数理解析研究所講究録 1297 巻 2002 年 192-199

High Performance Grid Computing for Optimization Problem

京都大学大学院 工学研究科 建築学専攻
藤沢 克樹 (Fujisawa Katsuki)
Department of Architecture and Architectural Systems,
Kyoto University

Abstract: Grid computing has recently received much attention as a powerful and inexpensive methodology for solving large numerical problems that an existing single CPU cannot process. Ninf is a grid computing infrastructure which enables us to easily access computational resources including hardware and software library distributed across a wide area network. The basic Ninf system employs a client-server model, where the server and client machines are connected via a local area network or the Internet. We have been applying the Ninf system to optimization problems and polynomial systems of equations. Among others, Takeda et al. reported that the Successive Convex Relaxation Method (SCRM) on a LAN setting Ninf system can deal with some large size Quadratic Optimization Problems through numerical experiments. To solve larger size QOPs, however, we need more computing resources, and we implemented a highly parallel SCRM which utilizes several PC clusters connected via the Internet. In this talk, we discuss grid computing for optimization problems through some numerical experiments of the SCRM and polynomial systems of equations.

Key words: high performace computing, optimization software, cluster computing, grid computing, semidefinite programming

1 はじめに

半正定値計画問題（Semidefinite Programming : SDP）は組合せ最適化, 制御分野, 金融分野及び合成化学などの工学的分野に幅広い応用を持ち, 主双対内点法によって多項式時間で最適解を導き出すことができるので最近では注目度も高く頻繁に研究が行われている. 著者らのグループでは 1995 年より SDP を解くための主双対内点法を実現したソフトウェア SDPA (SemiDefinite Programming Algorithm) を開発を継続して行い、インターネット上より公開している [1]. 最新の SDPA は Ver 6.0 であり, その数値実験結果を公開している [2].

SDPA などの数理計画法のソフトウェアはアルゴリズムの改善や計算機能力の向上によって大幅に性能を向上させているが, 想定される数理計画問題のサイズも大きくなっているために単体のみの性能向上だけでは短時間に問題を解くことは困難である. それらの困難を克服するための並列計算の手法としてクラスタ計算 (Cluster Computing) とグリッド計算 (Grid Computing) が最近注目を集めている. 従来, 並列計算機はスーパーコンピュータをはじめとする非常に高価な計算機によって実現されていたが,PC などの安価な計算機を数十台, 数百台を非常に高速なネットワーク機器を用いて高密度に結合して, 一つの仮想的な並列計算機 (PC クラスタと呼ばれる) として使用する研究がさかんに行われている (クラスタ計算). また複数の PC クラスタ計算機をインターネットなどの

広域計算機ネットワークで接続してさらに大規模な分散型仮想並列計算機として使用するための研究も同時に行われている (グリッド計算).

そこで本論文ではクラスタ計算やグリッド計算を最適化問題に応用した例を紹介する. 一つは非凸最適化問題に対する逐次凸緩和法 [3, 4], もう一つは多変数多項式方程式系の全ての孤立解 (実根及び複素根) を求める研究である [5, 6]. またグリッド計算を実現するためには特別なソフトウェアを利用する必要がある. 今回は独立行政法人 産業技術総合研究所で開発されているグリッド計算を実現するためのソフトウェア Ninf [7] を用いる.

2 クラスタ計算とグリッド計算システム Ninf

クラスタ計算に関する研究は PC やワークステーションなどの高性能化, 低価格化に伴って 1990 年代半ばより開始された. 特に比較的に安価な PC で構成されたクラスタ計算機は PC クラスタと呼ばれており, 現在のクラスタ計算機の主流になっている. クラスタ計算機は複数の独立した計算機を高性能 LAN で結合し, 単一システムのイメージを提供する並列計算技術である. 特に一般の PC 技術を活用する PC クラスタでは, 近年の PC の高性能化と低価格化によって従来のスーパーコンピューターの数十倍から数百倍のコストパフォーマンスを達成している.

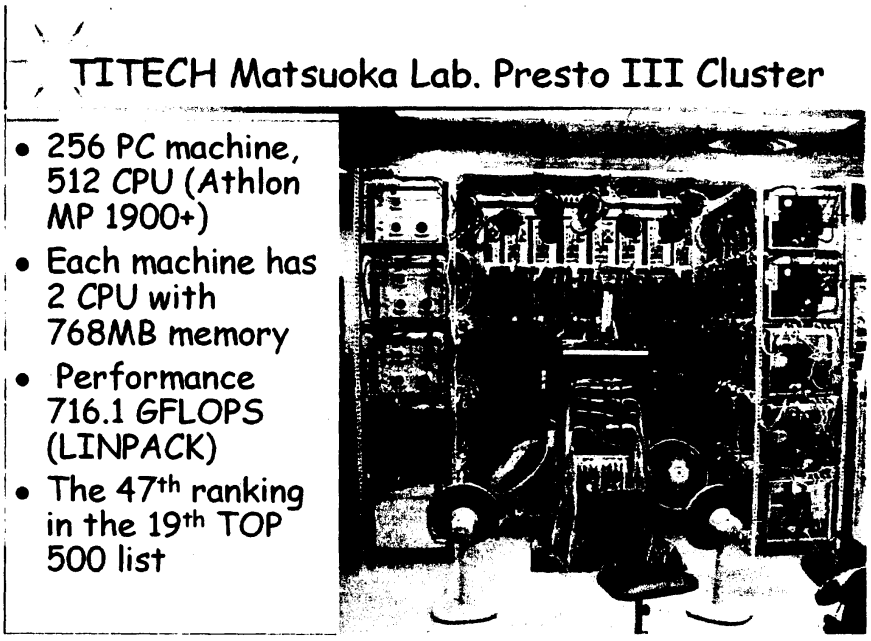

図 1: PC クラスタ (Athlon 1900+ : 512 CPU) : 東京工業大学 数理・計算科学専攻 松岡研究室. 世界第 47 位 (2002 年 6 月) の性能を持つ

クラスタ計算に関する技術は主にハードウェアとソフトウェアの部分に分けることができる. PC クラスタでは CPU やメモリなどの直接数値計算に関わる部分に重点的に投資されている. 特にネットワークカードやネットワークスイッチは Myrinet などの高価で高性能の部品が使用されることが多い. 逆にビデオカードや外部入出力装置は必要最小限の部品で済まされることが多い. またソフトウェアでは並列計算を実現するためのソフトウェア MPI や OpenMP などが有名である. また PC クラスタの全体性能を上げるために

直接ハードウェアの制御を行ったり効率の良いプロセスのスケジューリングを行う SCore が使用されている. 図 1 は本研究で使用している PC クラスタの一例である. 図で示したように東京工業大学 数理・計算科学専攻 松岡研究室に設置されている. 256 台の PC に合計 512 個の CPU が搭載されている. 全体性能は, 716.1 GFlops に達し,2002 年 6 月の時点で世界第 47 位の計算機として認定されている (TOP500: http://www.top500.org).

次にグリッド計算を実現するためのソフトウェア Ninf(広域並列計算システム) [7] について説明を行う. ソフトウェアの並列化の技術では既に述べたように MPI や OpenMP などが有名だが Ninf はこれらの技術とはやや異なった特徴を持っている. これらのクラスタ計算の技術は基本的に全ての計算機が LAN 等て接続され近接している場合に用いられる. Ninf では遠隔地のクラスタ計算機同士を高速なネットワーク (インターネット等) で接続して, お互いの計算機資源 (特に CPU パワー) を有効に活用し, 大規模な問題を効率良く解くことが主要な目的になっている.

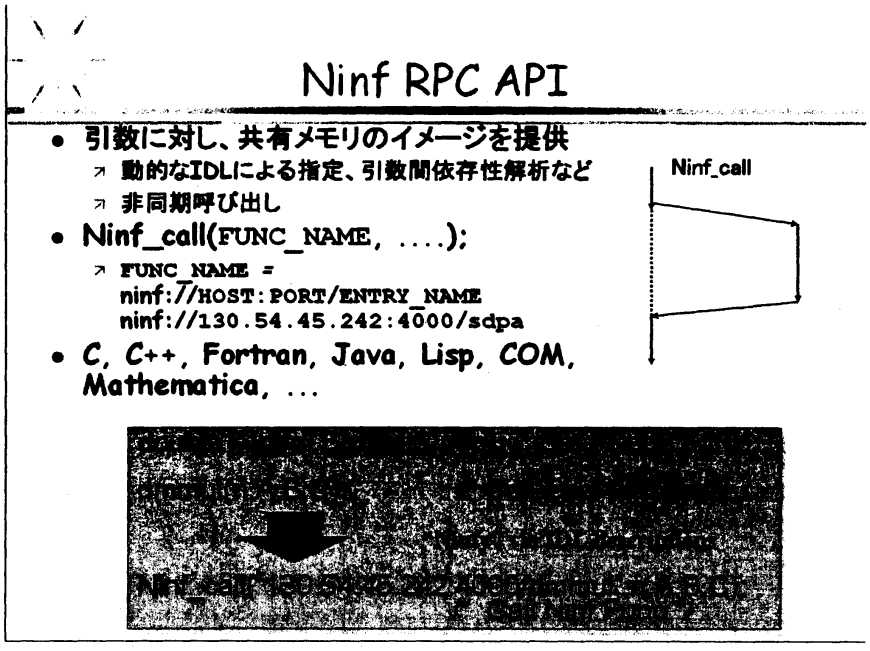

図 2: Ninf RPC API

Ninf 上には以下のような特徴がある. 1: 通常の関数に似たインターフェイスを持っているので簡単な関数等の記述で並列動作するソフトウェアを実現することが可能である. 2: インターネット等によって広域に接続, 提供されているハードウェア, ソフトウェアの利用が可能である. 3: 多様な言語 (C, C++, Fortran, Java など) を開発に用いることが可能である. 図 2 は Ninf の RPC API を簡単に説明したものである. 通常の関数呼び出しにおいては関数名と引数を指定するのが普通だが Ninf においては, 広域に配置されたライブラリにアクセスが可能なので, 図 2 のようにアクセスするライブラリがインストールされている計算機の IP アドレスやポート番号を指定する. また Ninf では同期, 非同期の呼出しがサポートされており, 非同期の場合では効率良く大量のプロセスの並列動作を行うことができる.

図 3 は Ninf を用いて実現された Client-Server system である. 次節で数値実験結果を二つ (逐次凸緩和法, ホモトピー法) を示す. この場合 Ninf Client は 1 台の PC である. 図 3 にあるような役割を担う. そして Ninf Client から 非同期に Ninf Server (PC クラ

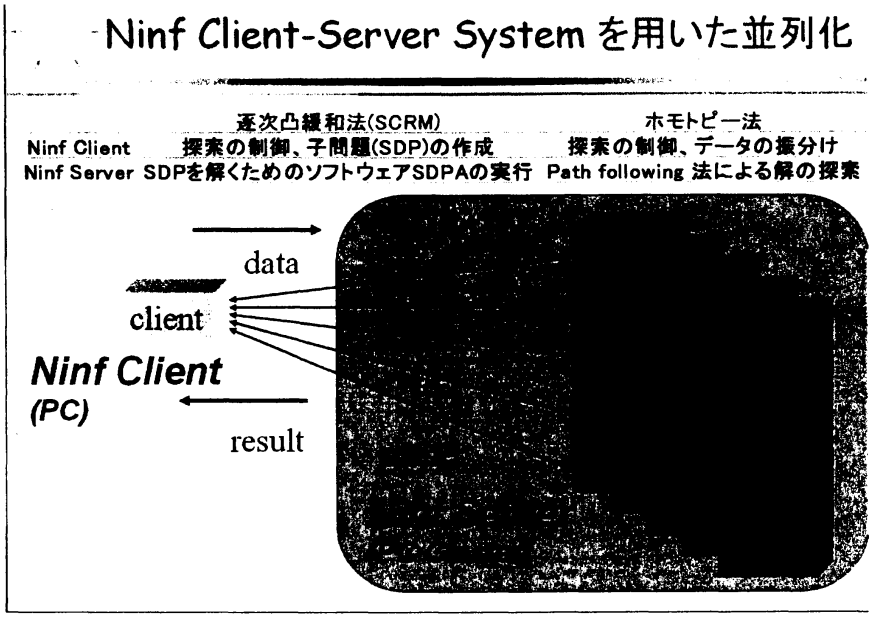

図 3: Ninf Client-Server system

スタ) 上の複数の PC が呼び出される. そして Ninf Server 上では同時に多くの子問題が生成されて問題が解かれることになる. この場合 Ninf Server は複数の PC クラスタで構成されていてもよく, また複数の PC クラスタはそれぞれ遠隔地に設置されていてもよい. その場合は接続されているインターネットなどのネットワークの転送速度が重要になってくる.

3 数値実験結果

3.1 Ninf を用いた逐次凸緩和法 (SCRM) の並列化

次に以下のような非凸二次最適化問題 (NQP) を考える. ここで $\boldsymbol{c} \in \boldsymbol{R}^n, \boldsymbol{a}_i \in \boldsymbol{R}^n, \boldsymbol{q}_j \in \boldsymbol{R}^n$ は定数ベクトル, $b_i \in \boldsymbol{R}^1, \gamma_j \in \boldsymbol{R}^1$ は定数, $\boldsymbol{Q} \in \boldsymbol{R}^{n \times n}$ は $n \times n$ の定数行列である.

$$\text{(NQP)} \quad \left| \begin{array}{ll} \max & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subj.to} & \boldsymbol{x} \in F \end{array} \right. \tag{1}$$

$$F \equiv \left\{ \boldsymbol{x} \in R^n : \begin{array}{r} \boldsymbol{a}_i^T \boldsymbol{x} + b_i \leq 0 \ (i = 1, \cdots, m_1) \\ \boldsymbol{x}^T \boldsymbol{Q}_j \boldsymbol{x} + \boldsymbol{q}_j^T \boldsymbol{x} + \gamma_j \leq 0 \ (j = 1, \cdots, m_2) \end{array} \right\}$$

ここで領域 F は有界であると仮定する. F は上記のように m_1 本の線形制約式と m_2 本の二次制約式から構成されている. ここで行列 $\boldsymbol{Q}$ は正定値でもなくてもよいので, LP や SDP だけでなく, 一般の非凸 (二次) 計画問題も含まれている. 武田ら [4] は NQP に対して, 逐次半正定値計画緩和法 [3] を適用して, Ninf 上で並列動作する SDPA を用いて

計算を行っている. 逐次半正定値計画緩和法では, はじめに領域 F を完全に含む凸領域を定義する. そのあと LP 緩和や SDP 緩和などを繰り返し用いて, この領域が F の凸包に収束するまで計算を継続する. 目的関数は上記のように一般性を失うことなく線形関数 $(\boldsymbol{c}^T\boldsymbol{x})$ を仮定することができる. F 上での $\boldsymbol{c}^T\boldsymbol{x}$ の最大値と F の凸包上での $\boldsymbol{c}^T\boldsymbol{x}$ の最大値は一致するので, この性質を用いて NQP を解くのが大きな特徴である. このアルゴリズムでは 1 反復の中で複数の SDP 緩和問題を解かなければならない. これらの複数の SDP 緩和問題は前節で説明した Ninf を用いて異なる計算機上で非同期に解くことができる. その結果, 多くの計算機資源を利用することによって大幅な高速化が達成されている. 次に PC クラスタ (CPU : Pentium III 824MHz 128 台での実験結果を示す. 使用する問題は非凸二次計画問題 [4] で LC80-144 ($n = 81$, $m_1 = 120$, $m_2 = 1$) と Frac50-20 ($n = 51$, $m_1 = 21$, $m_2 = 1$) の二つである. 表 1 は PC クラスタでの CPU の台数と高速化の比率との関係を示している. 例えば CPU が 128 台のときに LC80-144 では約 92 倍, Frac50-20 では約 90 倍の高速化を達成している. Ninf Server 上の SDPA の起動に要するオーバーヘッドを考慮すると, 解く問題の規模が大きい方が Ninf Server 上での SDPA の実行時間の占める割合が大きくなって, 並列化の効率は向上することになる. 通常 CPU 128 台を使用した場合に 約 90 倍の並列化効率が得られることは珍しい.

表 1: PC クラスタ (CPU 128 台) での CPU の台数と高速化の比率との関係

	LC80-144		Frac50-20	
CPU(台)	実行時間 (秒)	比率	実行時間 (秒)	比率
1	33125	1.00	289259	1.00
2	16473	2.01	145980	1.98
4	8238	4.02	72343	3.99
8	4145	7.99	36272	7.97
16	2099	15.78	18595	15.56
32	1118	29.62	9424	30.69
64	624	53.08	4822	60.00
128	361	91.76	3200	90.39

表 2: Grid 環境 (図 4) での数値実験結果

問題名	LC80-144		BLevel20-3	
Ninf Server の場所	Kyoto	Tokyo	Kyoto	Tokyo
解かれた子問題の数	1795	1333	883	597
サーバー上での総実行時間 (秒)	16486.5	16395.6	1053.6	984.2
総転送時間 C → S(秒)	0.150	0.117	0.046	0.028
総転送時間 C ← S(秒)	0.275	0.069	0.130	0.182

表 1 の数値実験では Ninf Client と Ninf Server の PC は LAN で接続されていたが, 次に Ninf Server の PC クラスタを二つ増やして一つを遠隔地に設置する. 具体的には図 2 のような Grid 環境で実験を行う. Ninf Client PC は京大に設置して LAN で接続された Ninf Server (PC クラスタ: Pentium III 850MHz 4 台) を使用すると同時に,Internet で接続されている東工大の Ninf Server (PC クラスタ: Pentium III Xeon 700MHz 4 台) も利用する. 表 2 が Grid 環境下における数値実験結果である. 解かれた子問題の数ではやや差が見られるが, これは上記のように CPU の種類が異なるためであり, サーバーの上の総実行時間は京都と東京ともにほぼ同じである. また C → S (C ← S) とは Client から Server(Server から Client) への総データ転送時間であるが, この値も非常に小さく抑えることに成功している. この場合京都と東京ではネットワークの転送時間に大きな

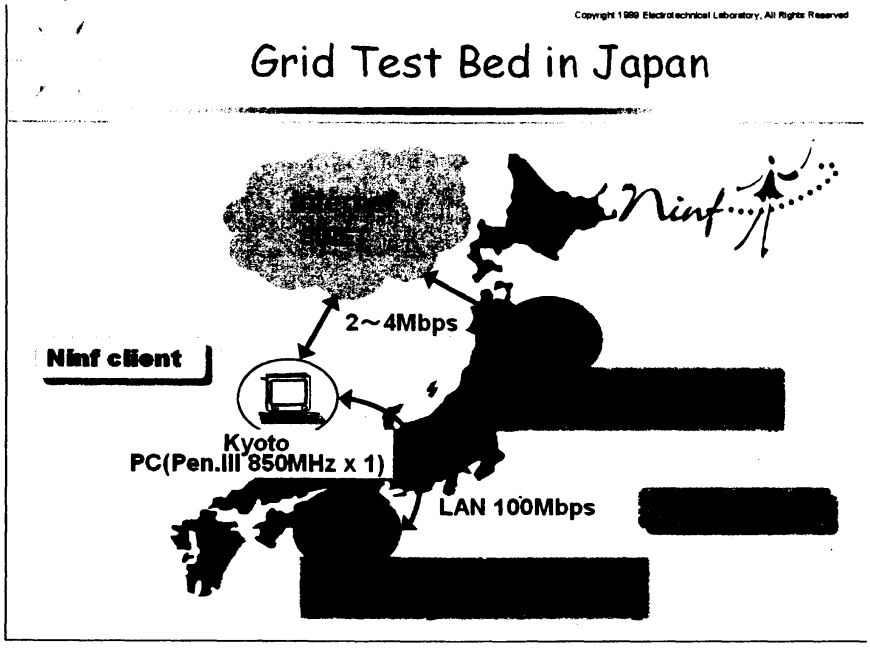

図 4: Grid 環境 (京都 $\Longleftrightarrow$ 東京)

差が見られるが, サーバー上での総実行時間が均等していることから Grid 環境上での分散並列計算が成功していることがわかる. これらの結果のように Ninf による高速化は目覚しく SDPA だけでなく他の最適化ソフトウェアとの組合せも今後は期待されるところである.

3.2 ホモトピー法による多変数多項式方程式の全解探索

多変数多項式方程式系の全ての孤立解 (実根及び複素根) を求めることは, 数学的な基本問題であるだけでなく, 工学的にはシステムと制御, 最適化, ロボット工学, 計算幾何学, 化学工学等に応用を持つ重要な問題である. [5, 6] では大規模な多変数多項式方程式系の全ての根を計算する実用的な解法を開発, 実験することを目的としている. 解法は多面体的ホモトピー法を用いられている. また [5, 6] では下記のような cyclic n-roots problem と呼ばれる問題に対して数値実験が行われている.

$$
\begin{aligned}
&x_1 + x_2 + \cdots + x_n = 0 \\
&x_1x_2 + x_2x_3 + \cdots + x_nx_1 = 0 \\
&x_1x_2x_3 + \cdots + x_nx_1x_2 = 0 \\
&\cdots \\
&x_1x_2x_3 \cdots x_n - 1 = 0,
\end{aligned}
$$

多項式方程式の次数及び変数の個数が増加するにつれて, 求めるべき根の個数がそれらの組合せ的な関数で増加するので (表 3 参照), 単一の計算機上では大規模な問題を解くことは事実上不可能である. よってクラスタ計算などの手法を駆使してアルゴリズムを並列化して大規模問題の解を求めている. 表 3 に見られるように求める解の個数は n が

大きくなるに連れて膨大な数になる. このため従来の方法では実用的な時間で全ての解を求めることは困難である. 一つの解を求めるためには, 初期解から出発してホモトピーパスを追跡するソフトウェアを一つ実行する必要がある. そのため重根などの場合を除いて全ての解を求めるためには解の個数と同数のプロセスを起動する必要がある. しかし個々のプロセスは独立性が高いので, 多数のプロセスを並列動作させることが可能である. n が 12 以下の場合には, すでに全ての孤立解が求められているが, n が 13 以上の場合には, いまだ全ての孤立解は求められていなかった. しかしこれらの手法を駆使して n が 13 の場合において全ての孤立解を求めることに成功している.

表 3: cyclic n-roots problem($n = 11, 12, 13$) の求める孤立解の個数

n	解の個数
11	184,756
12	367,488
13	2,696,044

表 4: PC クラスタでの CPU の台数と高速化の比率との関係

	cyclic-11		cyclic-12		cyclic-13	
# CPUs	実行時間 (秒)	比率	実行時間 (秒)	比率	実行時間 (秒)	比率
2	47,345	1.00				
4	23,674	2.00				
8	11,852	3.99				
16	5,927	7.99				
32	2,967	15.96				
64	1,487	31.84	2,592	1.00		
128			1,332	1.95	10,151	1.00
256			703	3.69	5,191	1.95
# paths traced	16,796		41,696		208,012	

表 4 は $n = 11, 12, 13$ の場合においてホモトピーパスを追跡するソフトウェアを図 3 の Ninf Client-Server system を用いて並列化した実験結果である. CPU の数を変化させながら, 実行時間の高速化の比率を計算している. # paths traced とは何本のホモトピーパスが計算されたかを示していて, Ninf Server 上で実行されたホモトピーパス追跡のソフトウェアの実行回数と一致している. $n = 11$ の場合では PC クラスタ (Celeron 500MHz 64 台) で実験を行い, $n = 12, 13$ の場合では前出の PC クラスタ (Athlon 1900+ 512 台 : 図 1 参照) を用いている. 表 4 を見ると使用する CPU の台数に見合った効率が得られていることがわかる. また 1 CPU のときでは非常に実行時間がかかると推定される問題も短時間で解を算出することに成功している.

謝辞 : 共同研究や情報, 資料提供などをしていただきました東京工業大学の小島 (政) 先生, 松岡先生, 合田先生, 中田先生及び各研究室の皆様,(株) 東芝の武田さん及び独立行政法人 産業技術総合研究所グリッド研究センターの関口先生, 田中先生, 中田先生, 建部先生, 首藤先生らには深く感謝致します.

参考文献

[1] K. Fujisawa, M. Kojima, K. Nakata and M. Yamashita, "SDPA Ver 6.0 (Semidefinite Programming Algorithm) User's Manual ,", August 2002, http://www.is.titech.ac.jp/~kojima/sdpa/index.html

[2] M. Yamashita, K. Fujisawa and M. Kojima, "Implementation and Evaluation of SDPA 6.0 (SemiDefinite Programming Algorithm 6.0)," Technical Report B-383, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo, Japan, http://www.is.titech.ac.jp/research/research-report/B/B-383.ps.gz

[3] M. Kojima and L. Tuncel, "Discretization and Localization in Successive Convex Relaxation for Nonconvex Quadratic Optimization Problems", Mathematical Programming, **89**, pp. 79 111, 2000.

[4] A. Takeda, K. Fujisawa, Y. Fukaya and M. Kojima, "Parallel Implementation of Successive Convex Relaxation Methods for Quadratic Optimization Problems", To appear in Journal of Global Optimization.

[5] A. Takeda, M. Kojima and K. Fujisawa, "Enumeration of All Solutions of a Combinatorial Linear Inequality System Arising from the Polyhedral Homotopy Continuation Method," Journal of the Operations Research Society of Japan, Vol.45, No.1, pp. 64–82, 2002.

[6] Y. Day, S. Kim and M. Kojima, "Computing All Nonsingular Solutions of Cyclic-n Polynomial Using Polyhedral Homotopy Continuation Methods,", To appear in Journal of Computational and Applied Mathematics.

[7] M. Sato, H. Nakada, S. Sekiguchi, S. Matsuoka, U. Nagashima and H. Takagi, "Ninf: A Network based Information Library for a Global World-Wide Computing Infrastructure", HPCN'97 (LNCS-1225), pp. 491-502, 1997.