

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

Kartik K. Sivaramakrishnan

Department of Mathematics
North Carolina State University

kksivara@ncsu.edu

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

SIAM Conference on Optimization (OP08)

Boston

May 13, 2008

Contents

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

Semidefinite programming

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

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

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

- The matrices X and S are required to be positive semidefinite.
- Problems where $m > 10,000$ and $n > 5000$ are difficult to solve.

Polynomial optimization

1. Consider the POP

$$\begin{aligned} \min \quad & p(x) \\ \text{s.t.} \quad & g_i(x) \geq 0, \quad i = 1, \dots, m \end{aligned}$$

whose feasible region S is contained in the unit hypercube. Let $2d_0$ or $2d_0 - 1$ denote degree of $p(x) = \sum_{\alpha} p_{\alpha} x^{\alpha}$ where $\alpha \in \mathcal{Z}_+^n$, and $2d_i$ or $2d_i - 1$ denote degrees of $g_i(x) = \sum_{\alpha} (g_i)_{\alpha} x^{\alpha}$.

2. μ is a probability measure, $y_{\alpha} = \int x^{\alpha} d\mu$ is its sequence of moments, and $S^r = \{\alpha \in \mathcal{Z}_+^n : \sum_i \alpha_i \leq r\}$.

3. Let $\alpha, \beta \in S^r$. $M_r(y)(\alpha, \beta) = y_{\alpha+\beta}$ is the truncated moment matrix, and $M_r(g * y)(\alpha, \beta) = \sum_{\gamma} g_{\gamma} y_{\alpha+\beta+\gamma}$ is a localizing matrix. The dimension $M_r(y)$ is $\binom{n+r}{r}$ and it contains $\binom{n+2r}{2r}$ distinct variables.

4. Let $r \geq \max_{i=0, \dots, m} d_i$. The r th Lasserre relaxation for POP is

$$\begin{aligned} \min \quad & \sum_{\alpha} p_{\alpha} y_{\alpha} \\ \text{s.t.} \quad & M_r(y) \succeq 0 \\ & M_{r-d_j}(g_j * y) \succeq 0, \quad j = 1, \dots, m. \end{aligned}$$

Motivation

- (a) Exploit **sparsity** in underlying SDP and preprocess it into an equivalent SDP having a **block-angular** structure.
- (b) Exploit **sparsity** in underlying POP to get an SDP relaxation having a **block-angular** structure.
- (c) Solve block-angular SDP **iteratively** between a **master problem** (quadratic program) and **decomposed and distributed subproblems** (smaller SDPs) in a parallel computing environment.
- (d) 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.

Primal SDPs 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

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

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

Dual SDPs with block angular structure

$$\begin{aligned} \max \quad & \sum_{i=1}^r c_i^T y_i \\ \text{s.t.} \quad & \sum_{i=1}^r A_i y_i = b \\ & y_i \in \mathcal{C}_i, \quad i = 1, \dots, r \end{aligned}$$

• Notes

- $y_i, c_i \in \mathbb{R}^{n_i}$, $b \in \mathbb{R}^m$, and $A_i \in \mathbb{R}^{m \times n_i}$.
- $\mathcal{C}_i = \{y_i : D_i - \mathcal{B}_i^T y_i \succeq 0\}$ - compact semidefinite feasibility sets where $D_i \in \mathcal{S}^{m_i}$ and $\mathcal{B}_i^T : \mathbb{R}^{n_i} \rightarrow \mathcal{S}^{m_i}$.
- The objective function and coupling constraints are **block separable**.
- In the absence of coupling constraints, solve r independent problems

$$\max_{y_i \in \mathcal{C}_i} c_i^T y_i$$

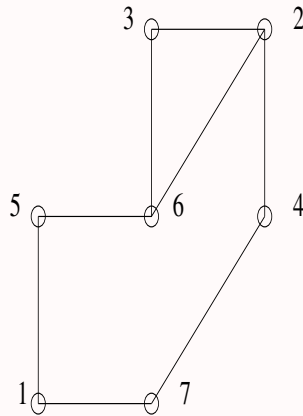
-7-

Exploiting sparsity to get block-angular SDPs

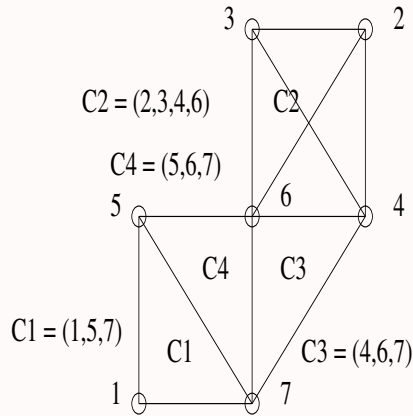
Consider the SDP

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

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 primal **block-angular** form.

Preprocessing algorithm - Phase I

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 $G' = (V, E')$.

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 primal block-angular form using our preprocessing scheme (Sivaramakrishnan 2007).

Generating block-angular SDP relaxations for POPs

1. Consider following POP whose feasible region is contained in the unit cube

$$\begin{array}{ll} \min & x_1x_2 + x_1x_3 - x_1 - 0.5x_2 - 0.5x_3 \\ \text{s.t.} & (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.25 \\ & (x_1 - 0.5)^2 + (x_3 - 0.5)^2 \leq 0.25. \end{array}$$

2. Set F_k contain variables that appear in the k th monomial of $p(x)$.
3. Sets H_j contain all variables appearing in constraint polynomial $g_j(x)$.
4. Construct CSP graph from POP, where $V = \{1, 2, \dots, n\}$ and

$$E = \{(i, j) : i, j \in F_k \text{ for some } k \text{ or } i, j \in H_l \text{ for some } l\}$$

5. Let $I_i, i = 1, \dots, r$ be a set of maximal cliques of CSP graph.
(find chordal extension of graph and then maximal cliques in extension)
In our example, $I_1 = \{1, 2\}$ and $I_2 = \{1, 3\}$.
6. Let $J_i, i = 1, \dots, r$ be an index set of $g_j(x)$ whose list of variables is entirely in I_i . In our example, $J_1 = \{1\}$ and $J_2 = \{2\}$.

Generating block-angular SDP relaxations for POPs

The first SDP relaxation in the Waki et al. (2006) hierarchy is

$$\begin{aligned}
 \min \quad & y_4^1 + y_4^2 - 0.5y_1^1 - 0.5y_1^2 - 0.5y_2^1 - 0.5y_2^2 \\
 \text{s.t.} \quad & \begin{pmatrix} 1 & y_1^1 & y_2^1 \\ y_1^1 & y_3^1 & y_4^1 \\ y_2^1 & y_4^1 & y_5^1 \end{pmatrix} \succeq 0 \\
 & \begin{pmatrix} 1 & y_1^2 & y_2^2 \\ y_1^2 & y_3^2 & y_4^2 \\ y_2^2 & y_4^2 & y_5^2 \end{pmatrix} \succeq 0 \\
 & y_1^1 + y_2^1 - y_3^1 - y_5^1 \geq 0.25 \\
 & y_1^2 + y_2^2 - y_3^2 - y_5^2 \geq 0.25 \\
 & y_1^1 - y_1^2 = 0 \\
 & y_3^1 - y_3^2 = 0 \\
 & y_i^j \geq 0, \quad i = 1, \dots, 5, \quad j = 1, \dots, 2 \\
 & y_i^j \leq 1, \quad i = 1, \dots, 5, \quad j = 1, \dots, 2.
 \end{aligned}$$

which is in the dual block-angular form (see Sivaramakrishnan 2008).

Generating block-angular SDP relaxations for POPs

1. Let I_i , $i = 1, \dots, r$ be maximal cliques in the CSP graph and \mathcal{I}_i is the set of all subsets in I_i .
2. $S_{\mathcal{I}}^r = \{\alpha \in \mathcal{Z}_+^n : \text{supp}(\alpha) \in \mathcal{I}, \sum_i \alpha_i \leq r\}$.
3. $M_r(y, \mathcal{I})$ is a sparse truncated moment matrix and $M_r(g * y, \mathcal{I})$ is a sparse localizing matrix with rows indexed by $\alpha \in S_{\mathcal{I}}^r$.
4. Our contribution (Sivaramakrishnan 2008) is to write the r th sparse SDP relaxation in the Waki et al. (2006) hierarchy as

$$\begin{array}{ll}
 \min & \sum_{i=1}^r f_i^T y_i \\
 \text{s.t.} & M_r(y_i, \mathcal{I}_i) \succeq 0, \quad i = 1, \dots, r \\
 & M_{r-r_{ij}}(g_{ij} * y_i, \mathcal{I}_i) \succeq 0, \quad i = 1, \dots, r, \quad j \in J_i \\
 & \sum_{i=1}^r G_i y_i = 0 \\
 & y^i \succeq 0, \quad i = 1, \dots, r \\
 & y^i \preceq e, \quad i = 1, \dots, r.
 \end{array}$$

which is in the dual block-angular format.

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

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

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

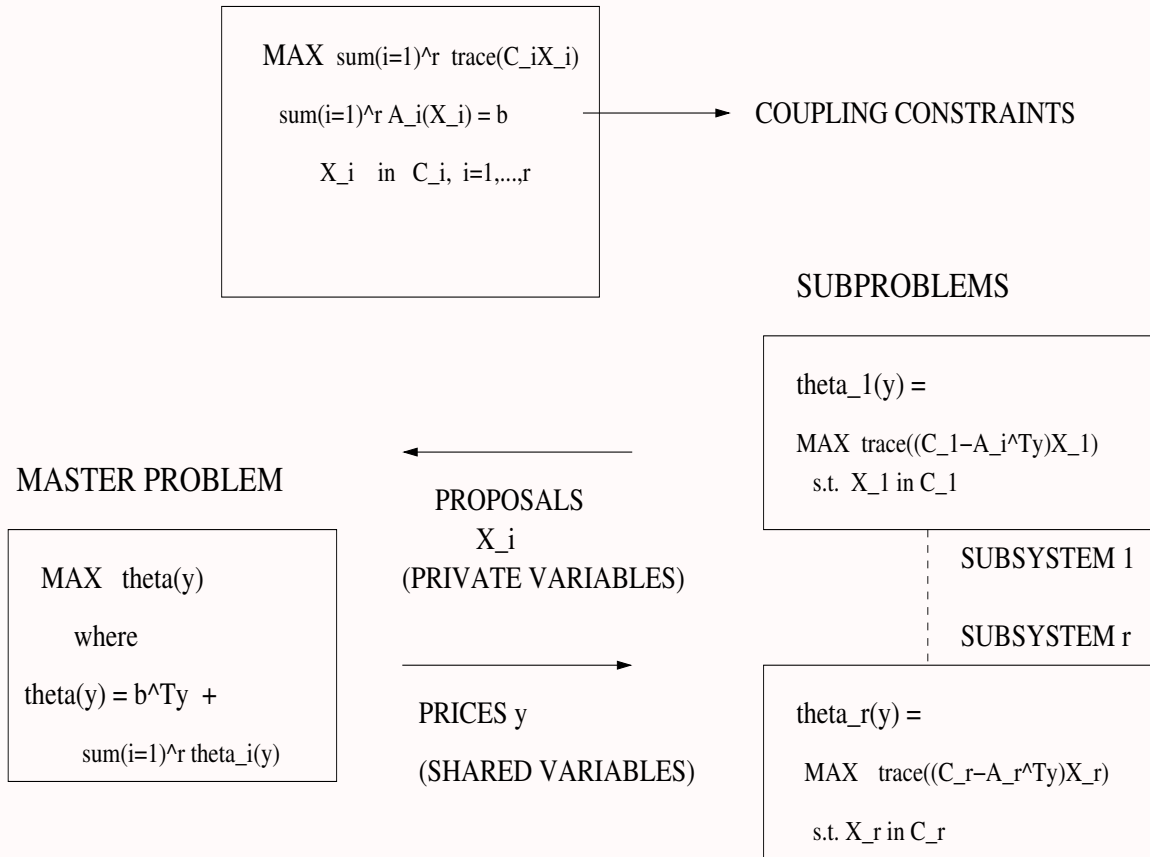
$$\theta(y^k) \leq (1 - m_L)\theta(x^{k-1}) + m_L\theta^{k-1}(y^k)$$

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 1: Decomposition by prices

Setup for Computational Experiments

- 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 (2GB per processor).
- Our code is in C and uses MPI for communication between processors.
- CPLEX 10.0 solves master problem and CSDP 6.0 solves subproblems.
- Accuracy: $\epsilon = \frac{UB-LB}{1+|UB|} < 10^{-2}$.
- Used procedure in [Helmberg-Kiwiel](#) to adjust weight u .
- Employed aggregation scheme of [Kiwiel](#) to limit size of master problem.
- SDPs are generated from graphs in Gset library. We exploit the sparsity in SDPs to get block-angular SDPs in Phase I of our algorithm.
- OpenMP version of CSDP is run on IBM Power5 shared memory system at NC State with eight 1.9GHz processors and 32GB of shared memory.
- See Sivaramakrishnan (2007) for more details.

Test Instances

Prob	n	m	n_p	m_p	n_{total}
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

Computational Results on 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	3903.62	17:31	7.1e-3	16	46	10:23	7:06
thetaG32	1000.00	16	1004.93	3:58	4.9e-3	19	48	1:21	2:36
thetaG48	1500.00	16	1505.49	25:17	3.6e-3	18	70	3:42	21:33
thetaG57	2500.00	16	2509.78	44:00	3.9e-3	21	83	15:35	28:22
maxG62	5410.91	16	5452.37	48:40	7.6e-3	17	57	31:58	16:32
maxG65	6185.04	16	6227.18	42:58	6.8e-3	16	41	17:26	25:24
maxG66	7097.21	16	7144.64	1:15:39	6.6e-3	17	54	45:28	30:01
maxG67	7708.93	8	7764.16	1:31:25	7.1e-3	16	42	28:32	1:01:30
maxG77	11097.67	16	11187.50	2:32:35	8.0e-3	18	52	1:18:34	1:13:41
thetaG62	3500.00	16	3511.57	1:46:40	3.3e-3	21	104	54:34	51:58
thetaG65	4000.00	16	4014.76	3:25:14	3.6e-3	20	95	48:35	2:36:31
thetaG67	5000.00	16	5024.48	6:19:05	4.8e-3	22	118	2:41:26	3:37:25
thetaG77	7000.00	16	7028.62	6:29:00	4.0e-3	22	119	3:42:30	2:46:05

Comparison with OpenMP version of CSDP

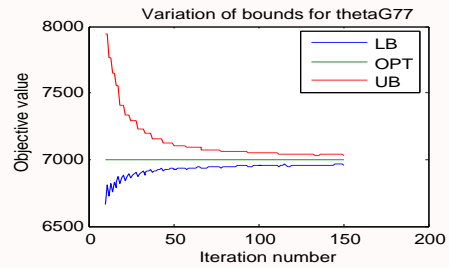
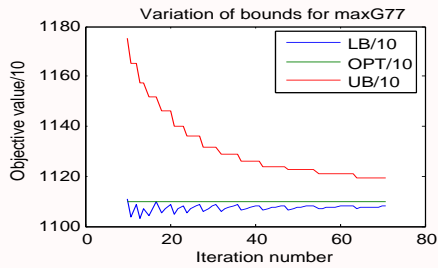
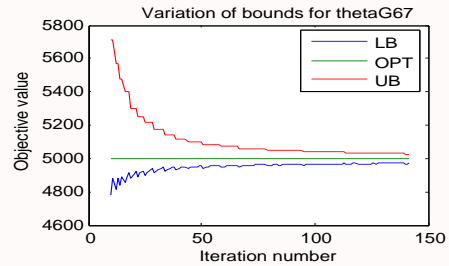
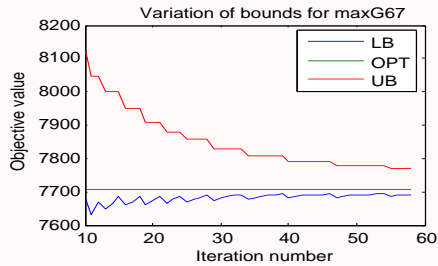
Prob	rel_acc	Time (h:m:s)					Iter	Memory (GB)
		p=1	p=2	p=4	p=8	p=16		
thetaG32	5.0e-3	19:44	14:26	8:40	5:35	3:58	68	0.19
thetaG48	3.6e-3	56:39	54:25	48:04	39:45	25:17	89	0.32
maxG57	7.1e-3	42:36	37:36	32:18	29:01	17:31	63	0.51
thetaG57	3.9e-3	2:42:35	1:29:30	1:18:56	47:07	44:00	103	0.52
maxG62	7.7e-3	1:48:54	1:39:15	1:21:34	1:02:14	48:40	75	0.78
maxG65	6.8e-3	2:54:56	1:39:26	1:25:17	1:06:02	42:58	58	0.70
maxG66	6.9e-3	3:31:45	2:43:22	2:08:31	1:31:34	1:15:39	72	0.92
thetaG62	3.3e-3	4:28:24	3:08:32	2:44:23	2:22:54	1:46:40	126	0.89
maxG67	7.1e-3	3:16:40	2:08:40	1:31:25	1:30:21	1:31:25	61	0.98
thetaG65	3.6e-3	7:03:48	4:58:38	4:19:52	3:59:31	3:25:14	116	0.92
thetaG67	4.8e-3	-	11:13:25	9:43:20	8:25:47	6:19:05	141	1.5
maxG77	8.0e-3	-	4:21:11	3:35:31	3:02:44	2:32:35	71	1.43
thetaG77	4e-3	-	14:04:37	11:50:07	11:38:05	6:29:00	142	1.49

Table 1: MPI code on preprocessed SDP - Henry2 distributed memory system

Comparison with OpenMP version of CSDP

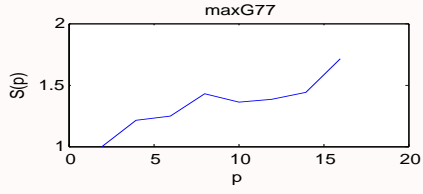
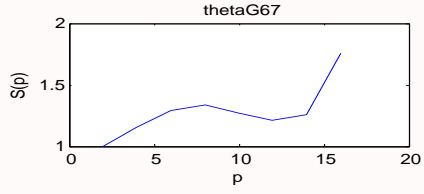
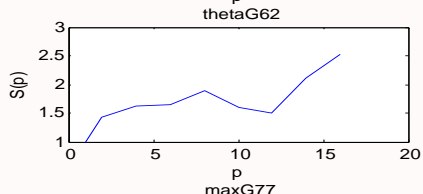
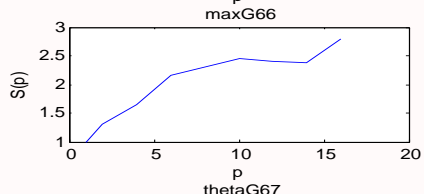
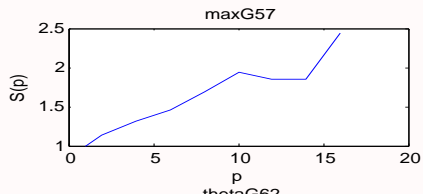
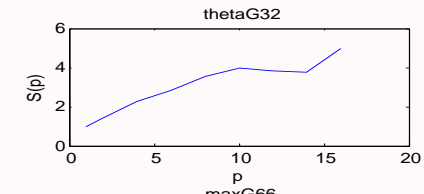
Prob	dual_gap	Time (h:m:s)				Iter	Memory (GB)
		p=1	p=2	p=4	p=8		
thetaG32	4e-7	12:01	6:54	4:07	2:43	23	0.63
thetaG48	5e-7	39:42	21:38	12:51	8:09	24	1.39
maxG57	8.5e-7	1:00:00	32:28	21:41	13:55	16	2.31
thetaG57	6.4e-7	2:44:10	1:29:17	52:37	36:01	23	3.84
maxG62	6.3e-7	2:32:10	1:26:08	52:48	36:01	16	4.51
maxG65	3e-7	3:58:30	2:08:26	1:18:51	51:11	18	5.88
maxG66	4e-7	5:27:01	2:59:38	1:51:10	1:12:39	17	7.43
thetaG62	7.6e-7	6:33:02	4:42:50	2:05:16	1:20:50	21	7.51
maxG67	2.5e-7	7:45:47	4:13:35	2:34:42	1:42:05	18	9.18
thetaG65	1.2e-7	10:39:10	5:50:43	3:23:05	2:08:55	23	9.81
thetaG67	1.6e-7	22:32:58	12:09:56	6:57:47	4:23:10	26	15.3
maxG77	7.3e-7	20:01:32	Failed	Failed	4:02:15	17	17.97
thetaG77	-	MM	MM	MM	MM	-	31.2

Table 2: OpenMP CSDP on original SDP - Power5 shared memory system



Scalability of algorithm on selected problems on up to 16 processors

Prob	p=1	p=2	p=4	p=8	p=16
thetaG32	19:44 (65)	14:26 (70)	8:40 (63)	5:35 (70)	3:58 (68)
	18:28 (46)	13:16 (51)	7:32 (44)	4:25 (52)	2:36 (48)
maxG57	42:36 (65)	37:36 (65)	32:18 (64)	29:01 (71)	17:31(63)
	31:41 (48)	27:29 (48)	20:35 (47)	17:10 (54)	7:06 (46)
maxG66	3:31:45 (72)	2:43:22 (70)	2:08:31 (70)	1:31:34 (71)	1:15:39 (72)
	2:24:26 (54)	1:53:17 (53)	1:26:51 (53)	50:56 (53)	30:01 (54)
thetaG62	4:28:24 (127)	3:08:32 (113)	2:44:23 (120)	2:22:54 (138)	1:46:40 (126)
	3:33:20 (105)	2:26:44 (91)	2:02:30 (98)	1:28:26 (116)	51:58 (104)
thetaG67	-	11:13:25 (161)	9:43:20 (152)	8:25:47 (145)	6:19:05 (141)
	-	8:01:01 (138)	6:47:16 (128)	5:34:20 (122)	3:37:25 (118)
maxG77	-	4:21:11 (72)	3:35:31 (71)	3:02:44 (69)	2:32:35 (71)
	-	2:51:05 (53)	2:16:23 (52)	1:49:42 (50)	1:13:41 (52)



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.
- Large scale parallel solution of structured POPs (more details in Sivaramakrishnan 2008).

Thank you for your attention!

Questions, Comments, Suggestions ?

The slides from this talk are available online at
[http://www4.ncsu.edu/~kksivara/
publications/kartik-siam08.pdf](http://www4.ncsu.edu/~kksivara/publications/kartik-siam08.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. **K. SIVARAMAKRISHNAN**, *A parallel dual interior point decomposition algorithm for structured polynomial optimization*, forthcoming. Available soon at <http://www4.ncsu.edu/~kksivara/publications/parallel-polyopt.pdf>
3. **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.
4. **K. NAKATA, K. FUJISAWA, M. FUKUDA, M. KOJIMA, AND K. MUROTA**, *Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results*, Mathematical Programming, 95(2003), pp. 303-327.
5. **K. FUJISAWA, M. FUKUDA, AND K. NAKATA**, *Preprocessing sparse semidefinite programs via matrix completion*, Optimization Methods and Software, 21(2006), pp. 17-39.
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. **J.B. LASSERRE**, *Convergent SDP-relaxations in polynomial optimization with sparsity*, SIAM Journal on Optimization, 17(2006), pp. 822-843.
8. **H. WAKI, S. KIM, M. KOJIMA, AND M. MURAMATSU**, *SparsePOP: a sparse semidefinite programming relaxation of polynomial optimization problems*, Available at <http://www.is.titech.ac.jp/~kojima/SparsePOP/>

Bibliography: 2

9. **K.C. KIWIEL**, *Proximity control in bundle methods for convex nondifferentiable optimization*, Mathematical Programming, 46(1990), pp. 105-122.
10. **K.C. KIWIEL**, *An aggregate subgradient method for nonsmooth convex minimization*, Mathematical Programming, 27(1983), pp. 320-341.
11. **B. BORCHERS**, *CSDP, a C library for semidefinite programming*, Optimization Methods and Software, 11 & 12(1999), pp. 613-623. Available at <https://projects.coin-or.org/Csdp/>.
12. **C. HELMBERG AND K.C. KIWIEL**, *A spectral bundle method with bounds*, Mathematical Programming, 93(2002), pp. 173-194.
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/>.