Implementación de un método de programación semidefinida usando computación paralela

Oscar Francisco Peredo Andrade

Presentación para optar al Título de Ingeniero Civil Matemático
Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile

3 de Junio de 2010

Esquema

Esquema

Motivación

Motivación

- Experimentar el proceso de desarrollo de una aplicación científica desde su diseño, realizado en sistemas de alto nivel (*MATLAB*), hasta su implementación basada en cálculo paralelo, realizada en sistemas de bajo nivel (*C*).

Motivación

- Experimentar el proceso de desarrollo de una aplicación científica desde su diseño, realizado en sistemas de alto nivel (*MATLAB*), hasta su implementación basada en cálculo paralelo, realizada en sistemas de bajo nivel (*C*).
- Resolver un problema de programación semidefinida no lineal:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.a} \quad & h(x) = 0 \\ & G(x) \preceq 0 \end{aligned} \tag{1}$$

donde $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^p$ y $G : \mathbb{R}^n \to \mathbb{S}^m$ son clase C^2, usando una versión del algoritmo **Filter-SDP** desarrollado en [GR06].

Objetivos

Objetivos

- Estudio de herramientas de cálculo paralelo aplicables en el algoritmo **Filter-SDP**.

Objetivos

- Estudio de herramientas de cálculo paralelo aplicables en el algoritmo **Filter-SDP**.
- Implementación del algoritmo **Filter-SDP** utilizando C.

Objetivos

- Estudio de herramientas de cálculo paralelo aplicables en el algoritmo **Filter-SDP**.
- Implementación del algoritmo **Filter-SDP** utilizando C.
- Diseño de nuevas fases de restauración para el algoritmo **Filter-SDP** utilizando MATLAB.

Objetivos

- Estudio de herramientas de cálculo paralelo aplicables en el algoritmo **Filter-SDP**.
- Implementación del algoritmo **Filter-SDP** utilizando C.
- Diseño de nuevas fases de restauración para el algoritmo **Filter-SDP** utilizando MATLAB.
- Realización de pruebas numéricas (implementación C y nuevas fases de restauración).

Esquema

Introducción

Antecedentes

- Programación semidefinida lineal (SDP)
- Filter-SDP
- Computación paralela

Trabajo realizado

- Estudio de sistemas de cálculo paralelo
- Implementación en C: `fnlsdp`
- Diseño de distintas fases de restauración

Resultados numéricos

Conclusiones

Trabajo futuro

Esquema

Programación semidefinida lineal (SDP)

Programación semidefinida lineal (SDP)

Formulación primal ([VB96]):

$$\begin{aligned} \min_{x \in \mathbb{R}^m} \quad & c^T x \\ \text{s.a} \quad & F_0 + \sum_{i=1}^{m} x_i F_i \succeq 0 \end{aligned} \tag{2}$$

con $c \in \mathbb{R}^m$ y $F_0, \ldots, F_n \in \mathbb{S}^n$.

Programación semidefinida lineal (SDP)

Formulación primal ([VB96]):

$$
\begin{aligned}
\min_{x \in \mathbb{R}^m} \quad & c^T x \\
\text{s.a} \quad & F_0 + \sum_{i=1}^{m} x_i F_i \succeq 0
\end{aligned}
\tag{2}
$$

con $c \in \mathbb{R}^m$ y $F_0, \ldots, F_n \in \mathbb{S}^n$.
Formulación dual:

$$
\begin{aligned}
\max_{Z \in \mathbb{S}^n} \quad & -\mathrm{Tr}(F_0 Z) \\
\text{s.a} \quad & Tr(F_i Z) = c_i,\ i = 1, ..., m \\
& Z \succeq 0
\end{aligned}
\tag{3}
$$

Programación semidefinida: aplicaciones

Programación semidefinida: aplicaciones

El problema de programación lineal:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^T x \\ \text{s.a} \quad & Ax + b \geq 0 \end{aligned} \tag{4}$$

Programación semidefinida: aplicaciones

El problema de programación lineal:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^T x \\ \text{s.a} \quad & Ax + b \geq 0 \end{aligned} \tag{4}$$

tiene una formulación SDP:

Programación semidefinida: aplicaciones

El problema de programación lineal:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^T x \\ \text{s.a} \quad & Ax + b \geq 0 \end{aligned} \tag{4}$$

tiene una formulación SDP:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^T x \\ \text{s.a} \quad & \operatorname{diag}(Ax + b) \succeq 0 \end{aligned} \tag{5}$$

Programación semidefinida: aplicaciones

El problema de programación lineal:

$$\begin{array}{cc} \min\limits_{x\in\mathbb{R}^n} & c^T x \\ \text{s.a} & Ax + b \geq 0 \end{array} \tag{4}$$

tiene una formulación SDP:

$$\begin{array}{cc} \min\limits_{x\in\mathbb{R}^n} & c^T x \\ \text{s.a} & \operatorname{diag}(Ax + b) \succeq 0 \end{array} \tag{5}$$

o equivalentemente:

$$\begin{array}{cc} \min\limits_{x\in\mathbb{R}^n} & c^T x \\ \text{s.a} & \underbrace{\operatorname{diag}(b)}_{F_0} + \sum_{i=1}^n x_i \underbrace{\operatorname{diag}(A_{\cdot,i})}_{F_i} \succeq 0 \end{array} \tag{6}$$

Programación semidefinida: aplicaciones

Programación semidefinida: aplicaciones

El problema de programación cuadrática convexa también se puede formular como SDP.

Programación semidefinida: aplicaciones

El problema de programación cuadrática convexa también se puede formular como SDP. Para $c \in \mathbb{R}^n$, $b, g \in \mathbb{R}^m$, $d \in \mathbb{R}$, $A, H \in \mathbb{R}^{m \times n}$:

Programación semidefinida: aplicaciones

El problema de programación cuadrática convexa también se puede formular como SDP. Para $c \in \mathbb{R}^n, b, g \in \mathbb{R}^m, d \in \mathbb{R}, A, H \in \mathbb{R}^{m \times n}$:

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & (Ax + b)^T (Ax + b) - c^T x - d \\ \text{s.a} & Hx + g \geq 0 \end{array} \tag{7}$$

Programación semidefinida: aplicaciones

El problema de programación cuadrática convexa también se puede formular como SDP. Para $c \in \mathbb{R}^n$, $b, g \in \mathbb{R}^m$, $d \in \mathbb{R}$, $A, H \in \mathbb{R}^{m \times n}$:

$$\begin{array}{cl} \min\limits_{x \in \mathbb{R}^n} & (Ax+b)^T(Ax+b) - c^T x - d \\ \text{s.a} & Hx + g \geq 0 \end{array} \tag{7}$$

Equivalentemente:

$$\begin{array}{crcl} \min\limits_{x \in \mathbb{R}^n, t \in \mathbb{R}} & & t & \\ \text{s.a} & (Ax+b)^T(Ax+b) - c^T x - d & \leq & t \\ & Hx + g & \geq & 0 \end{array} \tag{8}$$

Programación semidefinida: aplicaciones

El problema de programación cuadrática convexa también se puede formular como SDP. Para $c \in \mathbb{R}^n$, $b, g \in \mathbb{R}^m$, $d \in \mathbb{R}$, $A, H \in \mathbb{R}^{m \times n}$:

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & (Ax+b)^T(Ax+b) - c^T x - d \\ \text{s.a} & Hx + g \geq 0 \end{array} \tag{7}$$

Equivalentemente:

$$\begin{array}{lrcl} \min\limits_{x \in \mathbb{R}^n, t \in \mathbb{R}} & & t & \\ \text{s.a} & (Ax+b)^T(Ax+b) - c^T x - d & \leq & t \\ & Hx + g & \geq & 0 \end{array} \tag{8}$$

y utilizando el complemento de Schur (▸ Ir a Apéndice) se tiene:

Programación semidefinida: aplicaciones

El problema de programación cuadrática convexa también se puede formular como SDP. Para $c \in \mathbb{R}^n$, $b, g \in \mathbb{R}^m$, $d \in \mathbb{R}$, $A, H \in \mathbb{R}^{m \times n}$:

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & (Ax+b)^T(Ax+b) - c^T x - d \\ \text{s.a} & Hx + g \geq 0 \end{array} \tag{7}$$

Equivalentemente:

$$\begin{array}{lrcl} \min\limits_{x \in \mathbb{R}^n, t \in \mathbb{R}} & & t & \\ \text{s.a} & (Ax+b)^T(Ax+b) - c^T x - d & \leq & t \\ & Hx + g & \geq & 0 \end{array} \tag{8}$$

y utilizando el complemento de Schur (Ir a Apéndice) se tiene:

$$\begin{array}{lrcl} \min\limits_{x \in \mathbb{R}^n, t \in \mathbb{R}} & & t & \\ \text{s.a} & \begin{bmatrix} I_{m \times m} & Ax + b \\ (Ax+b)^T & c^T x + d + t \end{bmatrix} & \succeq & 0 \\ & \operatorname{diag}(Hx+g) & \succeq & 0 \end{array} \tag{9}$$

Programación semidefinida: métodos de resolución

Programación semidefinida: métodos de resolución

- Punto interior

Programación semidefinida: métodos de resolución

- Punto interior
- Paquete o haz espectral (*bundle methods*)

Programación semidefinida: métodos de resolución

- Punto interior
- Paquete o haz espectral (*bundle methods*)

Esquema

Filter-SDP

Filter-SDP

- Algoritmo desarrollado por Gómez & Ramirez [GR06] para resolver problemas del tipo:

$$\begin{array}{cc} \min\limits_{x \in \mathbb{R}^n} & f(x) \\ \text{s.a} & h(x) = 0 \\ & G(x) \preceq 0 \end{array} \tag{10}$$

Filter-SDP

- Algoritmo desarrollado por Gómez & Ramirez [GR06] para resolver problemas del tipo:

$$\begin{array}{cc} \min\limits_{x\in\mathbb{R}^n} & f(x) \\ \text{s.a} & h(x) = 0 \\ & G(x) \preceq 0 \end{array} \tag{10}$$

- Basado en el algoritmo de *filter-SQP*, desarrollado en [FGLT01] para resolver problemas de programación no lineal.

Filter-SDP

- Algoritmo desarrollado por Gómez & Ramirez [GR06] para resolver problemas del tipo:

$$\begin{array}{cc} \min\limits_{x \in \mathbb{R}^n} & f(x) \\ \text{s.a} & h(x) = 0 \\ & G(x) \preceq 0 \end{array} \tag{10}$$

- Basado en el algoritmo de *filter-SQP*, desarrollado en [FGLT01] para resolver problemas de programación no lineal.
- Principales aspectos: utilización de un filtro, resolución de un problema tangencial cuadrático y desarrollo de una fase de restauración.

Filter-SDP: filtro

Filter-SDP: filtro

- Idea central: utilizar un enfoque multiobjetivo, donde se minimice la función objetivo $f(x)$ y una función de mérito $\theta(x)$, que cuantifica la factibilidad de un punto, donde:

$$\theta(x) = \|h(x)\|_2 + \max\{0, \lambda_1(G(x))\} \tag{11}$$

Filter-SDP: filtro

- Idea central: utilizar un enfoque multiobjetivo, donde se minimice la función objetivo $f(x)$ y una función de mérito $\theta(x)$, que cuantifica la factibilidad de un punto, donde:

$$\theta(x) = \|h(x)\|_2 + \max\{0, \lambda_1(G(x))\} \tag{11}$$

- El filtro $\mathcal{F} = \{(\theta(x_i), f(x_i))\}_{i=1}^{n}$ almacena información de puntos que no se dominan entre si:

$$x_k \text{ domina a } x_p \Leftrightarrow f(x_k) \leq f(x_p) \text{ y } \theta(x_k) \leq \theta(x_p)$$

Filter-SDP: filtro

- Idea central: utilizar un enfoque multiobjetivo, donde se minimice la función objetivo $f(x)$ y una función de mérito $\theta(x)$, que cuantifica la factibilidad de un punto, donde:

$$\theta(x) = \|h(x)\|_2 + \max\{0, \lambda_1(G(x))\} \tag{11}$$

- El filtro $\mathcal{F} = \{(\theta(x_i), f(x_i))\}_{i=1}^n$ almacena información de puntos que no se dominan entre si:

$$x_k \text{ domina a } x_p \Leftrightarrow f(x_k) \leq f(x_p) \text{ y } \theta(x_k) \leq \theta(x_p)$$

- Se utiliza como criterio para escoger nuevos puntos x_k de la sucesión convergente (salvo subsucesiones) al óptimo (local).

Filter-SDP: filtro

- Idea central: utilizar un enfoque multiobjetivo, donde se minimice la función objetivo $f(x)$ y una función de mérito $\theta(x)$, que cuantifica la factibilidad de un punto, donde:

$$\theta(x) = \|h(x)\|_2 + \max\{0, \lambda_1(G(x))\} \tag{11}$$

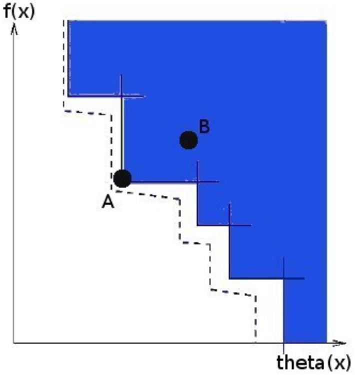
- El filtro $\mathcal{F} = \{(\theta(x_i), f(x_i))\}_{i=1}^{n}$ almacena información de puntos que no se dominan entre si:

$$x_k \text{ domina a } x_p \Leftrightarrow f(x_k) \leq f(x_p) \text{ y } \theta(x_k) \leq \theta(x_p)$$

- Se utiliza como criterio para escoger nuevos puntos x_k de la sucesión convergente (salvo subsucesiones) al óptimo (local).
- Todos los puntos x_k que se guarden en el filtro deben ser aceptables:

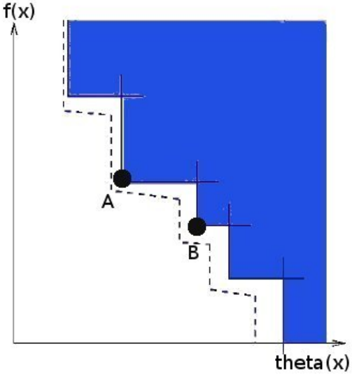
$$\overline{x} \text{ es aceptable por } \mathcal{F}$$
$$\Leftrightarrow$$
$$\forall i \in \{1, \ldots, n\} : \theta(\overline{x}) \leq \beta\theta(x_i) \text{ ó } f(\overline{x}) + \gamma\theta(\overline{x}) \leq f(x_i)$$

Filter-SDP: filtro



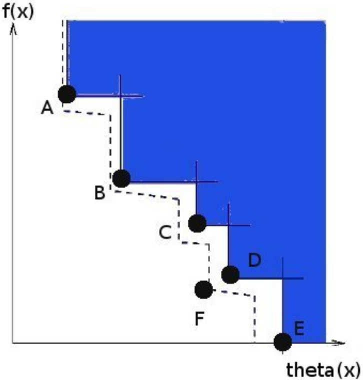
A domina a B

Filter-SDP: filtro



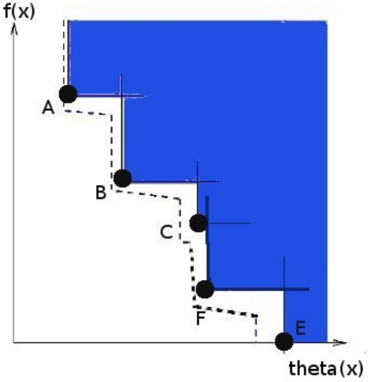
A no domina a B

Filter-SDP: filtro



F es aceptable por el filtro $\{A, B, C, D, E\}$

Filter-SDP: filtro



Agregar punto F al filtro $\{A, B, C, D, E\}$.
Nuevo filtro: $\{A, B, C, F, E\}$

Filter-SDP: problema tangencial cuadrático

Filter-SDP: problema tangencial cuadrático

En cada iteración del algoritmo, un punto $x_{k+1} = x_k + d_k$ se contruye utilizando la solución d_k de un problema tangencial cuadrático (*trust region local semidefinite approximation*), asociado al punto x_k y a un radio ρ:

Filter-SDP: problema tangencial cuadrático

En cada iteración del algoritmo, un punto $x_{k+1} = x_k + d_k$ se contruye utilizando la solución d_k de un problema tangencial cuadrático (*trust region local semidefinite approximation*), asociado al punto x_k y a un radio ρ:

$$
\begin{array}{lll}
QP(x,\rho): & \min\limits_{d \in \mathbb{R}^n} & \nabla f(x) d + \frac{1}{2} d^T B d \\
 & s.a. & h(x) + Dh(x) d = 0 \\
 & & G(x) + DG(x) d \preceq 0 \\
 & & \|d\|_\infty \leq \rho
\end{array}
$$

Filter-SDP: fase de restauración

Filter-SDP: fase de restauración

La fase de restauración tiene como objetivo generar un punto x_k que cumpla con las siguientes condiciones:

Filter-SDP: fase de restauración

La fase de restauración tiene como objetivo generar un punto x_k que cumpla con las siguientes condiciones:

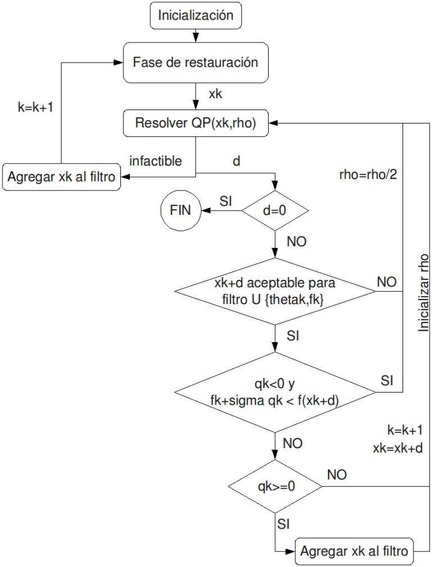
A1 $(\theta(x_k), f(x_k))$ es aceptable para el filtro $\mathcal{F}^{k-1}$.

B1 $QP(x_k, \rho)$ es factible.

donde $\mathcal{F}^{k-1}$ es el filtro obtenido de la iteración anterior y $\theta(x)$ es la función de mérito escogida.

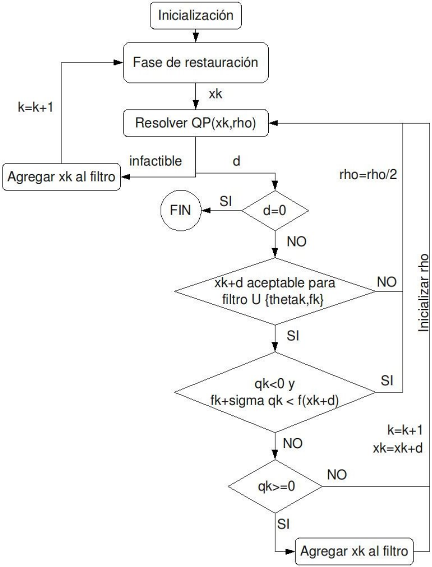
Filter-SDP: algoritmo

▸ Ir a pseudocódigo



Filter-SDP: algoritmo

Ir a pseudocódigo



Esquema

Computación paralela

Computación paralela

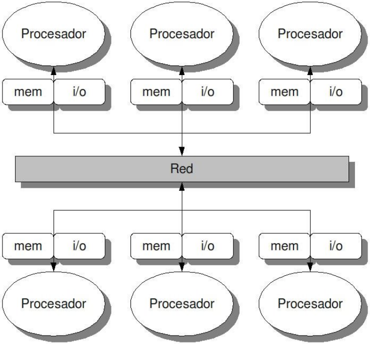
- Situación en la que al menos 2 procesadores cooperan intercambiando información mientras trabajan en diferentes partes de uno o más problemas.

Computación paralela

- Situación en la que al menos 2 procesadores cooperan intercambiando información mientras trabajan en diferentes partes de uno o más problemas.
- Diferentes clasificaciones según: número de procesadores, acceso a la memoria, redes que comunican a los procesadores, I/O, etc.

Computación paralela

- Situación en la que al menos 2 procesadores cooperan intercambiando información mientras trabajan en diferentes partes de uno o más problemas.
- Diferentes clasificaciones según: número de procesadores, acceso a la memoria, redes que comunican a los procesadores, I/O, etc.
- En este trabajo se estudiaron sistemas que funcionan en arquitecturas de memoria distribuída:



Computación paralela: aspectos paralelizables de Filter-SDP

Computación paralela: aspectos paralelizables de Filter-SDP

- Cálculo de $\theta(x)$

Computación paralela: aspectos paralelizables de Filter-SDP

- Cálculo de $\theta(x)$
- Resolución de $QP(x, \rho)$

Computación paralela: aspectos paralelizables de Filter-SDP

- Cálculo de $\theta(x)$
- Resolución de $QP(x, \rho)$
- Otras operaciones algebraicas

Computación paralela: aspectos paralelizables de Filter-SDP

- Cálculo de $\theta(x)$
- Resolución de $QP(x, \rho)$
- Otras operaciones algebraicas

Esquema

Esquema

Estudio de sistemas de cálculo paralelo

Estudio de sistemas de cálculo paralelo

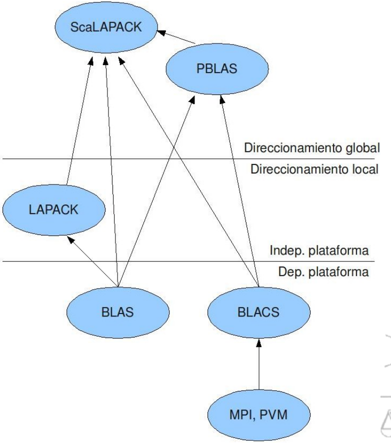
- ScaLAPACK: *Scalable Linear Algebra Package*

Estudio de sistemas de cálculo paralelo

- ScaLAPACK: *Scalable Linear Algebra Package*
- PCSDP: *C Library for Parallel Linear Semidefinite Programming*

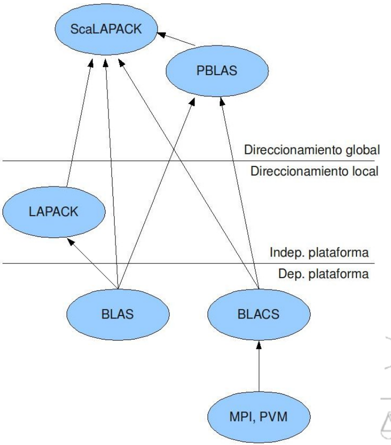
ScaLAPACK

ScaLAPACK


ScaLAPACK
PBLAS
Direccionamiento global
Direccionamiento local
LAPACK
Indep. plataforma
Dep. plataforma
BLAS
BLACS
MPI, PVM

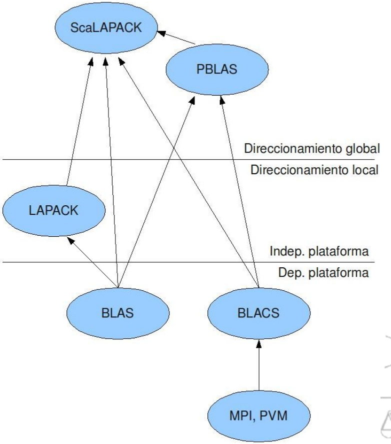
ScaLAPACK

- Estudio de las librerías básicas: BLAS, LAPACK, BLACS, PBLAS.



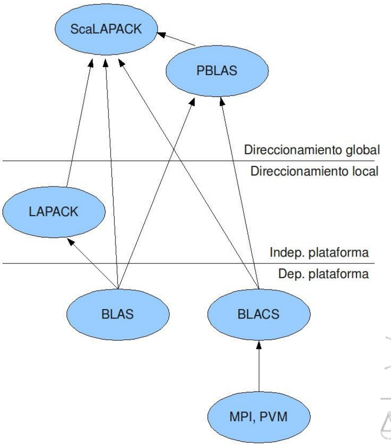
ScaLAPACK

- Estudio de las librerías básicas: BLAS, LAPACK, BLACS, PBLAS.
- *Intel Math Kernel Library*.



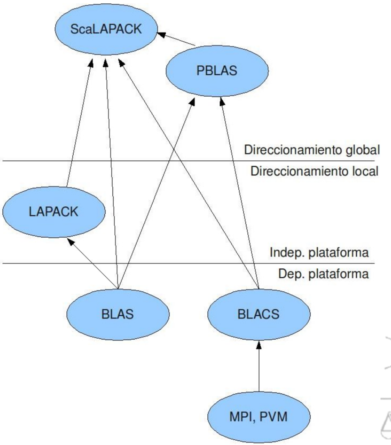
ScaLAPACK

- Estudio de las librerías básicas: BLAS, LAPACK, BLACS, PBLAS.
- *Intel Math Kernel Library*.
- `DGEMM`, `PDPOTRF`, `PDPOTRS` y `PDSYEVX`.



ScaLAPACK

- Estudio de las librerías básicas: BLAS, LAPACK, BLACS, PBLAS.
- *Intel Math Kernel Library*.
- `DGEMM`, `PDPOTRF`, `PDPOTRS` y `PDSYEVX`.
- Se generó una documentación con ejemplos simples para instalar, conectar y utilizar esta librería.



PCSDP

PCSDP

- CSDP: librería/solver (FLOSS) para SDP desarrollada por B. Borchers, U. New Mexico ([Bor99]).

PCSDP

- CSDP: librería/solver (FLOSS) para SDP desarrollada por B. Borchers, U. New Mexico ([Bor99]).
- PCSDP: librería/solver (FLOSS) basado en CSDP, desarrollado para sistemas de memoria distribuída (Beowulf) por Ivan Ivanov, TU Delft ([IK07]).

PCSDP

- CSDP: librería/solver (FLOSS) para SDP desarrollada por B. Borchers, U. New Mexico ([Bor99]).
- PCSDP: librería/solver (FLOSS) basado en CSDP, desarrollado para sistemas de memoria distribuída (Beowulf) por Ivan Ivanov, TU Delft ([IK07]).
- Utilización de ScaLAPACK para operaciones algebraicas.

PCSDP

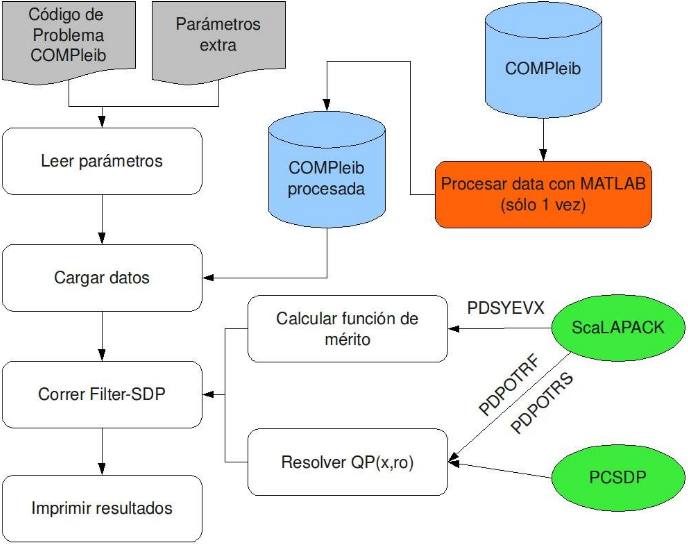
- CSDP: librería/solver (FLOSS) para SDP desarrollada por B. Borchers, U. New Mexico ([Bor99]).
- PCSDP: librería/solver (FLOSS) basado en CSDP, desarrollado para sistemas de memoria distribuída (Beowulf) por Ivan Ivanov, TU Delft ([IK07]).
- Utilización de ScaLAPACK para operaciones algebraicas.
- Claridad en el código, además de reutilización de rutinas.

PCSDP

- CSDP: librería/solver (FLOSS) para SDP desarrollada por B. Borchers, U. New Mexico ([Bor99]).
- PCSDP: librería/solver (FLOSS) basado en CSDP, desarrollado para sistemas de memoria distribuída (Beowulf) por Ivan Ivanov, TU Delft ([IK07]).
- Utilización de ScaLAPACK para operaciones algebraicas.
- Claridad en el código, además de reutilización de rutinas.
- Se utiliza para resolver $QP(x, \rho)$.

Esquema

fnlsdp



Código de Problema COMPleib
Parámetros extra
COMPleib
Leer parámetros
COMPleib procesada
Procesar data con MATLAB (sólo 1 vez)
Cargar datos
Calcular función de mérito
PDSYEVX
ScaLAPACK
PDPOTRF
PDPOTRS
Correr Filter-SDP
Resolver QP(x,ro)
PCSDP
Imprimir resultados

Esquema

Diseño de distintas fases de restauración

Diseño de distintas fases de restauración

- Fase de restauración: encontrar x_k que satisface A1 y B1.

A1 $(\theta(x_k), f(x_k))$ es aceptable para el filtro $\mathcal{F}^{k-1}$.

B1 $QP(x_k, \rho)$ es factible.

Diseño de distintas fases de restauración

- Fase de restauración: encontrar x_k que satisface A1 y B1.

A1 $(\theta(x_k), f(x_k))$ es aceptable para el filtro $\mathcal{F}^{k-1}$.
B1 $QP(x_k, \rho)$ es factible.

- Enfoques estudiados:

Diseño de distintas fases de restauración

- Fase de restauración: encontrar x_k que satisface A1 y B1.

A1 $(\theta(x_k), f(x_k))$ es aceptable para el filtro $\mathcal{F}^{k-1}$.
B1 $QP(x_k, \rho)$ es factible.

- Enfoques estudiados:
  - Enfoque original (implementación MATLAB)

Diseño de distintas fases de restauración

- Fase de restauración: encontrar x_k que satisface A1 y B1.

A1 $(\theta(x_k), f(x_k))$ es aceptable para el filtro $\mathcal{F}^{k-1}$.

B1 $QP(x_k, \rho)$ es factible.

- Enfoques estudiados:
  - Enfoque original (implementación MATLAB)
  - Restauración inexacta

Diseño de distintas fases de restauración

- Fase de restauración: encontrar x_k que satisface A1 y B1.

A1 $(\theta(x_k), f(x_k))$ es aceptable para el filtro $\mathcal{F}^{k-1}$.

B1 $QP(x_k, \rho)$ es factible.

- Enfoques estudiados:
  - Enfoque original (implementación MATLAB)
  - Restauración inexacta
  - Soluciones suboptimales SOF (*Static Output Feedback*)

Diseño de distintas fases de restauración

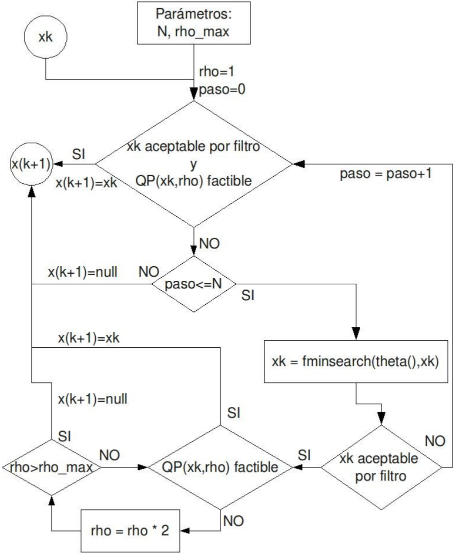
- Fase de restauración: encontrar x_k que satisface A1 y B1.

A1 $(\theta(x_k), f(x_k))$ es aceptable para el filtro $\mathcal{F}^{k-1}$.

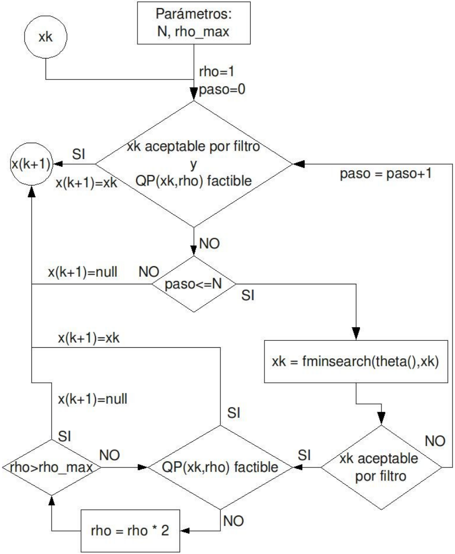
B1 $QP(x_k, \rho)$ es factible.

- Enfoques estudiados:
  - Enfoque original (implementación MATLAB)
  - Restauración inexacta
  - Soluciones suboptimales SOF (*Static Output Feedback*)
  - Posicionamiento de polos

Enfoque original ▸ Ir a pseudocódigo

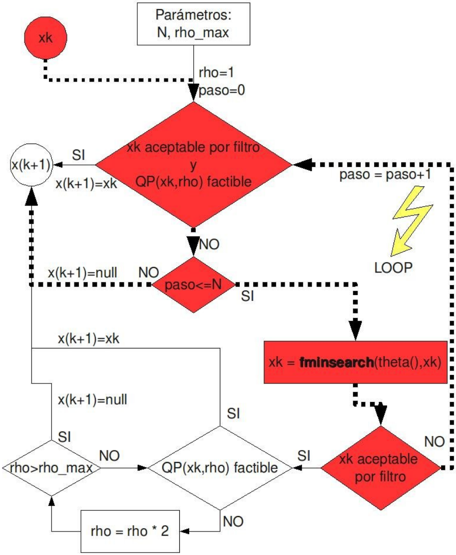


Enfoque original ▸ Ir a pseudocódigo

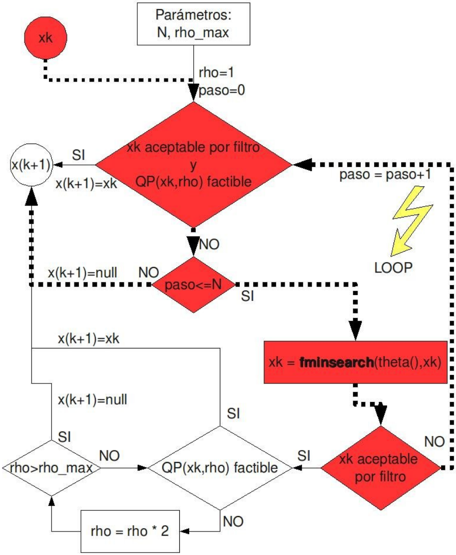


Enfoque original

▸ Ir a pseudocódigo



Enfoque original ▸ Ir a pseudocódigo



Restauración inexacta

Restauración inexacta

- Método desarrollado en [SM08] para programación no lineal, que utiliza 2 fases: factibilidad y optimalidad.

Restauración inexacta

- Método desarrollado en [SM08] para programación no lineal, que utiliza 2 fases: factibilidad y optimalidad.
- Etapa de factibilidad entrega una dirección de descenso para $\theta(x)$.

Restauración inexacta

- Método desarrollado en [SM08] para programación no lineal, que utiliza 2 fases: factibilidad y optimalidad.
- Etapa de factibilidad entrega una dirección de descenso para $\theta(x)$.
- Se adaptó la etapa de factibilidad para nuestro problema $\min\{f(x) : h(x) = 0, G(x) \preceq 0, x \in \mathbb{R}^n\}$:

$$
\begin{array}{lllll}
LP(x_k): & \min\limits_{d \in \mathbb{R}^n} & \sum_{i=1}^{|J_G^+|} \lambda_i (G(x_k) + DG(x_k)d) & & \\
 & & +\sigma \sum_{j \in J} \|h_j(x_k) + Dh_j(x_k)d\|^2 & & \\
 & \text{s.a} & h_j(x_k) + Dh_j(x_k)d & = & 0,\ j \in J^* \\
 & & E^T(G(x_k) + DG(x_k)d)E & \preceq & 0
\end{array}
\tag{12}
$$

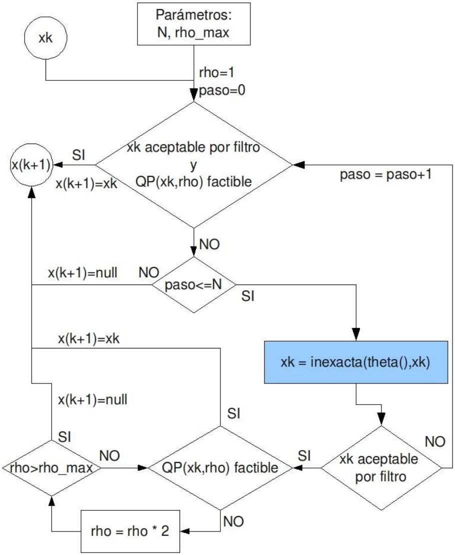
Restauración inexacta

- Método desarrollado en [SM08] para programación no lineal, que utiliza 2 fases: factibilidad y optimalidad.
- Etapa de factibilidad entrega una dirección de descenso para $\theta(x)$.
- Se adaptó la etapa de factibilidad para nuestro problema $\min\{f(x) : h(x) = 0, G(x) \preceq 0, x \in \mathbb{R}^n\}$:

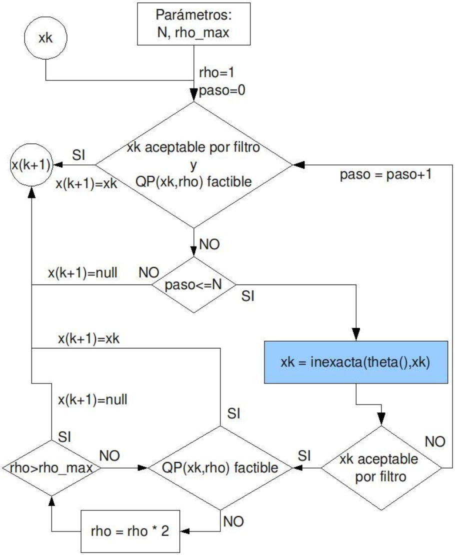
$$
\begin{array}{llllr}
LP(x_k): & \displaystyle\min_{d\in\mathbb{R}^n} & \sum_{i=1}^{|J_G^+|} \lambda_i(G(x_k) + DG(x_k)d) & & \\
& & +\sigma \sum_{j\in J} \|h_j(x_k) + Dh_j(x_k)d\|^2 & & (12) \\
& \text{s.a} & h_j(x_k) + Dh_j(x_k)d & = & 0,\ j \in J^* \\
& & E^T(G(x_k) + DG(x_k)d)E & \preceq & 0
\end{array}
$$

- Este método entrega como resultado un punto $z_k = x_k + \alpha d$, con d solución de $LP(x_k)$ y α el paso de descenso (backtracking), y se interpreta como un punto suficientemente más factible que x_k de tal manera que es aceptable por el filtro.

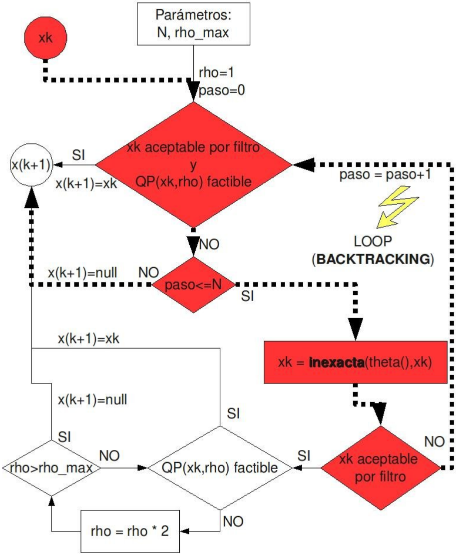
Restauración inexacta ▸ Ir a pseudocódigo



Restauración inexacta ▸ Ir a pseudocódigo

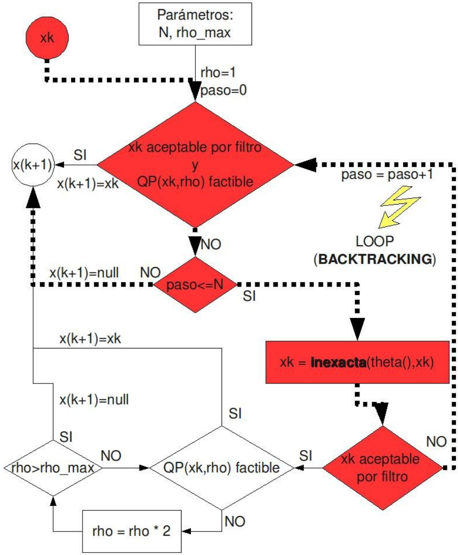


Restauración inexacta ▸ Ir a pseudocódigo



Restauración inexacta

▸ Ir a pseudocódigo



Soluciones suboptimales SOF

Soluciones suboptimales SOF

- En los 2 métodos anteriores se puede caer en una situación de loop que desencadena la detención del algoritmo (fallo en la fase de restauración).

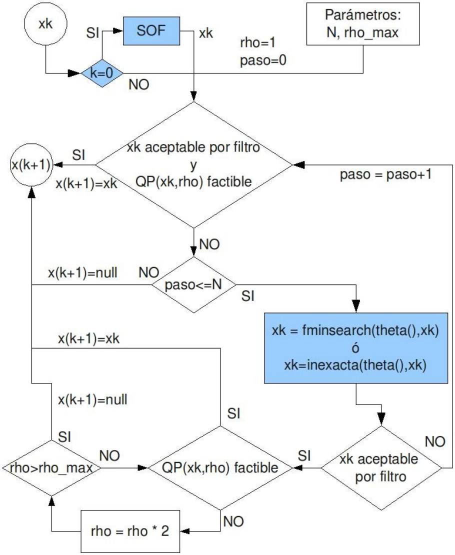
Soluciones suboptimales SOF

- En los 2 métodos anteriores se puede caer en una situación de loop que desencadena la detención del algoritmo (fallo en la fase de restauración).
- Se decidió investigar métodos para generar soluciones suboptimales asociadas al problema aplicado que se intenta resolver (diseño de controles SOF). Esas soluciones se usarán como puntos iniciales en la fase de restauración (se espera que hayan más puntos aceptables por el filtro).

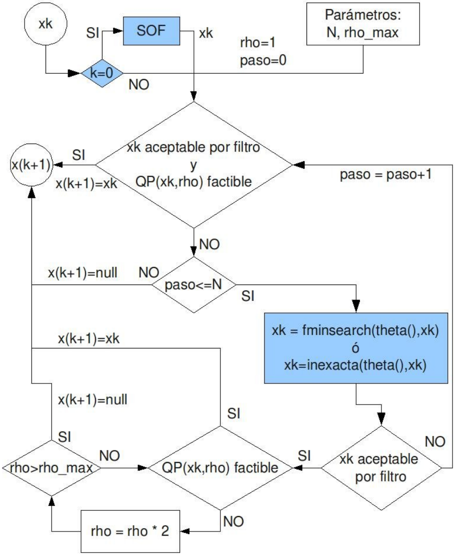
Soluciones suboptimales SOF

- En los 2 métodos anteriores se puede caer en una situación de loop que desencadena la detención del algoritmo (fallo en la fase de restauración).
- Se decidió investigar métodos para generar soluciones suboptimales asociadas al problema aplicado que se intenta resolver (diseño de controles SOF). Esas soluciones se usarán como puntos iniciales en la fase de restauración (se espera que hayan más puntos aceptables por el filtro).
- En [Mos08] se describen métodos para generar soluciones subóptimas para una aplicación simplificada de diseño de controles SOF.

Soluciones suboptimales SOF ▸ Ir a pseudocódigo



Soluciones suboptimales SOF ‣ Ir a pseudocódigo



Posicionamiento de polos

Posicionamiento de polos

- Consiste en un *upgrade* del método de soluciones optimales SOF.

Posicionamiento de polos

- Consiste en un *upgrade* del método de soluciones optimales SOF.
- Utiliza una rutina descrita en [YO07] que permite encontrar una matríz F dado un vector de valores propios λ^D de tal forma que $\lambda(A + BFC) = \lambda^D$.

Posicionamiento de polos

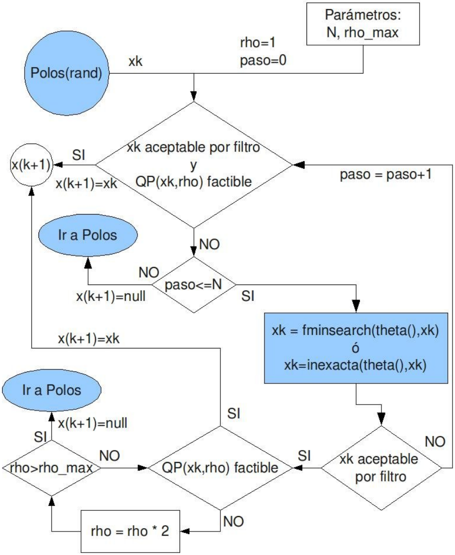
- Consiste en un *upgrade* del método de soluciones optimales SOF.
- Utiliza una rutina descrita en [YO07] que permite encontrar una matríz F dado un vector de valores propios λ^D de tal forma que $\lambda(A + BFC) = \lambda^D$.
- Con esto se pueden generar una serie de nuevos puntos de partida para el algoritmo.

Posicionamiento de polos

- Consiste en un *upgrade* del método de soluciones optimales SOF.
- Utiliza una rutina descrita en [YO07] que permite encontrar una matríz F dado un vector de valores propios λ^D de tal forma que $\lambda(A + BFC) = \lambda^D$.
- Con esto se pueden generar una serie de nuevos puntos de partida para el algoritmo.
- En caso de que falle la fase de restauración, se genera un nuevo punto x_k y se pasa como input nuevamente a esta fase, el número de veces que se defina.

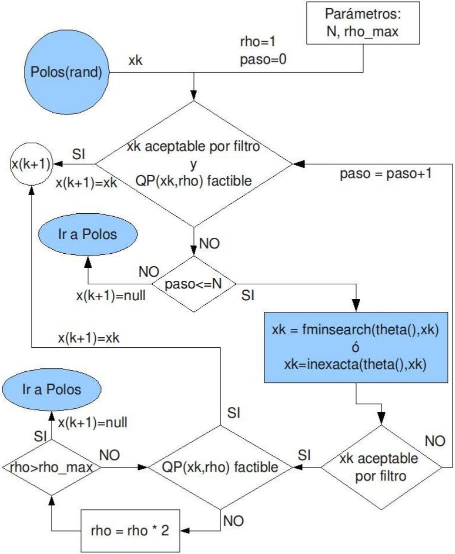
Posicionamiento de polos

Ir a pseudocódigo



Posicionamiento de polos

Ir a pseudocódigo



Esquema

Resultados

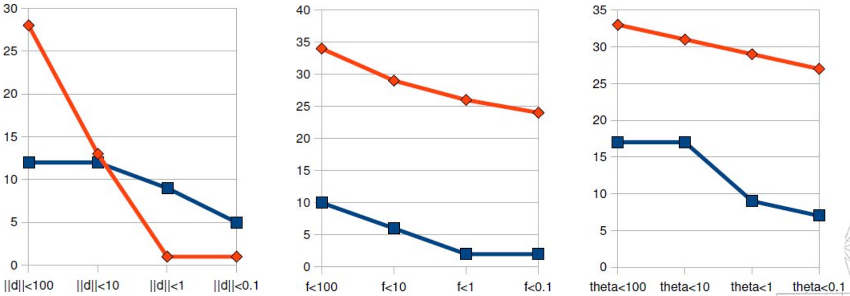
- Comparación `fnlsdp` e implementación MATLAB
- Speedup de `fnlsdp`
- Comparación de fases de restauración

Resultados: `fnlsdp` vs. MATLAB

- 54 problemas ((A, B)-controlables y (A, C)-observables).
- Igual fase de restauración (enfoque original, distintas implementaciones).
- Igual punto inicial (F_0, Q_0, V_0) (generado por método de posicionamiento de polos).
- $\epsilon_{TOL} = 10^{-4}$

Resultados: `fnlsdp` vs. MATLAB

Eje Y: número de problemas que satisfacen $\|d\| < U$ (izquierda), $f(x) < U$ (centro) y $\theta(x) < U$ (derecha).



Resultados: speedup

- Speedup $= \frac{t_{\text{paralelo}}}{t_{\text{secuencial}}}$ (mientras más pequeño, mejor).
- Se escogieron los tests de tamaño *grande* cuyo tiempo de ejecución fuera mayor a 100 segundos (4 de 54 problemas).
- Se realizaron pruebas de speedup para las componentes paralelizadas, de manera independiente (resolución de $QP(x, \rho)$ y cálculo de $\theta(x)$).
- Cluster `leloo`: 4 servidores con 8 procesadores c/u. Sólo se utilizó 1 servidor, pues no están conectados por una red ad-hoc (disminuye la velocidad de transferencia de mensajes y perturba las mediciones).

Resultados: speedup `fnlsdp`

Tiempos de ejecución de `fnlsdp`:

Problema	1 proceso	2 procesos	4 procesos	8 procesos
HE4	102.62	100.97	119.42	1153.21
HE5	138.33	222.38	248.03	858.92
BDT1	234.70	211.07	187.33	436.45
ROC8	795.10	814.02	653.07	2095.69

Speedup de `fnlsdp`:

Problema	1 proceso	2 procesos	4 procesos	8 procesos
HE1	1	0.98	1.16	11.24
HE5	1	1.61	1.79	6.21
BDT1	1	0.89	0.79	1.86
ROC8	1	1.02	0.82	2.63

Resultados: resolución de $QP(x, \rho)$

Speedup de PCSDP:

Problema	m	n	1 proceso	2 procesos	4 procesos	8 procesos
truss4	12	19	1	7.66	8.1	9.71
truss3	27	31	1	6.58	6.9	13.83
qap5	136	26	1	5.55	5.65	6.27
gpp124-1	125	124	1	2.52	2.56	4.33
arch0	174	335	1	0.93	0.81	1.39
gpp250-1	250	250	1	4.27	1.43	1.33
gpp500-1	501	500	1	2.5	0.75	1.37
equalG11	801	801	1	0.78	0.85	0.82
qap10	1021	101	1	1.02	0.84	0.84
control10	1326	150	1	0.7	0.53	0.64
qpG51	1000	2000	1	1.04	1	1.49

Resultados: cálculo de $\theta(x)$

Speedup de `PDSYEVX`:

Problema	Dimensión	1 proceso	2 procesos	4 procesos	8 procesos
sherman1	1000	1	2.16	2.09	2.25
lshp1009	1009	1	2.38	2.3	3.91
rajat02	1960	1	0.77	0.52	0.73
ex14	3251	1	0.61	0.4	0.41
c-26	4307	1	0.6	0.43	0.46
c-30	5321	1	0.64	0.47	0.48
bcsstk17	10974	1	0.63	0.45	0.54

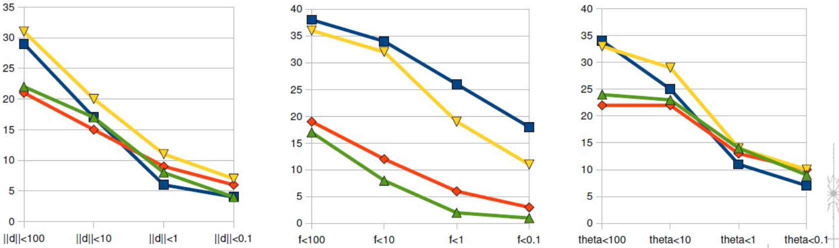
Resultados: fases de restauración

- Se compararon 4 métodos:
  1. Enfoque original + SOF
  2. Enfoque original + posicionamiento de polos
  3. Restauración inexacta + SOF
  4. Restauración inexacta + posicionamiento de polos
- Se escogió la batería de 54 problemas utilizada anteriormente.

Resultados: fases de restauración

Eje Y: número de problemas que satisfacen $\|d\| < U$ (izquierda), $f(x) < U$ (centro) y $\theta(x) < U$ (derecha).

método 1, método 2, método 3, método 4



Esquema

Conclusiones

[a] Enfoque original + SOF subóptimales

[b] Restauración inexacta + posicionamiento de polos

Conclusiones

- Implementación C:

[a] Enfoque original + SOF subóptimales

[b] Restauración inexacta + posicionamiento de polos

Conclusiones

- Implementación C:
  - Mejores resultados que la implementación desarrollada en MATLAB en términos del valor de la función objetivo y de mérito, sin embargo no se puede asegurar nada con respecto a la terminación del algoritmo, pues sólo en 1 caso se llegó a una solución donde $\|dx\| \approx 0$.

[a] Enfoque original + SOF subóptimales

[b] Restauración inexacta + posicionamiento de polos

Conclusiones

- Implementación C:
  - Mejores resultados que la implementación desarrollada en MATLAB en términos del valor de la función objetivo y de mérito, sin embargo no se puede asegurar nada con respecto a la terminación del algoritmo, pues sólo en 1 caso se llegó a una solución donde $\|dx\| \approx 0$.
  - La utilización de cálculo paralelo tiene utilidad sólo si el problema es de grandes dimensiones. Se logró obtener un speedup adecuado en pocos casos de prueba.

[a] Enfoque original + SOF subóptimales

[b] Restauración inexacta + posicionamiento de polos

Conclusiones

- Implementación C:
  - Mejores resultados que la implementación desarrollada en MATLAB en términos del valor de la función objetivo y de mérito, sin embargo no se puede asegurar nada con respecto a la terminación del algoritmo, pues sólo en 1 caso se llegó a una solución donde $\|dx\| \approx 0$.
  - La utilización de cálculo paralelo tiene utilidad sólo si el problema es de grandes dimensiones. Se logró obtener un speedup adecuado en pocos casos de prueba.
- Diseño de fases de restauración:

[a] Enfoque original + SOF subóptimales

[b] Restauración inexacta + posicionamiento de polos

Conclusiones

- Implementación C:
  - Mejores resultados que la implementación desarrollada en MATLAB en términos del valor de la función objetivo y de mérito, sin embargo no se puede asegurar nada con respecto a la terminación del algoritmo, pues sólo en 1 caso se llegó a una solución donde $\|dx\| \approx 0$.
  - La utilización de cálculo paralelo tiene utilidad sólo si el problema es de grandes dimensiones. Se logró obtener un speedup adecuado en pocos casos de prueba.
- Diseño de fases de restauración:
  - Método 1[a] y método 4[b], entregan los mejores rendimientos estadísticos con respecto a la batería de prueba COMPleib. Sin embargo, el método 1 entrega mejores resultados para la función objetivo.

[a] Enfoque original + SOF subóptimales

[b] Restauración inexacta + posicionamiento de polos

Conclusiones

- Implementación C:
  - Mejores resultados que la implementación desarrollada en MATLAB en términos del valor de la función objetivo y de mérito, sin embargo no se puede asegurar nada con respecto a la terminación del algoritmo, pues sólo en 1 caso se llegó a una solución donde $\|dx\| \approx 0$.
  - La utilización de cálculo paralelo tiene utilidad sólo si el problema es de grandes dimensiones. Se logró obtener un speedup adecuado en pocos casos de prueba.
- Diseño de fases de restauración:
  - Método 1[a] y método 4[b], entregan los mejores rendimientos estadísticos con respecto a la batería de prueba COMPleib. Sin embargo, el método 1 entrega mejores resultados para la función objetivo.
  - En varios casos, un método encontró una solución (suboptimal) y los otros no. Conviene tener varias fases de restauración activas simultáneamente.

[a] Enfoque original + SOF subóptimales

[b] Restauración inexacta + posicionamiento de polos

Esquema

Introducción

Antecedentes

- Programación semidefinida lineal (SDP)
- Filter-SDP
- Computación paralela

Trabajo realizado

- Estudio de sistemas de cálculo paralelo
- Implementación en C: `fnlsdp`
- Diseño de distintas fases de restauración

Resultados numéricos

Conclusiones

Trabajo futuro

Trabajo futuro

Trabajo futuro

- Implementación de una fase de restauración en C que garantice la convergencia y buen funcionamiento de la aplicación `fnlsdp`.

Trabajo futuro

- Implementación de una fase de restauración en C que garantice la convergencia y buen funcionamiento de la aplicación `fnlsdp`.
- Diseño de otras fases de restauración, o modificación de las ya existentes.

Trabajo futuro

- Implementación de una fase de restauración en C que garantice la convergencia y buen funcionamiento de la aplicación `fnlsdp`.
- Diseño de otras fases de restauración, o modificación de las ya existentes.
- Depuración de la aplicación `fnlsdp` para generar resultados utilizando problemas más grandes y verificar el speedup obtenido.

Trabajo futuro

- Implementación de una fase de restauración en C que garantice la convergencia y buen funcionamiento de la aplicación `fnlsdp`.
- Diseño de otras fases de restauración, o modificación de las ya existentes.
- Depuración de la aplicación `fnlsdp` para generar resultados utilizando problemas más grandes y verificar el speedup obtenido.
- Utilización de distintos solvers de programación semidefinida o distintas formas de calcular valores propios.

¿Preguntas?

Bibliografía

B. Borchers, *Csdp, a c library for semidefinite programming*, Optimization Methods and Software **11/12** (1999), 613–623.

R. Fletcher, N.I.M. Gould, S. Leyffer, and Ph.L. Toint, *Global convergence of trust region sqp filter algorithms for nonlinear programming*, Tech. report, 99/03, Department of Mathematics, University of Namur, Belgium, 2001.

W. Gómez and H. Ramírez, *A filter algorithm for nonlinear semidefinite programming*, Tech. report, Centro de Modelamiento Matemático, 2006.

I.D. Ivanov and E. de Klerk, *Parallel implementation of a semidefinite programming solver based on csdp in a distributed memory cluster*, Discussion Paper 2007-20, Tilburg University, Center for Economic Research, 2007.

El-S. M.E. Mostafa, *First-order penalty methods for computing suboptimal output feedback controllers*, Appl. and Comput. Math. **7** (2008), no. 1, pp. 66–83.

C. Silva and M. Monteiro, *A filter inexact-restoration method for nonlinear programming*, TOP: An Official Journal of the Spanish Society of Statistics and Operations Research **16** (2008), no. 1, 126–146.

L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Review **38** (1996), no. 1, pp. 49–95.

K. Yang and R. Orsi, *Static output feedback pole placement via a trust region approach*, IEEE Transactions on Automatic Control **52** (2007), no. 11, pp. 2146–2150.

Apéndice

◂ Regresar a Ejemplos SDP

Propiedad (Complemento de Schur)

Para $U = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$ *con* A, C *simétricas y* $A \succ 0$*, se tiene que* $U \succeq 0$ *sí y sólo sí* $C - B^T A^{-1} B \succeq 0$*. A la matríz* $C - B^T A^{-1} B$ *se le llama complemento de Schur de* A *en* U*.*

◂ Regresar a Ejemplos SDP

Propiedad (Criterio de Sylvester)

$A \succeq 0$ *sí y sólo sí todo menor principal (diagonal) de* A *(submatríz que se obtiene como resultado de eliminar filas de índices* I *y columnas de índices* J *a la matríz* A*, con* $I = J$*) es semidefinido positivo.*

Apéndice

◂ Regresar a Ejemplos SDP

Propiedad (Complemento de Schur)

Para $U = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$ *con* A, C *simétricas y* $A \succ 0$, *se tiene que* $U \succeq 0$ *sí y sólo sí* $C - B^T A^{-1} B \succeq 0$. *A la matríz* $C - B^T A^{-1} B$ *se le llama complemento de Schur de* A *en* U.

◂ Regresar a Ejemplos SDP

Propiedad (Criterio de Sylvester)

$A \succeq 0$ *sí y sólo sí todo menor principal (diagonal) de* A *(submatríz que se obtiene como resultado de eliminar filas de índices* I *y columnas de índices* J *a la matríz* A, *con* $I = J$*) es semidefinido positivo.*

◂ Regresar a Filter-SDP

(INICIALIZACIÓN) $k \leftarrow 1$, $\mathcal{F}^0 = \{(u, -\infty)\}$, $d_k \leftarrow \infty^n$, $\beta \in (0,1)$, $\gamma \in (0,\beta)$, $u > 0$, $\sigma \in (0,1)$, $\overline{\rho} > 0$, $\rho_{inicial} > \overline{\rho}$, max_iteraciones > 1.
Mientras $k <$ max_iteraciones **hacer**
    (FASE RESTAURACIÓN) Encontrar x_k y $\rho_{inicial} \geq \tilde{\rho} > \overline{\rho}$ tales que: $(\theta(x_k), f(x_k))$ es aceptable para $\mathcal{F}^{k-1}$ y $QP(x_k, \tilde{\rho})$ es factible.
    $\rho \leftarrow \tilde{\rho}$.
    (PROBLEMA TANGENCIAL) Resolver $QP(x_k, \rho)$.
    Si $\|d_k\| < +\infty$ ($QP(x_k, \rho)$ es factible) **entonces**
        Si $\|d_k\| < \epsilon$ **entonces**
            Fin del algoritmo. Solución: x_k.
        Fin (Si)
        Si $(\theta(x_k + d_k), f(x_k + d_k))$ no es aceptable por $\mathcal{F}^{k-1} \cup \{(\theta(x_k), f(x_k))\}$ **entonces**
            $\rho \leftarrow \frac{\rho}{2}$. Ir a **PROBLEMA TANGENCIAL**.
        De lo contrario,
            Si $\nabla f(x_k)^T d_k + \frac{1}{2} d_k^T B d_k < 0$ y $f(x_k) + \sigma(\nabla f(x_k)^T d_k + \frac{1}{2} d_k^T B d_k) < f(x_k + d_k)$ **entonces**
                $\rho \leftarrow \frac{\rho}{2}$. Ir a **PROBLEMA TANGENCIAL**.
            De lo contrario,
                Si $\nabla f(x_k)^T d_k + \frac{1}{2} d_k^T B d_k \geq 0$ **entonces**
                    $\mathcal{F}^k \leftarrow \mathrm{Add}((\theta(x_k), f(x_k)), \mathcal{F}^{k-1})$ **Iteración tipo** θ
                De lo contrario,
                    $\mathcal{F}^k \leftarrow \mathcal{F}^{k-1}$ **Iteración tipo** f
                Fin (Si)
                $x_{k+1} \leftarrow x_k + d_k$, $k \leftarrow k+1$, $\rho \leftarrow \rho_{inicial}$, Ir a **PROBLEMA TANGENCIAL**.
            Fin (Si)
        Fin (Si)
    De lo contrario,
        $\mathcal{F}^k \leftarrow \mathrm{Add}((\theta(x_k), f(x_k)), \mathcal{F}^{k-1})$ **Iteración tipo** θ
        $k \leftarrow k+1$. Ir a **FASE RESTAURACIÓN**.
    Fin (Si)
Fin (Mientras)

◂ Regresar a Filter-SDP

(INICIALIZACIÓN) $k \leftarrow 1$, $\mathcal{F}^0 = \{(u, -\infty)\}$, $d_k \leftarrow \infty^n$, $\beta \in (0,1)$, $\gamma \in (0,\beta)$, $u > 0$, $\sigma \in (0,1)$, $\overline{\rho} > 0$, $\rho_{inicial} > \overline{\rho}$, max_iteraciones > 1.
Mientras $k <$ max_iteraciones **hacer**
  (FASE RESTAURACIÓN) Encontrar x_k y $\rho_{inicial} \geq \tilde{\rho} > \overline{\rho}$ tales que: $(\theta(x_k), f(x_k))$ es aceptable para $\mathcal{F}^{k-1}$ y $QP(x_k, \tilde{\rho})$ es factible.
  $\rho \leftarrow \tilde{\rho}$.
  (PROBLEMA TANGENCIAL) Resolver $QP(x_k, \rho)$.
  Si $\|d_k\| < +\infty$ ($QP(x_k, \rho)$ es factible) **entonces**
    Si $\|d_k\| < \epsilon$ **entonces**
      Fin del algoritmo. Solución: x_k.
    Fin (Si)
    Si $(\theta(x_k + d_k), f(x_k + d_k))$ no es aceptable por $\mathcal{F}^{k-1} \cup \{(\theta(x_k), f(x_k))\}$ **entonces**
      $\rho \leftarrow \frac{\rho}{2}$. Ir a **PROBLEMA TANGENCIAL**.
    De lo contrario,
      Si $\nabla f(x_k)^T d_k + \frac{1}{2} d_k^T B d_k < 0$ y $f(x_k) + \sigma(\nabla f(x_k)^T d_k + \frac{1}{2} d_k^T B d_k) < f(x_k + d_k)$ **entonces**
        $\rho \leftarrow \frac{\rho}{2}$. Ir a **PROBLEMA TANGENCIAL**.
      De lo contrario,
        Si $\nabla f(x_k)^T d_k + \frac{1}{2} d_k^T B d_k \geq 0$ **entonces**
          $\mathcal{F}^k \leftarrow \mathrm{Add}((\theta(x_k), f(x_k)), \mathcal{F}^{k-1})$ **Iteración tipo** θ
        De lo contrario,
          $\mathcal{F}^k \leftarrow \mathcal{F}^{k-1}$ **Iteración tipo** f
        Fin (Si)
        $x_{k+1} \leftarrow x_k + d_k$, $k \leftarrow k+1$, $\rho \leftarrow \rho_{inicial}$, Ir a **PROBLEMA TANGENCIAL**.
      Fin (Si)
    Fin (Si)
  De lo contrario,
    $\mathcal{F}^k \leftarrow \mathrm{Add}((\theta(x_k), f(x_k)), \mathcal{F}^{k-1})$ **Iteración tipo** θ
    $k \leftarrow k+1$. Ir a **FASE RESTAURACIÓN**.
  Fin (Si)
Fin (Mientras)

Enfoque original

◂ Regresar a Enfoque original

$N \leftarrow$ número de veces que se realiza la búsqueda
$\rho_{\text{max}} \leftarrow$ radio máximo de la región de confianza de $QP(x_k, \rho)$
$\rho \leftarrow (0, \rho_{\text{max}})$
$x_k \leftarrow$ punto inicial o proveniente de la iteración $k-1$
Mientras $((x_k$ no es aceptable para $\mathcal{F}_{k-1}) \vee (QP(x_k, \rho)$ no es factible$))$ $\wedge$ paso $\leq N$ **hacer**
    $x_k \leftarrow \texttt{fminsearch}(\theta(\cdot), x_k)$
    Si x_k es aceptable para $\mathcal{F}_{k-1}$ **entonces**
        Mientras $(QP(x_k, \rho)$ no es factible$) \wedge (\rho < \rho_{\text{max}})$ **hacer**
            $\rho \leftarrow 2 * \rho$
            $d_k \leftarrow QP(x_k, \rho)$ (*si $QP(x_k, \rho)$ no es factible, d_k queda indefinido*)
        Fin (Mientras)
    Fin (Si)
    paso $\leftarrow$ paso $+ 1$
Fin (Mientras)

Enfoque original

◂ Regresar a Enfoque original

$N \leftarrow$ número de veces que se realiza la búsqueda
$\rho_{\max} \leftarrow$ radio máximo de la región de confianza de $QP(x_k, \rho)$
$\rho \leftarrow (0, \rho_{\max})$
$x_k \leftarrow$ punto inicial o proveniente de la iteración $k-1$
Mientras $((x_k$ no es aceptable para $\mathcal{F}_{k-1}) \vee (QP(x_k, \rho)$ no es factible$))$ $\wedge$ paso $\leq N$ **hacer**
    $x_k \leftarrow \texttt{fminsearch}(\theta(\cdot), x_k)$
    Si x_k es aceptable para $\mathcal{F}_{k-1}$ **entonces**
        Mientras $(QP(x_k, \rho)$ no es factible$) \wedge (\rho < \rho_{\max})$ **hacer**
            $\rho \leftarrow 2 * \rho$
            $d_k \leftarrow QP(x_k, \rho)$ *(si $QP(x_k, \rho)$ no es factible, d_k queda indefinido)*
        Fin (Mientras)
    Fin (Si)
    paso $\leftarrow$ paso $+ 1$
Fin (Mientras)

Restauración inexacta

◂ Regresar a R. inexacta

INPUT: $x_k, \mathcal{F}_{k-1}, \beta, \gamma$
$\alpha \leftarrow \pm 1$ (elegir el signo del descenso)
$ac \leftarrow 0$
$\epsilon \leftarrow 10^{-4}$
Contruir problema:

$$\begin{array}{cccc} \displaystyle\min_{t_1,t_2,s\in\mathbb{R},d\in\mathbb{R}^n,Z\in\mathbb{S}^n} & t_1+\sigma t_2 & & \\ \text{s.a} & h_j(x_k)+Dh_j(x_k)d & = & 0,\ j\in J^* \\ & E^T(G(x_k)+DG(x_k)d)E & \preceq & 0 \\ & t_1 - rs - \mathrm{Tr}(Z) & \geq & 0 \\ & Z-(G(x_k)+DG(x_k)d)+sI & \succeq & 0 \\ & t_2 - \|(h_j(x_k)+Dh_j(x_k)d)_{j\in J}\| & \geq & 0 \\ & Z & \succeq & 0 \end{array} \tag{13}$$

para el punto x_k.
Obtener d solución de (13).
Mientras $ac = 0 \wedge |\alpha| \geq \epsilon$ **hacer**
    $z = x_k + \alpha d$
    Si z es aceptable por $\mathcal{F}_{k-1} \wedge \theta(x_k) > \theta(z)$ **entonces**
        $ac \leftarrow 1$
    De lo contrario,

Restauración inexacta

◂ Regresar a R. inexacta

INPUT: $x_k, \mathcal{F}_{k-1}, \beta, \gamma$

$\alpha \leftarrow \pm 1$ (elegir el signo del descenso)

$ac \leftarrow 0$

$\epsilon \leftarrow 10^{-4}$

Contruir problema:

$$
\begin{array}{lrcl}
\min\limits_{t_1,t_2,s\in\mathbb{R},d\in\mathbb{R}^n,Z\in\mathbb{S}^n} & t_1+\sigma t_2 & & \\
\text{s.a} & h_j(x_k)+Dh_j(x_k)d & = & 0,\ j\in J^* \\
 & E^T(G(x_k)+DG(x_k)d)E & \preceq & 0 \\
 & t_1 - rs - \operatorname{Tr}(Z) & \geq & 0 \\
 & Z-(G(x_k)+DG(x_k)d)+sI & \succeq & 0 \\
 & t_2 - \|(h_j(x_k)+Dh_j(x_k)d)_{j\in J}\| & \geq & 0 \\
 & Z & \succeq & 0
\end{array}
\tag{13}
$$

para el punto x_k.

Obtener d solución de (13).

Mientras $ac = 0 \wedge |\alpha| \geq \epsilon$ **hacer**

  $z = x_k + \alpha d$

  Si z es aceptable por $\mathcal{F}_{k-1} \wedge \theta(x_k) > \theta(z)$ **entonces**

    $ac \leftarrow 1$

  De lo contrario,

Soluciones suboptimales SOF

◂ Regresar a SOF

$N \leftarrow$ número de veces que se realiza la búsqueda
$\rho_{\max} \leftarrow$ radio máximo de la región de confianza de $QP(x_k, \rho)$
$\rho \leftarrow (0, \rho_{\max})$
Si $k = 0$ **entonces**
    $x_k \leftarrow \texttt{sof}$
De lo contrario,
    $x_k \leftarrow$ punto inicial o proveniente de la iteración $k-1$
Fin (Si)
Mientras $((x_k$ no es aceptable para $\mathcal{F}_{k-1}) \vee (QP(x_k, \rho)$ no es factible$))$ $\wedge$ paso $\leq N$ **hacer**
    $x_k \leftarrow \texttt{fminsearch}(\theta(\cdot), x_k)$ ó $x_k \leftarrow \texttt{lsdp}(\theta(\cdot), x_k, \mathcal{F}_k, \beta, \gamma)$
    Si x_k es aceptable para $\mathcal{F}_{k-1}$ **entonces**
        Mientras $(QP(x_k, \rho)$ no es factible$) \wedge (\rho < \rho_{\max})$ **hacer**
            $\rho \leftarrow 2 * \rho$
            $d_k \leftarrow QP(x_k, \rho)$ (*si* $QP(x_k, \rho)$ *no es factible,* d_k *queda indefinido*)
        Fin (Mientras)
    Fin (Si)
    paso $\leftarrow$ paso $+ 1$
Fin (Mientras)

Soluciones suboptimales SOF

◂ Regresar a SOF

$N \leftarrow$ número de veces que se realiza la búsqueda
$\rho_{\mathsf{max}} \leftarrow$ radio máximo de la región de confianza de $QP(x_k, \rho)$
$\rho \leftarrow (0, \rho_{\mathsf{max}})$
Si $k = 0$ **entonces**
    $x_k \leftarrow \texttt{sof}$
De lo contrario,
    $x_k \leftarrow$ punto inicial o proveniente de la iteración $k - 1$
Fin (Si)
Mientras $((x_k$ no es aceptable para $\mathcal{F}_{k-1}) \vee (QP(x_k, \rho)$ no es factible$))$ $\wedge$ paso $\leq N$ **hacer**
    $x_k \leftarrow \texttt{fminsearch}(\theta(\cdot), x_k)$ ó $x_k \leftarrow \texttt{lsdp}(\theta(\cdot), x_k, \mathcal{F}_k, \beta, \gamma)$
    Si x_k es aceptable para $\mathcal{F}_{k-1}$ **entonces**
        Mientras $(QP(x_k, \rho)$ no es factible$) \wedge (\rho < \rho_{\mathsf{max}})$ **hacer**
            $\rho \leftarrow 2 * \rho$
            $d_k \leftarrow QP(x_k, \rho)$ (*si $QP(x_k, \rho)$ no es factible, d_k queda indefinido*)
        Fin (Mientras)
    Fin (Si)
    paso $\leftarrow$ paso $+ 1$
Fin (Mientras)

Posicionamiento de polos

◂ Regresar a Polos

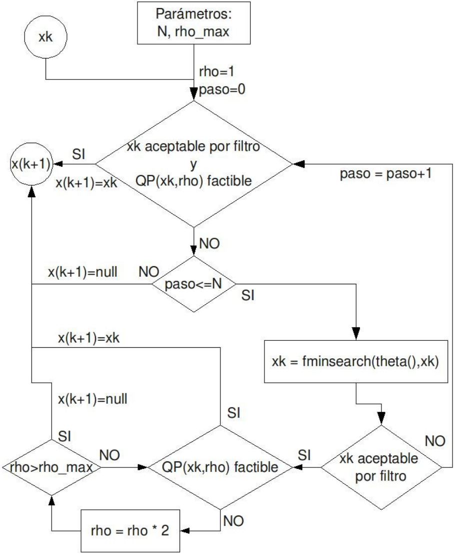
$N \leftarrow$ número de veces que se realiza la búsqueda
$\rho_{\text{max}} \leftarrow$ radio máximo de la región de confianza de $QP(x_k, \rho)$
$\rho \leftarrow (0, \rho_{\text{max}})$
Si $k = 0$ **entonces**
  $x_k \leftarrow \texttt{polos}(k)$
De lo contrario,
  $x_k \leftarrow$ punto inicial o proveniente de la iteración $k - 1$
Fin (Si)
Mientras ((x_k no es aceptable para $\mathcal{F}_{k-1}$) $\vee$ ($QP(x_k, \rho)$ no es factible)) $\wedge$ paso $\leq N$ **hacer**
  $x_k \leftarrow \texttt{fminsearch}(\theta(\cdot), x_k)$ ó $x_k \leftarrow \texttt{lsdp}(\theta(\cdot), x_k, \mathcal{F}_k, \beta, \gamma)$
  Si x_k es aceptable para $\mathcal{F}_{k-1}$ **entonces**
    Mientras ($QP(x_k, \rho)$ no es factible) $\wedge$ ($\rho < \rho_{\text{max}}$) **hacer**
      $\rho \leftarrow 2 * \rho$
      $d_k \leftarrow QP(x_k, \rho)$ (*si* $QP(x_k, \rho)$ *no es factible,* d_k *queda indefinido*)
    Fin (Mientras)
  Fin (Si)
  paso $\leftarrow$ paso $+ 1$
Fin (Mientras)
Si paso $> N$ **entonces**

Posicionamiento de polos

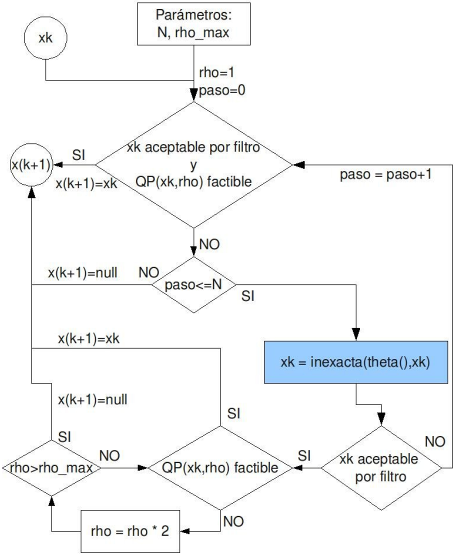
◂ Regresar a Polos

$N \leftarrow$ número de veces que se realiza la búsqueda
$\rho_{\max} \leftarrow$ radio máximo de la región de confianza de $QP(x_k, \rho)$
$\rho \leftarrow (0, \rho_{\max})$
Si $k = 0$ **entonces**
  $x_k \leftarrow \texttt{polos}(k)$
De lo contrario,
  $x_k \leftarrow$ punto inicial o proveniente de la iteración $k - 1$
Fin (Si)
Mientras ((x_k no es aceptable para $\mathcal{F}_{k-1}$) $\vee$ ($QP(x_k, \rho)$ no es factible)) $\wedge$ paso $\leq N$ **hacer**
  $x_k \leftarrow \texttt{fminsearch}(\theta(\cdot), x_k)$ ó $x_k \leftarrow \texttt{lsdp}(\theta(\cdot), x_k, \mathcal{F}_k, \beta, \gamma)$
  Si x_k es aceptable para $\mathcal{F}_{k-1}$ **entonces**
    Mientras ($QP(x_k, \rho)$ no es factible) $\wedge$ ($\rho < \rho_{\max}$) **hacer**
      $\rho \leftarrow 2 * \rho$
      $d_k \leftarrow QP(x_k, \rho)$ (*si* $QP(x_k, \rho)$ *no es factible,* d_k *queda indefinido*)
    Fin (Mientras)
  Fin (Si)
  paso $\leftarrow$ paso $+ 1$
Fin (Mientras)
Si paso $> N$ **entonces**

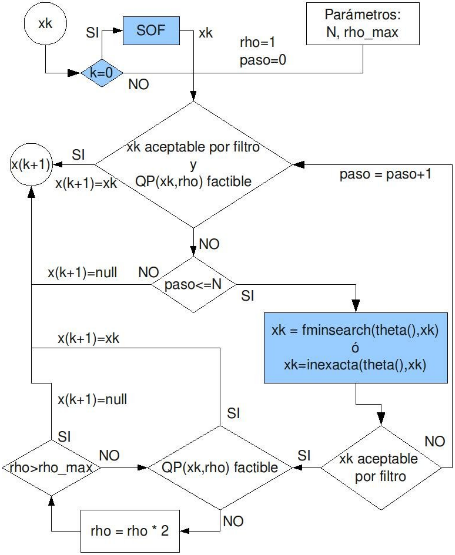
Enfoque original



Restauración inexacta



Soluciones suboptimales SOF



Posicionamiento de polos