

A PARALLEL INTERIOR POINT DECOMPOSITION ALGORITHM FOR BLOCK-ANGULAR SEMIDEFINITE PROGRAMS

Kartik K. Sivaramakrishnan

Department of Mathematics
North Carolina State University

kksivara@ncsu.edu

<http://www4.ncsu.edu/~kksivara>

ICCOPT II & MOPTA 07

McMaster University

August 15, 2007

Contents

- **Semidefinite Programming (SDP)**
- **Motivation for our two stage decomposition algorithm**
- **Semidefinite programs with block angular structure**
- **Preprocessing SDPs into equivalent block-angular SDPs**
- **Regularized decomposition algorithm for block-angular SDPs**
- **Results on the distributed Henry2 cluster at NCSU**
- **Comparisons with the OpenMP version of CSDP**
- **Conclusions**

Conic programming

$$\begin{array}{ll} \text{(P)} & \max \quad c^T x \\ & \text{s.t.} \quad Ax = b \\ & \quad \quad x \in \mathcal{K} \end{array}$$

$$\begin{array}{ll} \text{(D)} & \min \quad b^T y \\ & \text{s.t.} \quad A^T y - s = c \\ & \quad \quad s \in \mathcal{K} \end{array}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $\mathcal{K} = \mathcal{K}_1 \times \dots \times \mathcal{K}_r$

- $r = 1, \mathcal{K} = \mathbb{R}_+^n = \{x \in \mathbb{R}^n : x \geq 0\}$ **LP**
Very large LPs ($m, n \leq 1,000,000$) solvable by the simplex method and/or IPMs.
- $\mathcal{K}_i = \mathbb{Q}_+^{n_i} = \{x \in \mathbb{R}^{n_i} : x_1 \geq \|x_{2:n_i}\|\}$ **SOCP**
Large SOCPs ($m, n \leq 100,000$) solvable by IPMs.
- $\mathcal{K}_i = \mathcal{S}_+^{n_i} = \{X \in \mathcal{S}^{n_i} : X \succeq 0\}$ **SDP**
Medium sized SDPs ($m, n \leq 5000$) solvable by IPMs.
(Problems where $n > 5000$ and $m > 10000$ are very difficult to solve!)

Semidefinite programming

$$\begin{array}{ll} \text{(SDP)} & \max C \bullet X \\ & \text{s.t. } \mathcal{A}(X) = b \\ & X \succeq 0 \end{array}$$

$$\begin{array}{ll} \text{(SDD)} & \min b^T y \\ & \text{s.t. } \mathcal{A}^T y - S = C \\ & S \succeq 0 \end{array}$$

• Notation

- $X, S, C \in \mathcal{S}^n, b \in \mathbb{R}^m$
- $A \bullet B = \text{trace}(AB) = \sum_{i,j=1}^n A_{ij}B_{ij}$ (Frobenius inner product)
- The operator $\mathcal{A} : \mathcal{S}^n \rightarrow \mathbb{R}^m$ and its adjoint $\mathcal{A}^T : \mathbb{R}^m \rightarrow \mathcal{S}^n$ are

$$\mathcal{A}(X) = \begin{pmatrix} A_1 \bullet X \\ \vdots \\ A_m \bullet X \end{pmatrix}, \quad \mathcal{A}^T y = \sum_{i=1}^m y_i A_i$$

where $A_i \in \mathcal{S}^n, i = 1, \dots, m$

Motivation

- (a) Exploit the **sparsity** in the underlying SDP and preprocess it into an equivalent SDP having a **block-angular** structure.

- (b) Solve block-angular SDP **iteratively** between a **master problem** (quadratic program) and **decomposed and distributed subproblems** (smaller SDPs) in a parallel computing environment.

- (c) Improve the **scalability** of interior point methods (IPMs) by applying them instead on the smaller master problem, and subproblems (which are solved in parallel!)

This is our **parallel interior point decomposition** algorithm.

Semidefinite programming with block angular structure

$$\begin{aligned} \max \quad & \sum_{i=1}^r C_i \bullet X_i \\ \text{s.t.} \quad & \sum_{i=1}^r \mathcal{A}_i(X_i) = b \\ & X_i \in \mathcal{C}_i, \quad i = 1, \dots, r \end{aligned}$$

• Notes

- $X_i, C_i \in \mathcal{S}^{n_i}, b \in \mathbb{R}^m$.
- $\mathcal{C}_i = \{X_i : \mathcal{B}_i(X_i) = d_i, X_i \succeq 0\}$ - compact semidefinite feasibility sets.
- The objective function and coupling constraints are **block separable**.
- In the absence of coupling constraints, solve r independent problems

$$\max_{X_i \in \mathcal{C}_i} C_i \bullet X_i$$

SDPs with a block-angular structure

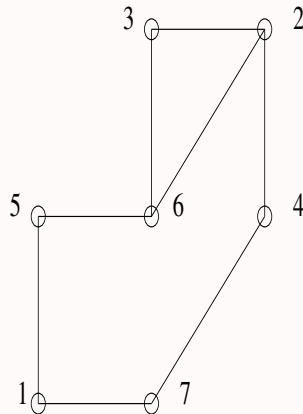
1. **Preprocessed SDPs after matrix completion:**
Fukuda-Kojima-Nakata-Murota (2000)
2. **Control and stability analysis of interconnected subsystems:**
Langbort-D'Andrea-Xiao-Boyd (2003)
3. **Two stage stochastic semidefinite programs with recourse:**
Mehrotra-Ozëvin (2007)
4. **Polynomial optimization problems with sparsity:**
Waki-Kim-Kojima-Muramatsu (2006)
5. **Exploiting group symmetry in SDPs:**
Parrilo-Gatermann (2004)

Exploiting sparsity to get block-angular SDPs

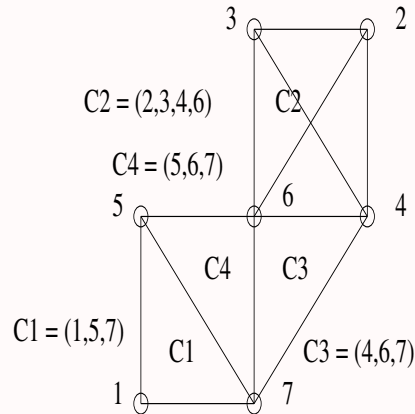
Consider the SDP

$$\begin{aligned} \max \quad & L \bullet X \\ \text{s.t.} \quad & X_{ii} = 1, \quad i = 1, \dots, n, \\ & X \succeq 0, \end{aligned}$$

where L is the adjacency matrix of the graph



GRAPH



CHORDAL EXTENSION OF GRAPH

Exploiting sparsity to get block-angular SDPs

Using **matrix completion**, one can reformulate the earlier SDP as

$$\max \sum_{k=1}^4 (L^k \bullet X^k)$$

$$\text{s.t.} \quad X_{23}^1 - X_{13}^4 = 0,$$

$$X_{34}^2 - X_{12}^3 = 0,$$

$$X_{23}^3 - X_{23}^4 = 0,$$

$$X_{ii}^k = 1, \quad i = 1, \dots, |C_k|, \quad k = 1, \dots, 4,$$

$$X^k \succeq 0, \quad k = 1, \dots, 4,$$

which is in **block-angular** form.

Preprocessing SDPs into block-angular form

1. Consider the SDP

$$\begin{aligned} \max \quad & C \bullet X \\ \text{s.t.} \quad & X_{ij} = B_{ij} \quad (i, j) \in I, \\ & X_{pq} \leq F_{pq}, \quad (p, q) \in J, \\ & A_k \bullet X = b_k, \quad k = 1, \dots, m_{\text{org}}, \\ & X \succeq 0. \end{aligned}$$

2. I and J are disjoint sets that include $D = \{(1, 1), \dots, (n, n)\}$.

3. Examples include SDP relaxations of maxcut, stable set, and box QPs.

4. Such SDPs can be processed into an equivalent SDP in block-angular form using our preprocessing scheme.

5. Construct aggregate sparsity graph $G = (V, E)$ for SDP, where $V = \{1, 2, \dots, n\}$ and

$$E = \{(i, j) \in V \times V : C_{ij} \neq 0 \text{ or } \exists k \in \{1, \dots, m_{\text{org}}\} (A_k)_{ij} \neq 0\}$$

Preprocessing SDPs into block-angular form

1. Construct minimal chordal extension $G' = (V, E')$ and its graph $A(E')$ from $G = (V, E)$. Find maximal cliques $Cl_i, i = 1, \dots, r$ in G' .
2. Construct block-angular SDP from clique information as follows:
 - (a) For each nonzero element $(i, j) \in A(E')$, let $\hat{r}(i, j) = \min\{s : (i, j) \in Cl_s \times Cl_s\}$ and let i and j be elements p and q in $Cl_{\hat{r}(i, j)}$. Set $C_{pq}^{\hat{r}(i, j)} = C_{ij}$ and $[A_k]_{pq}^{\hat{r}(i, j)} = [A_k]_{ij}, k = 1, \dots, m_{\text{org}}$.
 - (b) Set $EC_{ij} = 1, i, j = 1, \dots, n$. Repeat (b) for all pairs of cliques: Suppose Cl_k and Cl_l share an element (i, j) ; and i and j are elements p and q in Cl_k ; and elements s and t in Cl_l .
 - If $(i, j) \in I$, add $X_{pq}^k = B_{ij}$ and $X_{st}^l = B_{ij}$ to new SDP. Update $EC_{ij} = EC_{ij} + 1$.
 - If $(i, j) \in J$, add $X_{pq}^k - X_{st}^l = 0$ and $X_{pq}^k \leq E_{ij}$ and $X_{st}^l \leq E_{ij}$ to new SDP. Update $EC_{ij} = EC_{ij} + 1$.
 - If $(i, j) \notin I \cup J$, add $X_{pq}^k - X_{st}^l = 0$ to new SDP.
 - (c) Set $C_{pq}^k = \frac{C_{ij}}{EC_{ij}}, p, q = 1, \dots, |Cl_k|$, where p and q in Cl_k are nodes i and j in $G' = (V, E')$.

The Lagrangian dual problem

- The Lagrangian dual problem is

$$\min_y \theta(y) = b^T y + \sum_{i=1}^r \theta_i(y)$$

where

$$\theta_i(y) = \max_{X_i \in \mathcal{C}_i} (C_i - \mathcal{A}_i^T y) \bullet X_i$$

- Dual is an unconstrained **convex** but **nonsmooth** problem.
- Given y^k , we have $\theta(y^k) = b^T y^k + \sum_{i=1}^r (C_i - \mathcal{A}_i^T y^k) \bullet X_i^k$

and a subgradient $g(y^k) = (b - \sum_{i=1}^r \mathcal{A}_i(X_i^k))$ where

$$X_i^k = \operatorname{argmax}_{X_i \in \mathcal{C}_i} (C_i - \mathcal{A}_i^T y^k) \bullet X_i$$

(these can be computed in parallel!)

Solving the Lagrangian dual

1. Construct a model $\theta^k(y)$ an **underestimate** for $\theta(y)$

$$\theta^k(y) = b^T y + \sum_{i=1}^r \max_{j=1, \dots, J^k(i)} (C_i - \mathcal{A}_i^T y) \bullet X_i^j$$

from the function values and subgradient information.

2. The **regularized** master problem then is

$$\min_y \theta^k(y) + \frac{u^k}{2} \|y - x^k\|^2$$

where $u^k \geq 0$ and x^k is our current center (best iterate so far!)

3. This can be formulated as the following quadratic program

$$\begin{aligned} \min \quad & b^T y + \sum_{i=1}^r z_i + \frac{u^k}{2} \|y - x^k\|^2 \\ \text{s.t.} \quad & \mathcal{A}_i (X_i^j)^T y + z_i \geq C_i \bullet X_i^j, \quad i = 1, \dots, r, j \in J^k(i). \end{aligned}$$

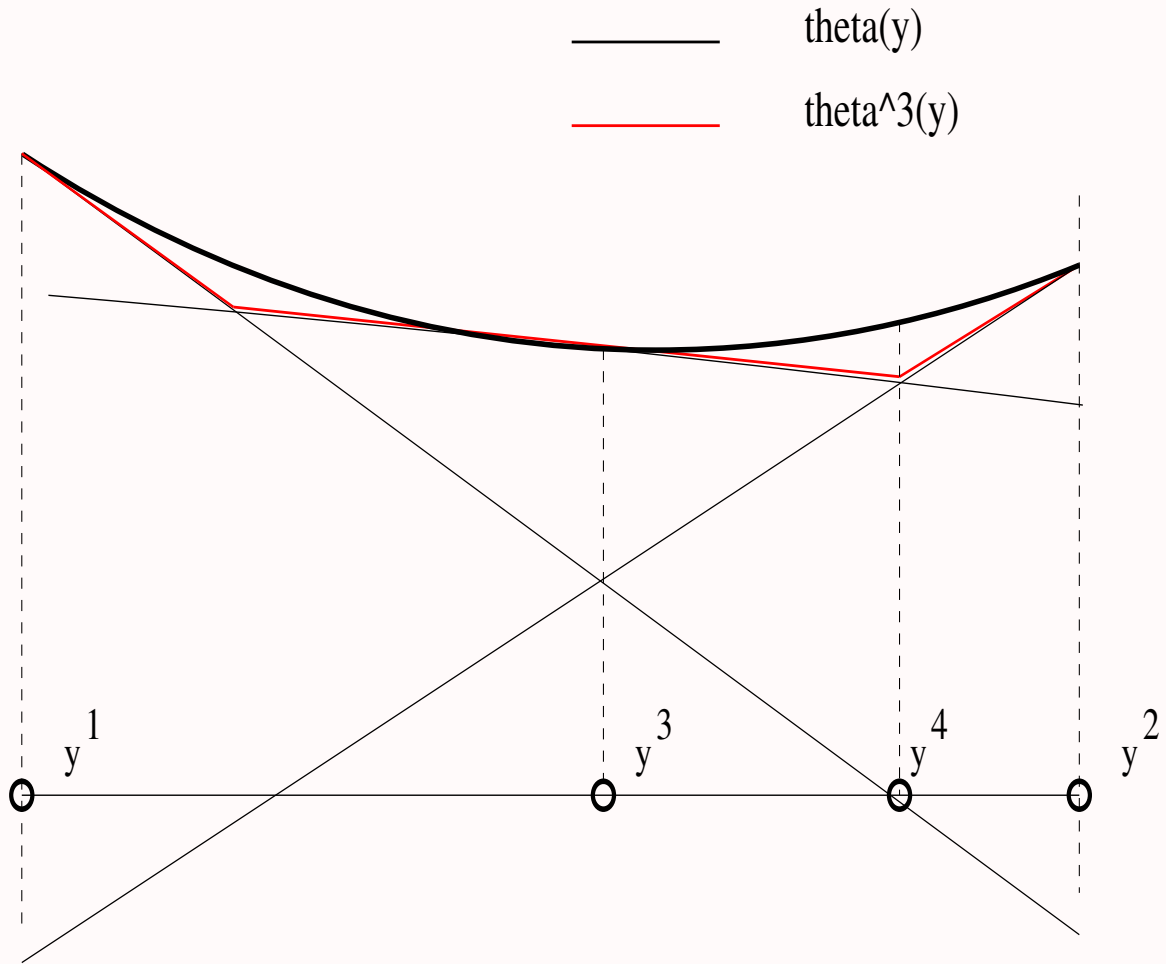


Figure 1: Solving the Lagrangian dual problem

Decomposition algorithm for block-angular SDPs

1. **Initialize:** Set $k = 1$, $J^0(i) = \emptyset$, $z_i^0 = -\infty$, $i = 1, \dots, r$. Choose $m_L \in (0, 1)$, starting point $y^0 \in \mathbb{R}^m$, and set $x^0 = y^0$.

2. **Solve subproblems in parallel:** X_i^k is solution to i th subproblem.

3. **Update master problem:** If $\theta_i(y^k) > z_i^k$, add constraint

$$\mathcal{A}_i(X_i^k)^T y + z_i \geq C_i \bullet X_i^k$$

to master problem and set $J^k(i) = J^{k-1}(i) \cup \{k\}$. Else, $J^k(i) = J^{k-1}(i)$.

4. **Update center x^k and weight u^k :** If $k = 1$ or if

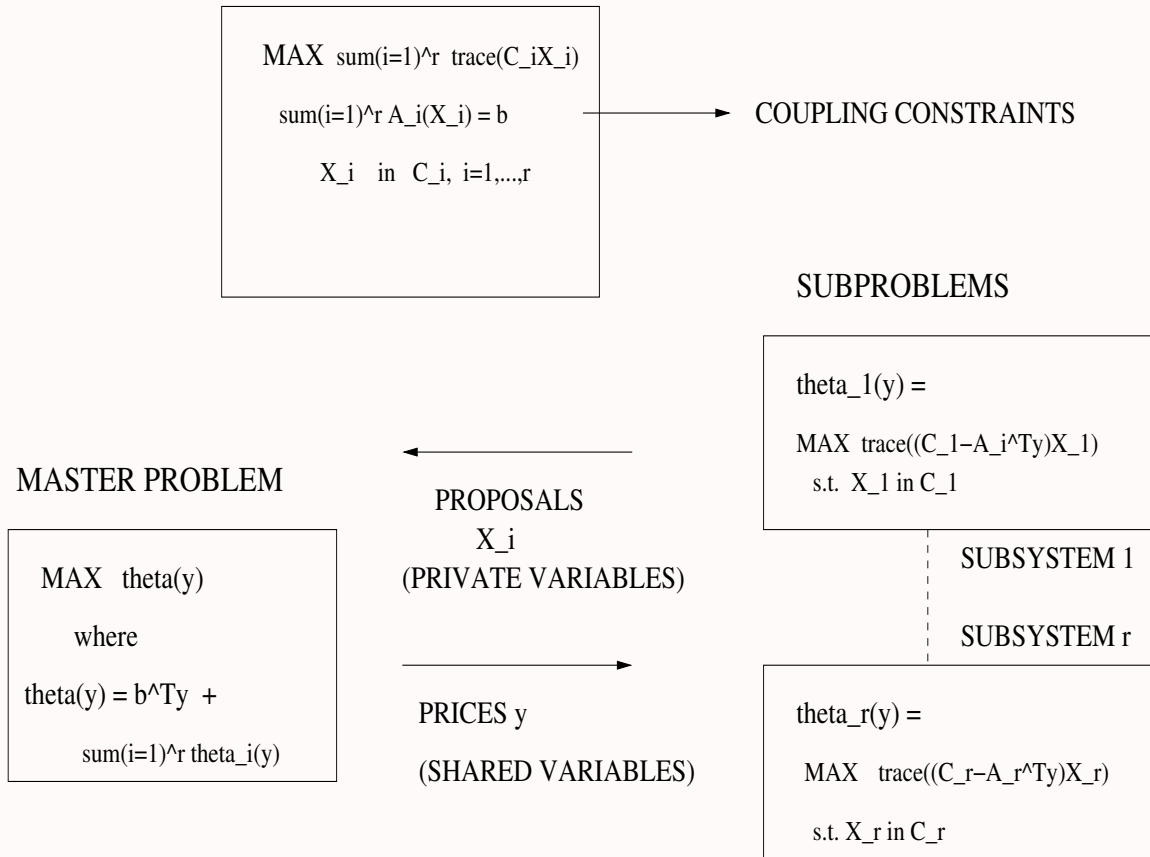
$$\theta(y^k) \leq (1 - m_L)\theta(x^{k-1}) + m_L\theta^{k-1}(y^k)$$

then set $x^k = y^k$ (serious step). Else, set $x^k = x^{k-1}$ (null step).

5. **Solve the master problem:** Let (y^{k+1}, z^{k+1}) and λ^k be primal and dual solutions. Let $\theta^k(y^{k+1}) = b^T y^{k+1} + \sum_{i=1}^r z_i^{k+1}$.

6. **Termination Check:** If $\theta^{k+1}(y^{k+1}) = \theta(x^k)$ stop!

7. **Aggregation:** Aggregate constraints in master problem with small λ_i^k . Set $k = k + 1$ and return to Step 2.



C_i are convex sets defined by LMIs

Figure 2: Decomposition by prices

Computational Results: 1

- MPI code is run on IBM Blade Center Linux Cluster (Henry2) at NC State: 175 nodes, each node is a dual 2.8-3.2GHz processor with 4GB of memory.
- Our code is in C and uses MPI for communication between processors.
- CPLEX 9.0 solves the master problem and serial CSDP 6.0 solves the subproblems.
- Accuracy: 3 digits on smaller SDPs and 2 digits on larger SDPs.
- Used procedure in [Helmberg-Kiwiel](#) to adjust weight u .
- Employed aggregation scheme of [Kiwiel](#) to limit size of master problem.
- Smaller SDPs are from SDPLIB while larger SDPs are generated from graphs in Gset library.
- OpenMP version of CSDP is run on IBM Power5 shared memory system at NC State. Each node has eight 1.9GHz processors and 32GB of shared memory.

Prob	n	m	n_p	m_p	n_{total}
maxG11	800	800	216 (4)	1208 (360)	848
qpG11	1600	800	180 (61)	3808 (2528)	2560
maxG32	2000	2000	286 (22)	8690 (6318)	2500
thetaG11	801	2401	194 (23)	17036 (13867)	1019
maxG48	3000	3000	542 (64)	24580 (20650)	3930
qpG32	4000	2000	432 (97)	16402 (13313)	6178
maxG57	5000	5000	405 (516)	61595 (51883)	9712
thetaG32	2001	6001	212 (80)	22774 (16762)	3187
thetaG48	3001	9001	441 (102)	39468 (30632)	4586
thetaG57	5001	15001	434 (182)	75369 (60711)	9313
maxG62	7000	7000	515 (692)	100873 (87232)	13641
maxG65	8000	8000	656 (791)	101300 (85961)	15339
maxG66	9000	9000	633 (898)	134904 (117344)	17560
maxG67	10000	10000	903 (942)	131593 (112532)	19061
maxG77	14000	14000	916 (1395)	219118 (191591)	27527
thetaG62	7001	21001	506 (540)	118828 (98180)	13496
thetaG65	8001	24001	687 (555)	134199 (110726)	14974
thetaG67	10001	30001	853 (694)	201570 (172268)	19028
thetaG77	14001	42001	782 (933)	263318 (222229)	26315

Table 1: Test Instances

Computational Results on small instances

Prob	Opt val	p	UB	Time (h:m:s)	rel_acc	Serious steps	Null steps	Master (h:m:s)	Subproblem (h:m:s)
maxG11	629.16	4	629.54	40	6.0e-4	9	21	1	39
qpG11	2448.66	9	2450.36	1:06	6.9e-4	18	39	8	58
maxG32	1567.64	12	1568.47	10:00	5.3e-4	19	138	1:32	8:24
thetaG11	400.00	4	400.24	1:25	5.9e-4	14	14	2	1:23
maxG48	4500.00	16	4500.00	12:32	0.0	1	37	52	11:40
qpG32	6226.55	16	6230.99	31:01	7.1e-4	27	133	3:13	27:47

Comparison with serial version of CSDP

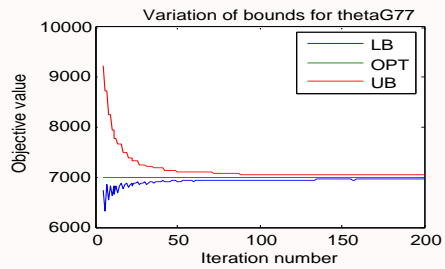
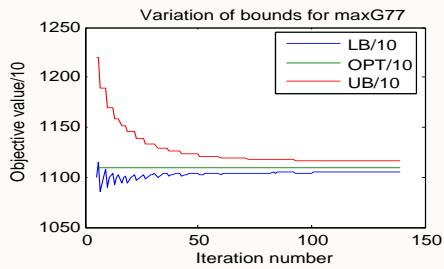
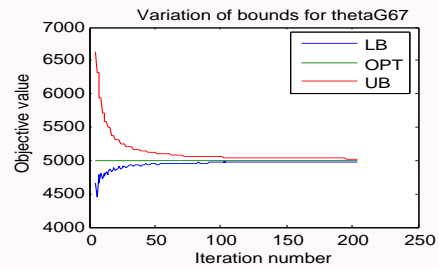
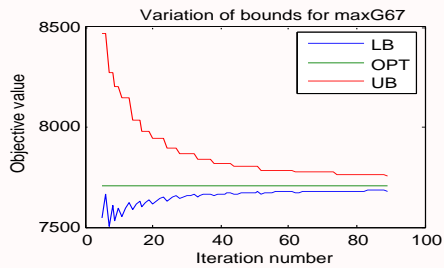
Prob	Best time with MPI code on preprocessed SDPs – Table 2			Serial CSDP on original SDP		
	rel_acc	Time (Iter) (h:m:s)	Memory (GB)	dual_gap	Time (Iter) (h:m:s)	Memory (GB)
maxG11	6.0e-4	40 (30)	0.03	4.8e-7	14 (14)	0.06
qpG11	6.9e-4	1:06 (57)	0.07	8.7e-7	1:57 (16)	0.22
maxG32	5.3e-4	10:00 (157)	0.17	6.5e-7	3:54 (15)	0.37
thetaG11	5.9e-4	1:25 (28)	0.04	3.9e-7	55 (21)	0.1
maxG48	0.0	12:32 (38)	0.32	6.5e-7	9:47 (13)	0.83
qpG32	7.1e-4	31:01 (160)	0.28	7.9e-7	26:19 (16)	1.38

Computational Results on large instances

Prob	Opt val	p	UB	Time (h:m:s)	rel_acc	Serious steps	Null steps	Master (h:m:s)	Subproblem (h:m:s)
maxG57	3875.98	16	3891.01	31:26	3.8e-3	17	123	13:12	18:07
thetaG32	1000.00	16	1004.00	7:25	3.9e-3	17	118	1:40	5:45
thetaG48	1500.00	16	1507.49	35:34	5.0e-3	17	102	3:02	32:32
thetaG57	2500.00	16	2511.12	1:18:59	4.4e-3	20	177	29:41	49:34
maxG62	5410.91	16	5436.64	1:07:23	4.7e-3	16	128	29:40	37:31
maxG65	6185.04	16	6214.72	1:25:42	4.8e-3	15	81	25:08	1:00:25
maxG66	7097.21	16	7130.36	1:43:33	4.6e-3	16	107	44:14	59:09
maxG67	7708.93	16	7756.14	2:50:00	6.1e-3	17	72	1:01:42	1:48:06
maxG77	11097.67	16	11160.06	3:49:12	5.6e-3	17	83	1:27:39	2:21:02
thetaG62	3500.00	16	3517.34	2:18:16	5.0e-3	21	157	1:09:57	1:08:13
thetaG65	4000.00	16	4018.71	4:18:02	4.7e-3	19	181	46:16	3:31:40
thetaG67	5000.00	16	5027.52	7:57:53	5.5e-3	22	182	1:27:39	6:30:01
thetaG77	7000.00	16	7038.77	7:22:06	5.5e-3	20	180	2:20:25	5:01:23

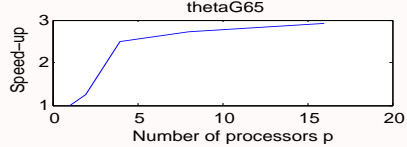
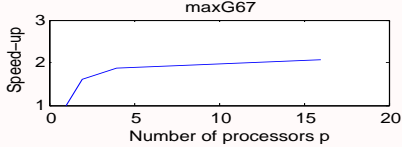
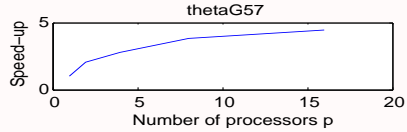
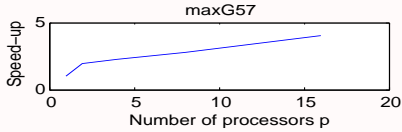
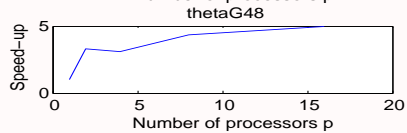
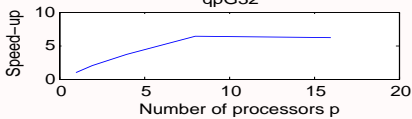
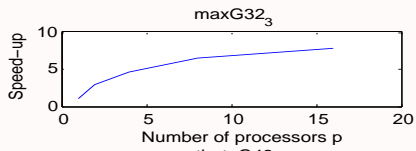
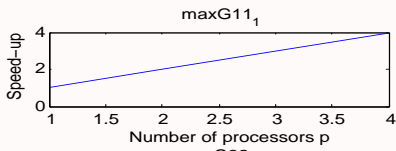
Comparison with OpenMP version of CSDP

Prob	Our MPI code on preprocessed SDP – Henry2 distributed memory system (p=16)			OpenMP CSDP on original SDP – Power5 shared memory system (p=1,4,8)				
	rel_acc	Time (Iter) (h:m:s)	Memory (GB)	dual_gap	Time (Iter) (h:m:s)			Memory (GB)
					p=1	p=4	p=8	
thetaG32	3.9e-3	7:25 (135)	0.20	4e-7	12:01 (23)	4:07	2:43	0.63
thetaG48	5e-3	35:34 (119)	0.32	5e-7	39:42 (24)	12:51	8:09	1.39
maxG57	3.9e-3	31:26 (140)	0.55	8.5e-7	1:00:00 (16)	21:41	13:55	2.31
thetaG57	4e-3	1:18:59 (197)	0.64	6.4e-7	2:44:10 (23)	52:37	36:01	3.84
maxG62	4.9e-3	1:07:23 (144)	1.07	6.3e-7	2:32:10 (16)	52:48	36:01	4.51
maxG65	5e-3	1:25:42 (96)	1.05	3e-7	3:58:30 (18)	1:18:51	51:11	5.88
maxG66	4.7e-3	1:43:33 (123)	1.63	4e-7	5:27:01 (17)	1:51:10	1:12:39	7.43
thetaG62	5e-3	2:18:16 (178)	1.14	7.6e-7	6:33:02 (21)	2:05:16	1:20:50	7.51
maxG67	6.2e-3	2:50:00 (89)	1.59	2.5e-7	7:45:47 (18)	2:34:42	1:42:05	9.18
thetaG65	4.7e-3	4:18:02 (200)	1.23	1.2e-7	10:39:10 (23)	3:23:05	2:08:55	9.81
thetaG67	5.5e-3	7:57:53 (204)	1.97	1.6e-7	22:32:58 (26)	6:57:47	4:23:10	15.3
maxG77	5.7e-3	3:49:12 (100)	3.10	7.3e-7	20:01:32 (17)	Failed	4:02:15	17.97
thetaG77	5.5e-3	7:22:06 (200)	2.91	-	MM	MM	MM	31.2



Scalability of algorithm on test problems on up to 16 processors

Prob	p=1	p=2	p=4	p=8	p=16
maxG11.1	2:42 (30) 2:39 (21)	1:19 (30) 78 (21)	40 (30) 39 (21)	- -	- -
maxG32.3	2:11:16 (300) 1:45:28 (277)	45:03 (300) 34:31 (276)	28:56 (300) 18:13 (276)	20:11 (300) 11:33 (276)	16:53 (282) 8:57 (268)
qpG32	3:07:24 (144) 3:03:59 (118)	1:30:57 (141) 1:28:15 (115)	50:26 (138) 47:36 (113)	29:44 (145) 26:55 (120)	31:01 (160) 27:47 (133)
thetaG48	2:57:40 (126) 2:51:46 (108)	53:57 (116) 51:43 (99)	58:24 (143) 55:08 (125)	40:43 (119) 37:56 (102)	35:34 (119) 32:32 (102)
maxG57	2:06:03 (142) 1:51:11 (126)	1:06:31 (137) 53:40 (121)	55:35 (154) 39:24 (137)	46:18 (121) 30:43 (105)	31:26 (140) 18:07 (123)
thetaG57	5:51:20 (172) 5:06:00 (153)	2:53:03 (174) 2:20:55 (154)	2:08:29 (198) 1:25:57 (178)	1:32:14 (188) 1:09:56 (169)	1:18:59 (197) 49:34 (177)
maxG67	6:00:39 (90) 4:44:56 (73)	3:45:51 (90) 2:38:02 (74)	3:12:37 (94) 2:05:50 (77)	3:06:02 (93) 1:59:30 (76)	2:50:00 (89) 1:48:06 (72)
thetaG65	12:31:12 (200) 11:39:13 (181)	9:58:16 (200) 9:01:11 (181)	5:01:39 (200) 4:15:31 (181)	4:37:07 (200) 3:51:20 (182)	4:18:02 (200) 3:31:40 (181)



Conclusions and Future work

- The decomposition algorithm is much faster and requires considerably less memory storage than the serial version of CSDP on selected SDP problems.
- Our solution times are very competitive with the OpenMP version of CSDP on selected large SDP instances.
- Algorithm attains good parallel scalability as one increases the number of processors.
- We demonstrate that the matrix completion scheme can be modified and used in the solution of much larger SDPs than was previously possible.
- We are developing a hybrid (MPI-OpenMP) version of the decomposition algorithm, where each subproblem is solved in parallel with the OpenMP version of CSDP.
- Successfully warm-starting the subproblems.

Thank you for your attention!
Questions, Comments, Suggestions ?

The slides from this talk are available online at
[http://www4.ncsu.edu/~kksivara/
publications/kartik-iccopt2.pdf](http://www4.ncsu.edu/~kksivara/publications/kartik-iccopt2.pdf)

A technical report appears at
[http://www4.ncsu.edu/~kksivara/
publications/parallel-conic-blockangular.pdf](http://www4.ncsu.edu/~kksivara/publications/parallel-conic-blockangular.pdf)

Bibliography: 1

1. K. SIVARAMAKRISHNAN, *A parallel interior point decomposition algorithm for block angular semidefinite programs*, submitted. Available at <http://www4.ncsu.edu/~kksivara/publications/parallel-conic-blockangular.pdf>
2. M. FUKUDA, M. KOJIMA, K. MUROTA, AND K. NAKATA, *Exploiting sparsity in semidefinite programming via matrix completion I: General framework*, SIAM Journal on Optimization, 11(2000), pp. 647-674.
3. K. FUJISAWA, M. FUKUDA, AND K. NAKATA, *Preprocessing sparse semidefinite programs via matrix completion*, Optimization Methods and Software, 21(2006), pp. 17-39.
4. C. LANGBORT, R. D'ANDREA, L. XIAO, AND S. BOYD, *A decomposition approach to distributed analysis of networked systems*, Proceedings of the 43rd IEEE Conference on Decision and Control, pp. 3980-3985.
5. K. GATERMANN AND P.A. PARRILO, *Symmetry groups, semidefinite programs, and sums of squares*, Journal of Pure and Applied Algebra 192(2004), pp. 95-128.
6. H. WAKI, S. KIM, M. KOJIMA, AND M. MURAMATSU, *Sums of squares and semidefinite programming relaxations for polynomial problems with structured sparsity*, SIAM Journal on Optimization, 17(2006), pp. 218-242.
7. S. MEHROTRA AND M.G. ÖZEVİN, *Decomposition based interior point methods for two-stage stochastic semidefinite programming*, SIAM Journal on Optimization, 18(2007), pp. 206-222.

Bibliography: 2

8. **K.C. KIWIEL**, *Proximity control in bundle methods for convex nondifferentiable optimization*, Mathematical Programming, 46(1990), pp. 105-122.
9. **K.C. KIWIEL**, *An aggregate subgradient method for nonsmooth convex minimization*, Mathematical Programming, 27(1983), pp. 320-341.
10. **B. BORCHERS**, *CSDP, a C library for semidefinite programming*, Optimization Methods and Software, 11 & 12(1999), pp. 613-623. <http://infohost.nmt.edu/~borchers/csdp.html>.
11. **C. HELMBERG AND K.C. KIWIEL**, *A spectral bundle method with bounds*, Mathematical Programming, 93(2002), pp. 173-194.
12. **B. BORCHERS**, *SDPLIB 1.2, A library of semidefinite programming test problems*, Optimization Methods and Software, 11, 1999, pp. 683-690. <http://infohost.nmt.edu/~sdplib/>
13. **B. BORCHERS AND J. YOUNG**, *Implementation of a primal-dual method for SDP on a shared memory parallel architecture*, Computational Optimization and Applications, 37(2007), pp. 355-369.
14. **A. RUSZCZYŃSKI**, *Nonlinear Optimization*, Princeton University Press, 2006.
15. **P.S. PACHECO**, *Parallel Programming with MPI*, Morgan Kaufmann, 1997.
16. **GSET LIBRARY**, Available at <http://www.stanford.edu/~yyye/yyye/Gset/>.