Path Following primal-dual interior point method

David Cardoner - Arnau Mercader

En el presente documento se usará Matlab para implementar un algoritmo de tipo *path following*, en concreto el primal-dual. En términos generales, lo que intenta emular este algoritmo es computar (aproximadamente) puntos del camino central para hallar la solución óptima a un problema de optimización lineal (LP). El algoritmo propuesto solo sirve para problemas LP, sin embargo, es posible resolver problemas quadráticos con el algoritmo, modificando algunos parámetros. Para resolver el algoritmo, en cada iteración se usa el método de Newton para resolver nuestra matriz de restricciones derivadas de las condiciones KKT (*Karush Kuhn Tucker*), pero se perturba la restricción asociada a la complementariedad.

Al principio del algoritmo se permutarán las filas y las columnas mediante una heurística por tal de augmentar la *sparsity* de la matriz. En cada iteración, el esfuerzo computacional más complejo recae en el cálculo de la inversa asociada a las ecuaciones normales a resolver, después de la manipulación adecuada de la función a optimizar. Se aplicará la factorización de Cholesky a la matriz *sparse* adecuada con la función *cholinc* (permite 0 en la diagonal). La versión de Matlab que se usa es la 2009a ya que en las más recientes la función se encuentra despreciada.

Un problema general en los métodos de punto interior recae en hallar un punto bien centrado (*well centered*) como solución inicial, punto que puede llegar a ser costoso de encontrar en problemas no triviales. Sin embargo, el algoritmo converge si como punto de partida se usa una solución que cumpla las condiciones mínimas, es decir, $x, s > 0$. En concreto, como punto de partida se usará la siguiente heurística: $x = s = 100 * (max(abs(c))$, donde c es el vector de costes e x,s las variables de decisión del problema en el espacio primal y dual respectivamente.

Para contrastar nuestro algoritmo se usará la heurística *predictor-corrector direction* (propuesta por **Mehrotra**), ya que en general, permite obtener soluciones en la mitad de iteraciones.

Para evaluar el algoritmo llevado a cabo, se utilizarán distintos problemas de la colección Netlib.

Los problemas se pueden obtener mediante el siguiente enlace:

- http://www.cise.ufl.edu/research/sparse/matrices/LPnetlib/index.html

Aspectos relevantes del código

- Como punto de partida se usará la siguiente heurística: $x = s = 100 * (max(abs(c))$
- (1) Valor σ constante e igual a 0.1
- (2) Parámetro ρ en el cálculo del *step length*: max(rhoMin,1-mu), por si nos encontramos muy cerca de la solución objetivo que use el segundo argumento.
- (3) nº máximo de iteraciones: 200
- (4) tolerancia a la solución óptima: tol = 1.e-7, tanto para el error residual como para la estimación de μ, que refleja "lo cerca que estamos" de la solución óptima
- (5) Para calcular el error residual de las restricciones KKT perturbadas se usará la norma de los errores primales, duales y complementarios divididos entre el parámetro bc, que es el máximo entre la norma de b,c + 1.
- (6) Matriz d, se asigna min(maxDiag, x./s), donde maxDiag es un valor muy grande para evitar posibles problemas al usar Cholesky en las ecuaciones normales.
- (7) Al usar Mehrotra: para el corrector central σ el parámetro ρ=1. Para la parte final, donde se usan las 2 direcciones mezcladas se usará la heurística comentada anteriormente (2).

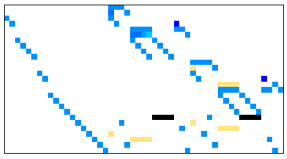
Pruebas en problemas Netlib

Descripción de los problemas

lp_afiro

Características del problema y representación:

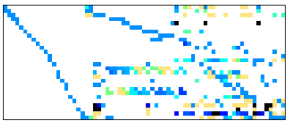
- número de filas (n): 27
- número de columnas (m): 51
- número de elementos distintos de 0: 102
- estructura: rectangular



lp_adlittle

Características del problema y representación:

- número de filas (n): 56
- número de columnas (m): 138
- número de elementos distintos de 0: 424
- estructura: rectangular



lp_osa_14

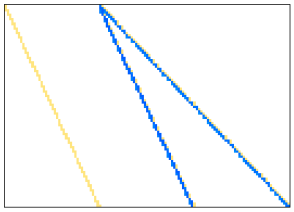
Características del problema y representación:

- número de filas (n): 2337
- número de columnas (m): 54797
- número de elementos distintos de 0: 317097
- estructura: rectangular

lp_stocfor2

Características del problema y representación:

- número de filas (n): 2157
- número de columnas (m): 3045
- número de elementos distintos de 0: 9357
- estructura: rectangular



Resultados obtenidos

A continuación se muestra una tabla resumen de los resultados obtenidos mediante ambos enfoques.

- *ME* define Mehrotra e *IP* primal-dual sin heurística
- El tiempo necesario (CPU_*), está expresado en segundos
- *linprog* define los resultados al usar función de Matlab. La función que se usa dado un problema Netlib es:

```
#Donde Problem es el archivo .MAT que contiene el LP

[x,fval,exitflag,output] = linprog(Problem.aux.c,[],[],
                                   Problem.A,Problem.b,
                                   Problem.aux.lo,Problem.aux.hi);
```

Table 1: Comparative ME and IP

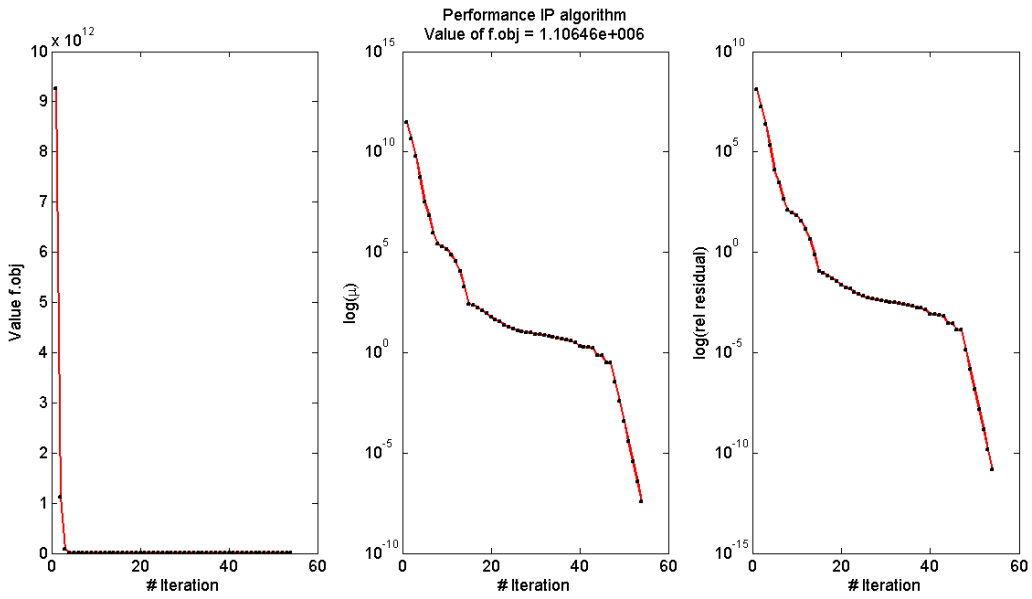
Problem	F_obj	iter_ME	iter_IP	CPU_ME	CPU_IP
lp_afiro	-464.75	13	18	0.0000	0.0000
lp_adlittle	225490.00	21	28	0.0312	0.0624
lp_osa_14	1106460.00	34	54	3.8064	5.7252
lp_stocfor2	-39024.40	27	42	0.4368	0.6084

Table 2: Comparative with linprog

Problem	F_obj	iter_lingprog
lp_afiro	-464.75	7
lp_adlittle	225490.00	12
lp_osa_14	1106500.00	39
lp_stocfor2	-39024.00	22

Output gráfico

La función implementada permite una representación gráfica del valor de la función objetivo, μ y el error residual en cada iteración. Para obtener el siguiente output, se debe escribir $'Y'$ en el *prompt* que proporciona la función. A continuación se muestra dicho output para el problema con más dimensión que se ha probado, lp_osa_14, aplicando heurística.



Comentarios finales

La idea de poder aplicar la teoría en este algoritmo nos ha proporcionado una herramienta que a la vista de los resultados parece bastante razonable. El número de iteraciones que se obtienen usando la heurística son muy parecidas y en algun caso inferiores a las que se obtienen mediante la función *lingprog*, que en todos los casos ha usado como solver un método de punto interior, al considerar el problema del tipo *large scale*. Por lo que hace a la hipótesis inicial sobre las iteraciones, podemos decir que en los problemas testeados usar la heurística provoca que las iteraciones sean aproximadamente la mitad respecto a no usarla.

En algunos problemas, en las iteraciones finales Matlab devuelve un *warning* al aplicar la descomposición *sparse* de Cholesky.

Como crítica y aspectos a mejorar:

- se tendría que adaptar el código para que pudiera proporcionar soluciones para problemas cuadráticos así como también soluciones en problemas con variables enteras o binarias.
- Estudiar alguna heurística para determinar el valor de σ
- Estudiar heurísticas para encontrar puntos inicial más *well centered*, y ver si el número de iteraciones refleja el esfuerzo de encontrar dichos puntos.

Por lo que hace a la función de Matlab, se han cogido ideas del material del campus, así como referencias vía internet. Como último, a continuación se detallan los inputs de las dos funciones usadas.

Función principal

```
"function [x,y,f,hist] = path_corrected(A,b,c,option)
% Algorithm Primal/Dual IP
% Description of problem:
% PRIMAL: min c'x s.t. Ax=b, x>=0,
% DUAL:   max b'y  s.t. A'y+s=c,  s>=0%
%
% input: A (mxn) matrix (in format sparse to apply cholesky)
%        b rhs vector
```

```
%        c cost vector
%
%        option  if is >= 1 perform heuristic Mehrotra, else perform
%        primal dual with constant sigma = 0.1
%        by deffect: algorithm perform Mehrotra
%
% output: x primal vars solution
%         y dual vars solution
%         f optimal objective value c'x = b'y (because of LP)
%
% internal parameters:
%         itmax max iterations
%         tol convergence tolerance
%         maxDiag element for the X^{-1}S matrix to avoid errors
%         rhoMin minimum value of the step length"
```

Función auxiliar

```
"function [alpha, alphax, alphas] = steplength(x, s, Dirx, Dirs, rho)
%
% We need x + alphax*Dirx>0
%         s + alphas*Dirs>0

% rho indicates the maximum fraction of
% step to the boundary (value close to 1,
% we fix in parameters rhoMin = 0.995)

% Calculate the ratio for x,s variables

  alphax = -1/min(min(Dirx./x),-1); alphax = min(1, rho * alphax);
  alphas = -1/min(min(Dirs./s),-1); alphas = min(1, rho * alphas);
  alpha = min(alphax, alphas);"
```

Referencias

Última consulta 07/06.

- www.stat.cmu.edu/~ryantibs/. . . /17-primal-dual-conic-scribed.pdf
- https://www.youtube.com/watch?v=7CMWdO5dgdQ
- ieor.berkeley.edu/~ilan/ilans_pubs/path_following_1_1989.pdf
- http://www.maths.ed.ac.uk/~prichtar/docs/Wright-Ch5-Path-Following-Algorithms.pdf