**Nama** : Felix Fernando

**NIM** : 10120082

**Tugas Kapita Selekta : Matematika Terapan – Branch and Bound Method**

Algoritma branch and bound ini dibuat dengan menggunakan bahasa pemrograman Python dengan beberapa library tambahan, yaitu

1. Numpy (untuk pendifinisian matrix)
2. Scipy (untuk melakukan metode simplex dalam penyelesaian LP relaxation)

**Fungsi:**

Algoritma ini akan dibagi dalam dua buah fungsi, yaitu fungsi is\_integer dan branch\_and\_bound,

1. Fungsi **is\_integer**

Fungsi ini berfungsi untuk menetukan apakah nilai yang dimasukkan sebagai parameter adalah anggota dari bilangan bulat.

**Code:**

|  |
| --- |
| def *is\_integer*(*n*):  if int(n) == n:  return True  else:  return False |

2. Fungsi **branch\_and\_bound**

Fungsi rekursi yang berfungsi untuk menyelesaikan permasalahan integer optimization dengan meggunakan algoritma **branch and bound.**

**Code:**

Pendefinisian fungsi branch\_and\_bound

|  |
| --- |
| def *branch\_and\_bound*(*c*, *A*, *b*, *integer\_var*, *lb*, *ub*, *isMax* =False, *depth*=1): |

Menyelesaikan LP relaxation dari sebuah masalah optimisasi bilangan bulat dan menyimpan hasil optimalnya dalam variable x\_candidate dan f\_canditate.

|  |
| --- |
| *# Optimal solution for LP Relaxation*  if(isMax):  optimized = scipy.optimize.*linprog*(-c, A, b, *method*="highs", *bounds*=list(zip(lb, ub)))  else:  optimized = scipy.optimize.*linprog*(c, A, b, *method*="highs", *bounds*=list(zip(lb, ub)))  x\_candidate = optimized.x  f\_candidate = optimized.fun |

Menentukan apakah hasil dari setiap variable sudah merupakan anggota dari bilangan bulat.

|  |
| --- |
| var\_constraint\_fulfilled = True  if(f\_candidate == None):  if(isMax):  return [], -np.inf, depth  else:  return [], np.inf, depth    for idx, bool in enumerate(integer\_var):  if(bool and not *is\_integer*(*x\_candidate*[*idx*])):  var\_constraint\_fulfilled = False  break  else:  var\_constraint\_fulfilled = True |

Melakukan proses rekursi dengan metode **Depth First Search (DFS)** untuk melakukan **branch and bound** sehingga didapatkan hasil optimal yang merupakan anggota dari bilangan bulat.

|  |
| --- |
| if(var\_constraint\_fulfilled):  if(isMax):  return x\_candidate, -f\_candidate, depth  else:  return x\_candidate, f\_candidate, depth  else:  left\_lb = np.*copy*(lb)  left\_ub = np.*copy*(ub)  right\_lb = np.*copy*(lb)  right\_ub = np.*copy*(ub)  max\_coeff\_idx = -1  for idx, val in enumerate(integer\_var):  if(val and not *is\_integer*(*x\_candidate*[*idx*])):  if(max\_coeff\_idx == -1):  max\_coeff\_idx = idx  elif(*c*[*max\_coeff\_idx*] < *c*[*idx*]):  max\_coeff\_idx = idx  *left\_ub*[*max\_coeff\_idx*] = np.*floor*(*x\_candidate*[*max\_coeff\_idx*])  *right\_lb*[*max\_coeff\_idx*] = np.*ceil*(*x\_candidate*[*max\_coeff\_idx*])  x\_left, f\_left, depth\_left = *branch\_and\_bound*(c, A, b, integer\_var, left\_lb, left\_ub, isMax, depth + 1)  x\_right, f\_right, depth\_right = *branch\_and\_bound*(c, A, b, integer\_var, right\_lb, right\_ub, isMax, depth + 1)  if(isMax):  if(f\_left > f\_right):  return x\_left, f\_left, depth\_left  else:  return x\_right, f\_right, depth\_right  else:  if(f\_left < f\_right):  return x\_left, f\_left, depth\_left  else:  return x\_right, f\_right, depth\_right |

Dengan menggunakan kedua fungsi tersebut, algoritma **branch and bound** pada permasalahan optimisasi bilangan bulat dapat dilakukan dengan cara memanggil fungsi branch\_and\_bound yang memuat beberapa parameter, yaitu sebagai berikut:

* **c**: Koefisien dari fungsi objektif dengan tipe data numpy array

ex: np.array([4, 3, 1])

* **A**: Koefisien dari kendala dengan tipe data numpy array 2D

ex: np.array([[3, 2, 1], [2, 1, 2])

* **b**: RHS dari masing-masing kendala dengan tipe data numpy array

ex: np.array([7, 11])

* **integer\_var**: Variable target yang merupakan anggota dari bilangan bulat dengan tipe data array dari Boolean

ex: [False, True, True] -> hal ini mengatakan bahwa

* **lb**: Batas bawah dari masing-masing variable target dengan tipe data array

ex: [0, 0, 0] -> hal ini mengatakan bahwa

* **ub**: Batas atas dari masing-masing variable target dengan tipe data array

ex: [1, 2, 3] -> hal ini mengatakan bahwa

* **isMax**: Boolean yang menetukan apakah masalah fungsi objektif-nya adalah permasalah memaksimumkan
* **depth**: Kedalaman dari pohon branch and bound method, dengan nilai defaultnya adalah 1

**Pengujian Code:**

Algoritma diatas akan digunakan untuk menyelesaikan permasalahan optimisasi bilangan bulat sebagai berikut,

Dengan kendala

dan dengan

dengan ini yang akan dimasukkan kedalam parameter fungsi **branch\_and\_bound** adalah

Atau dalam code Python, dituliskan sebagai,

|  |
| --- |
| c = np.*array*([4, 3, 1])  A = np.*array*([[3, 2, 1], [2, 1, 2]])  b = np.*array*([7, 11])  integer\_var = [False, True, True]  lb = [0, 0, 0]  ub = [None, None, None] |

Dengan ini maka dengan melakukan variable deconstructing pada pemanggilan fungsi **branch\_and\_bound**,

|  |
| --- |
| x\_optimal, f\_optimal, depth\_optimal = *branch\_and\_bound*(c, A, b, integer\_var, lb, ub, True) |

Didapatkan hasil optimal sebagai berikut,

A screenshot of a computer

Description automatically generated

atau