Prueba

Juanito Alonso Rebe

Implementación

El algoritmo Simplex modificado que decidimos implementar es el conocido como el método de la gran M. Con esta implementación podemos resolver cualquier Problema de Programación Lineal (PPL) bien definido, con el origen incluido o exluido del polítopo factible. Elegimos el método de la gran M en vez de las dos fases por su facilidad de implementación, y dado que al ser una implementación computacional, manipular una M considerablemente grande no afecta la velocidad de ejecución puesto que en el estándar IEEE de tipo de datos *float*, la multiplicación entre números toma el mismo tiempo sin importar el tamaño del número, lo cual nos permitió elegir M suficientemente grande para asegurar que el método fuese exitoso.

Para implementar el algoritmo Simplex dividimos el algoritmo en pequeñas subrutinas, cada una independiente de las demás, siguiendo la filosofía de diseño Unix. Para referirnos a cada una de estas subrutinas, escribimos su nombre en `este formato` para acentuar su papel como programas.

Cabe mencionar que por brevedad, reducimos la cantidad de detalles presentados en el pseudocódigo que mostramos a continuación. Para más detalles, se puede consultar el código que se entrega con el proyecto.

Pivoteo

Para el proceso básico de identificar pivotes según la regla de Bland, y después hacer un uno principal en el pivote identificado sobre la tabla, usamos dos subrutinas: una que encuentra el pivote y otra que hace la operación de pivoteo.

Algorithm 1: GA(n, χ, μ)

Result: individuo más apto de P_k

Inicializamos generación 0;
$k := 0$
$P_k :=$ población de n individuos generados al azar;
Evaluar P_k :
do
  Crear generación $k + 1$;
  1. Copia:;
  Seleccionar $(1 - \chi) \times n$ miembros de P_k e insertar en P_{k+1}
  2. Cruce $k + 1$;
  Seleccionar $\chi \times n$ miembros de P_k; emparejarlos; producir descendencia; insertar la descendencia en P_{k+1}
  3. Mutar:;
  Seleccionar $\mu \times n$ miembros de P_{k+1}; invertir bits seleccionados al azar
  Evaluar P_{k+1};
  Calcular $fitness(i)$ para cada $i \in P_k$
  Incrementar: $k := k + 1$;
while *el fitness del individuo más apto en* P_k *no sea lo suficientemente bueno*;

El algoritmo de pivoteo está inspirado en la idea de una factorización LU "incompleta". Calcular la factorización LU equivale a encontrar una matriz L que encodifica todo el proceso de pivoteo del Gauss-Jordan. Usamos una estrategia similar pero para un solo pivote.

Como se puede ver, no hay ningúna estructura de repetición en `pivotea`. Aprovechamos las propiedades de las matrices elementales y las optimizaciones del lenguaje MATLAB para hacer el algoritmo lo más rápido y eficiente posible al crear unos principales en un solo paso en vez de buscar entrada por entrada candidatos y hacer divisiones renglón por renglón[ˆ1].

Algoritmo Simplex

Con la subrutina `Simplexealo` implementamos el proceso de repetición del pivoteo hasta que se encuentra la tabla final, y además checamos las condiciones de terminación que nos permiten saber si el algoritmo Simplex terminó con soluciones óptimas, degeneradas, o si el polítopo factible del problema no está acotado.