1 Introduction

This package is a Python project to enumerate all nondominated points for instances of multiobjective integer and mixed integer programs with any arbitrary number of objective functions.

To compile the code, you must have Python3, Pyomo, Pulp, and Couenne (MINLP solver) installed on your machine.

2 Data Files

It is assumed the RVF algorithm solves the multiobjective problem as

$$\text{vmin}_{(x_I,x_C)\in X_{\text{MO}}} C_I x_I + C_C x_C,$$

where

$$X_{\text{MO}} = \{(x_I, x_C) \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ : A_I x_I + A_C x_C = b\},$$

is the feasible region; $A \in \mathbb{Q}^{m\times n}$ is the matrix of coefficients of the constraints; $b \in \mathbb{Q}^m$ is the right-hand side (RHS) of the constraints; and the rows of matrix $C \in \mathbb{Q}^{(l+1)\times n}$ are the multiple objectives of the problem. The vmin operator indicates that this is a vector minimization (multiobjective) problem. A_I and C_I are the submatrices of A and C consisting of columns associated with the integer variables (indexed by set $I = \{0, \ldots, r-1\}$), while A_C and C_C are the submatrices corresponding to the columns associated with the continuous variables (indexed by set $C = \{r, \ldots, n-1\}$).

It is assumed that each objective function must be minimized. If you have a maximization problem, you need to multiply all objective function coefficients by -1.

You need to prepare your input as follows. "OBJ" is a list of the first objective (the one in the objective function, i.e., c^0 in the RVF paper) coefficients. "Mat" is a dictionary for the coefficients of the objectives considered in the constraint set. "MatFixed" is a dictionary for the coefficients of the left-hand side of the constraints with the fixed right-hand sides. "RHS" is a dictionary containing the RHS of the constraints with the fixed right-hand sides.

For the mixed integer cases, if you have $A_I x_I + A_C x_C \leq b$ in your problem structure, you should convert it to equality constraints by adding slack variables (incorporated in the code). Similarly, for the pure integer case, if you have $A_I x_I \leq b$, you should add slack, but before that, make sure that the coefficients of the matrix A and the RHS vector b have integer values since the slack variable in the pure integer case is considered integer in the code. There is a "debug_print" parameter that you can set to "True" if you want the RVF algorithm to output the value of all variables.

For the linear version of the IP cases, you need to set the parameters "M" and "eps" to sufficiently large and small positive values, respectively.

3 Running

For running the project, we should go to the directory in which the project is saved and run the Python files simply by typing "python3 RVF_IP.py" for the nonlinear version of the pure integer programs, "python3 RVF_IP_LinearVersion.py" for the linear version of the pure integer programs, and "python3 RVF_MILP.py" for the mixed integer programs.

4 Results

When the problem is solved successfully, the phrase "Finished!" will be shown. The RVF algorithm will output the results in "Results_IP.txt" (nonlinear version of IP cases), "Results_IP_Linear" (linear version of IP cases), and "Results_MILP.txt" (MILP cases). All the NDPs are shown in a list of lists format. The running time is also shown as "Elapsed Time" in seconds. Moreover, note that the maximum runtime of the code is set to 86400 seconds. If the RVF algorithm cannot run the instance within the time limit, it will output "Finished due to time limit!" and the approximate efficient frontier will be shown in the result text file. If the "debug_print" parameter is set to True, the value of the variables will be logged in "Results_IP_details", "Results_IP_Linear_details", and "Results_MILP_details".