Guia de uso

Hola

`Solvers` — Module

Implementacion principal del metodo de conjunto activo

`Main.Solvers.OptimizeResult` — Type

```
OptimizeResult{T<:Real}
```

El el tipo de resultado de aplicar `activeSetMethod`, el algoritmo de optimización del conjunto activo para problemas de optimización cuadrática.

Si `R::OptimizeResult` es el resultado de aplicar el algoritmo del conjunto activo se pueden obtener las interaciones como `R.iters`, el punto óptimo como `R.x_star` y el valor de la función objetivo en el punto como `R.q_star`. También se puede obtener "estilo Matlab".

Examples

```
iters, x_star, q_star, μ = activeSetMethod(G, c, A_E, b_E, A_I, b_I)
```

`Main.Solvers.activeSetMethod` — Function

```
activeSetMethod(G, c, A, b, n_eq,  W_k=nothing, maxiter = 100, atol = 1e-9)
```

Aplica el algoritmo de conjunto activo para optimizar el problema cuadrático:

$$\min \frac{1}{2} x^\top G x + c^\top x$$

Sujeto a

$$A_E x = b_E$$
$$A_I x \leq B_I$$

Arguments

- `G::Matriz{Float64}(n, n)`: Matriz positiva definida de la definición del problema cuadrático.
- `c::Vector{Float64}(n)`: Vector de costos de la funciób objetivo.
- `A::Matrix{Float64}(m, n)`: La matriz de restricciones.
- `b::Vector{Float64}(n)`: Vector de constantes de las restricciones
- `n_eq::Int`: Número de restricciones de igualdad del problema cuadrático.
- `W_k::BitVector(m)`: Opcional. Restricciones a tomar como activas en el primer paso del método.
- `maxiter::Int = 100`: Opcional. Limita el máximo de iteraciones del método.
- `atol::Float64`: Opcional. Determina la distancia máxima a la que puede estar d_k de cero cuando se determina si entrar a rama1.

`Main.Solvers.Utils` — Module

Funciones de apoyo para implementar el algoritmo principal.

`Main.Solvers.Utils.klee_minty` — Method

```
klee_minty(n::Int)
```

Regresa la matriz `G`, `A` y el vector `b` de restricciones dadas por el problema de Klee-Minty descrito en el proyecto.

Arguments

- `n::Int`: Dimensión del problema de Klee-Minty.

Returns

Regresa las matrices `G`, `c`, `A`, `b` en ese orden.

`Main.Solvers.Utils.linprog` — Method

```
linprog(A, b, n_eq)
```

Envuelve JuMP y la rutina de simplex para devolver un punto factible al un problema cuadrático con restricciones de desigualdad con `m` restricciones y `n` variables.

Resuelve el problema

$$\begin{aligned} \min\ & 1^\top x \\ & Ax = b_E \\ & Ax \leq b_I \end{aligned}$$

Arguments

- `A::Matrix(m, n)`: La matriz de restricciones del problema.
- `b::Vector(n_e)`: Vector de restricciones.
- `n_eq::Int`: Número de restricciones de igualdad del problema

n = n*e* + *n*i

`Main.Solvers.Utils.rankMethod` — Method

```
rankMethod(G, A, c, b)
```

Implementación del método de rango para resolver problemas de programación cuadrática (PPC) con restricciones $Ax = b$.

Arguments

- `G::Matriz{Float64}(n, n)`: Matriz positiva definida de la definición del problema cuadrático.
- `A::Matrix{Float64}(m, n)`: La matriz de restricciones.
- `c::Vector{Float64}(n)`: Vector de costos de la funciób objetivo.
- `b::Vector{Float64}(n)`: Vector de constantes de las restricciones

`Main.Solvers.Utils.solve2_11` — Method

```
solve2_11(g_k, A, W_k, n_eq)
```

Resuelve el sistema lineal del problema (2.11) de las notas.

$$\sum_{i \in \mathcal{E}} \widehat{\lambda_i} a_i + \sum_{i \in \widehat{W} \cap \mathcal{I}} \widehat{\mu_i} a_i = -g_k$$

Arguments

- `g_k::Vector(n)`: `G * x_k + c`
- `A::Matrix(m, n)`: La matriz de restricciones del problema.
- `W_k::BitVector(m)`: Conjunto de restricciones activas en el punto actual.
- `n_eq::Int`: Número de restricciones de igualdad del problema cuadrático.

`Main.Solvers.Utils.solve2_8` — Method

```
solve2_8(G, A_k, g_k)
```

Envoltorio para el metodo del rango para resolucion de problemas cuadraticos con igualdades.

Arguments

- `G::Matriz{Float64}(n, n)`: Matriz positiva definida de la definición del problema cuadrático.
- `A_k::Matrix{Float64}(m, n)`: La matriz de restricciones de W_k.
- `g_k::Vector{Float64}(n)`: g_k del algoritmo de conjunto activo

`Main.Solvers.Utils.solve2_9` — Function

```
solve2_9(A, b, x_k, d_k, atol=1e-12)
```

Resuelve el problema (2.9) de las notas con tolerancia absoluta `atol`.

$$\breve{\alpha} = \min_{\substack{i \notin W_0 \\ a_i^\top b_0 > 0}} \left(\frac{b_i - a_i^\top x_0}{a_i^\top d_0} \right)$$

Arguments

- `A::Matrix(m, n)`: La matriz de restricciones del problema.
- `b::Vector(n)`: Vector de restricciones.

- `x_k::Vector(n)`: Punto actual del método del conjunto activo.
- `d_k::Vector(n)`: Dirección de descenso calculada con `solve2_8`
- `atol::Float64`: Tolerancia absoluta (opcional).

`Main.Solvers.Utils.𝒜` — Method

```
𝒜(A, b, x)
```

Regresa los indices de las restricciones de activas (de igualdad & desigualdad)

Arguments

- `A::Matrix(m, n)`: La matriz de restricciones del problema.
- `b::Vector(n)`: Vector de restricciones.
- `x::Vector(m)`: Punto donde se evalua la restriccion.