjugate Gradient algorithm for the solution of some large-scale applications on Cray T3E and SGI Origin 2000. The first application concerns stochastic programming and robust optimization problems and the second one arises from a reformulation of a special class of discrete optimal control problems.

1. Introduction

Many real-world applications require the numerical solution of optimization problems with a very large number of linear constraints as, for instance, multicommodity network flow problems, stochastic programming and robust optimization problems. Also special classes of optimal control problems can be reformulated as quadratic programs. All these problems have a very large and structured matrix defining the linear constraints.

For their solution, Interior-Point (IP) methods can be applied: the present state-of-the-art codes based on IP methods use direct method for the solution of the inner system occurring at each iteration.

To exploit the structure of these problems which is convenient for the parallel computing, in some recent works we have introduced an iterative solver within the IP methods. In particular, for linear and quadratic programming (LP, QP) pro-

blems, we have proposed an infeasible primal-dual IP algorithm combined with the Preconditioned Conjugate Gradient algorithm as inner solver. This method is named IPPCG method [1]. We have developed a parallel implementation of the IPPCG method for linear stochastic programming (SLP) and robust optimization (RO) problems [1] and for a special class of discrete optimal control problems [2].

2. Statement of the problem

SLP and RO are used in a wide range of practical applications with uncertain input data. When the input data are discretely distributed, two-stages SLP and RO can be formulated as deterministic linear or quadratic problems with a block bian-gular constraint matrix of the form

$$A = \begin{pmatrix} A_0 & & & & & \\ T_1 & W_1 & & & & \\ T_2 & & W_2 & & & \\ \vdots & & & \ddots & & \\ T_N & & & & & W_N \end{pmatrix} \quad (1)$$

where N is the number of realizations (called “scenarios”) of the uncertain data and $A_0 \in R^{m_0 \times n_0}$, $T_i \in R^{m_i \times n_0}$, $W_i \in R^{m_i \times n_i}$ with $\sum_{i=0}^N m_i = m$, $\sum_{i=0}^N n_i = n$. For a large number of scenarios, these problems become extremely large but remain extremely sparse and structured.

Within the class of optimal control problems, we have considered two cases: the first is the well-known problem of improving water quality by means of in-stream aeration process¹, while the other one concerns the study of diffusion-convection processes. Both these classical problems have a continuous formulation where the objective function is given by a double integral, the dynamic system is described by a system of partial differential equations and the unknown state and control depend on time and space.

In order to apply the IPPCG method, we consider a discretization of these problems, dividing the space interval into N subintervals of width Δh and the time interval into M subintervals of width Δt . The discretization is carried out in two phases: in the first one, the system of partial differential equations is discretized only respect to the space mesh points so that we obtain a continuous dynamic system of first order differential equations; then using a finite difference scheme,

¹ Hullet W., Optimal Estuary Aeration: *An Application of Distributed Parameter Control Theory*, Applied Mathematics & Optimization, Vol. 1, No 1, pp. 20-63, 1974.

we obtain a system of difference equations². The final formulation of these problems is given by a quadratic objective function with a diagonal positive semidefinite Hessian matrix and a constraint matrix having the following form

$$A = \left(\begin{array}{cccc|cccc} \Lambda & & & & \Delta t B & & & \\ \Omega & \Lambda & & & & \ddots & & \\ & \ddots & \ddots & & & & \ddots & \\ & & & \Omega & \Lambda & & & \\ & & & & & & & \Delta t B \end{array} \right) \quad (2)$$

where $A \in R^{m \times n}$ and Ω, Λ, B are matrices with special structure (diagonal, bidiagonal and tridiagonal).

3. Parallel approach and results

The structure of the matrix A , in both cases (1) and (2), enables us to individuate, for the Conjugate Gradient method, a suitable block-diagonal preconditioner M , where each block is a sparse matrix.

In the parallel implementation of the IPPCG method, we consider a block decomposition of the m and n -vectors and we distribute these blocks on the available processors.

The most part of the calculations are trivially carried out using parallel vector “saxpy” operations. Three procedures, exploiting the structure of the problem, are required: the first and the second are the parallel computation of the matrix-vector product $v = Aw$ and $w = A^T v$, while the third is the parallel computation of the preconditioner.

Two additional procedures are used to compute in parallel the factorization of each block of M and to solve the related system. For SLP and RO problems, an *a priori* analysis avoids the computation of the null elements of the diagonal blocks of M at each iteration of the IPPCG method. To exploit the sparsity of these blocks, their factorization is obtained by a Fortran package (version 0.3) of Ng and Peyton. This package computes *a priori* the symbolic factor of the blocks, using the multiple minimum degree ordering of Liu to minimize the fill-ins in this factor and the supernodal block factorization.³ By routines included in the package, each diagonal

² C. Durazzi, E. Galligani, *Nonlinear Programming Methods for Solving Optimal Control Problems*, Equilibrium Problems and Variational Models, Edited by F. Giannessi, A. Maugeri, and P. M. Pardalos, Kluwer Academic Publishers, Dordrecht, Netherlands (2001).

³ Liu, J. W., Ng, E. G., and Peyton B. W., *On Finding Supernodes for Sparse Matrix Computations*, SIAM Journal on Matrix Analysis and Applications, Vol. 1, pp. 242-252, 1993

block is factorized and solved. In the application arising in optimal control problems, the Cholesky factorization of the tridiagonal or pentadiagonal blocks of M is computed and used to solve the related system.

The IPPCG algorithm has been implemented in a parallel code to test its effectiveness on the described problems.

The interprocessor communications are based on standard MPI⁴. There exists also a version of the code where the communications are implemented via the routine of the SHared MEMory access library (SHMEM)⁵ of Cray T3E/256 and Origin 2000/64 at CINECA (Bologna, Italy).

Table I shows the speedup on Cray T3E for some test-problems belonging to the collection of the deterministic equivalent of the SLP problems (<ftp://ftp.sztaki.hu/pub/oplab/LPTESTSET/STOCHLP>): these are LP problems, while RO models are obtained by adding a quadratic term, given by a randomly generated diagonal matrix, to the objective function of SLP problems. In all cases, the spectral condition number of this matrix is 10^4 , the minimum diagonal entry is 10^{-2} and consequently the maximum is 10^2 . The letter “q” at the end of the name indicates that the problem is quadratic. The relative speedup is the ratio of the elapsed time for the execution of IPPCG method on one processor to the time on a prefixed number of processors. The number of processors is denoted by “Procs” in the tables. In some cases, the speedup is greater than the number of the considered processors, because we have observed that the total number of PCG iterations generally decreases as the number of processors increases. This is due to the different rounding error accumulation in the scalar products occurring in the PCG method: indeed, each scalar product is distributed on all processors and the arrangement of

Problem name	m	n	Relative speedup		
			12 Procs	36 Procs	108 Procs
sctap12.108	6510	10416	11.9	37.9	63.2
sctap12.216	12990	20784	11.4	35.7	91.9
scsd82.108	2170	15190	11.7	32.6	53.5
scsd82.216	4330	30310	11.2	32.7	87.1
scsd82.432	8650	60550	9.4	27.5	76.8
scsd82.432q	8650	60550	9.5	28.3	77.2
			10 Procs	60 Procs	120 Procs
sctap12.480	28830	46128	9.7	58.9	115.5
sctap12.480q	28830	46128	10.4	59.6	108.5
			10 Procs	40 Procs	100 Procs
sc205.200	4413	4414	12.5	30.0	33.2
sc205.400	8813	8814	12.2	41.4	61.7
sc205.800	17613	17614	10.6	46.9	89.0
sc205.800q	17613	17614	8.5	49.6	99.9

Table I Relative speedup for SLP and RO problems on Cray T3E

⁴ MPI: *A Message-Passing Interface Standard*, University of Tennessee, Knoxville, Tennessee, 1995

⁵ SHMEM *Technical Note for Fortran*, Cray Research, Seattle, Washington, 1994

the sums differs as the number of processors changes. For example, the number of outer iterations of the IPPCG method for sctap12.480 is 50, while the total number of inner PCG iterations is 1804 on 1 processor, 1759 on 10 processors,

Problem name	M	n	Speedup	LOQO It	IPPCG	
					Procs	It
sctap12.108	6510	10416	4.1	28	12	39(1159)
sctap12.216	12990	20784	6.0	34	12	44
sctap12.480	28830	46128	10.9	59	12	50
sctap12.480q	28830	46128	16.1	39	12	53
scsd82.108	2170	15190	2.1	20	12	30
scsd82.216	4330	30310	2.1	21	12	35
scsd82.432	8650	60550	2.9	21	12	36
scsd82.432q	8650	60550	3.7	16	12	32
sc205.200	4413	4414	0.4	21	10	34
sc205.800q	17613	17614	2.1	33	10	53
scrs82.128	3612	4901	1.2	21	16	42
scrs82.256	7196	9765	1.9	24	16	46
scrs82.512	14364	19493	2.8	26	16	49

Table II Results via LOQO and IPPCG method for SLP and RO problems on SGI Origin 2000.

1789 on 60 processors and 1707 on 120 processors. The IPPCG method shows a good scalability for a large number of scenarios and for a sufficiently large size of the matrices A_0 , T_i and W_i : for sc205.200, we observe a degradation of the speedup for 40 or 100 processors, because of the small granularity of the computational work on each processor (5 or 2 scenarios respectively for each processor and small size of the blocks of the matrix A in (1)).

In Table II we report the speedup between the IPPCG method and the serial code LOQO (version 5.04 for SGI IRIX 6.5)⁶ on SGI Origin 2000/64. Unlike the relative speedup, the speedup is defined as the ratio of the execution time of the fastest available serial code on a single processor to that of the parallel algorithm on multiple processors. In the column “Procs”, we report the number of processors used for running the parallel code. The column “It” denotes the number of the iterations for both methods. The total number of inner PCG iterations is given in brackets in the column “It”. Regarding to the accuracy, the effectiveness of the IPPCG method is comparable to that of LOQO. The number of outer iterations of the IPPCG algorithm is greater than that of LOQO which uses a

Procs	Relative speedup	
	$n=37800$	$n=144000$
2	1.8	1.7
5	4.4	4.2
10	8.8	8.1
15	13.2	12.0
30	24.2	24.4
50	31.9	40.2

Table III Relative speedup for the water quality problem on Cray T3E; $m = n/2$.

⁶ Potra, F., Roos, C., and Terlaky, T., Editors, *Special Issue on Interior Point Methods*, Optimization Method & Software, Vols. 11-12, pp. 451-484, 1999

predictor-corrector strategy⁷; nevertheless the IPPCG method is competitive with respect to LOQO when the number of the scenarios increases. In the numerical experiments reported in Table 2, the number of processors for the execution of the IPPCG method is generally greater than the minimum number of processors required in order to obtain a better performance of the IPPCG method with respect to LOQO.

Procs	Relative speedup		
	$n=28800$	$n=64800$	$n=115200$
4	3.8	3.8	3.9
6	5.8	5.6	5.8
12	11.9	11.7	11.2
20	19.9	19.6	19.2
30	28.2	29.1	28.6
60	40.6	51.9	54.6

Table IV Relative speedup for the diffusion-convection problem on Cray T3E; $m = n/2$.

Tables III and IV show the relative speedup obtained for problems arising from water quality and diffusion-convection processes: these tables confirm the good scalability of the algorithm; in particular, when the number of the processors increases, the performance of the algorithm improves if large size problems are considered, while the speedup degrades with many processors and a small problem. This is due to the different granularity of the computational work for each processor, while the overhead due to the communication and the synchronization time increases (even if slowly) as the number of processors increases.

We have also observed that for the water quality problem when $n = 37800$ on six processors, IPPCG takes half a time with respect to LOQO, while with $n = 144000$ the time required by IPPCG on six processors is eight times less than that needed by LOQO.

These considerations point out that the IPPCG algorithm is an effective method for large-scale quadratic programs with special structure on parallel computers.

Acknowledgments

We would like to thank Paolo Malfetti (CINECA) for the technical support on SGI Origin 2000 and Giovanni Erbacci (CINECA) for the helpful discussion on parallel implementation.

⁷ Mehrotra, S., *On the Implementation of a Primal-Dual Interior Point Method*, SIAM Journal on Optimization, Vol. 2, pp. 575-601, 1992.

Publications and References

- [1] C. Durazzi, V. Ruggiero, G. Zanghirati, *Journal of Optimization Theory and Applications* 2, 110 (2001).
- [2] C. Durazzi, V. Ruggiero, G. Zanghirati, *Solving a Special Class of Discrete Optimal Control Problems via a Parallel Interior-Point Method*, Proceedings of the Int. Workshop “Equilibrium Problems and Variational Models”.