1 Introduction

This project is a Netbeans IDE 7.3.1 C++ project which is written in Linux (Ubuntu). It should also be mentioned that the code is only able to solve instances of multi-objective integer programs (with an arbitrary number of objective functions) when all objective function coefficients are *integer*. As a consequence, it is not currently possible to solve instances with continuous objective function coefficients (please contact authors if you wish to solve such an instance). All source files except main.cpp exist in the *Sources* folder and all header files exist in *Headers* folder. You may use the code

- to enumerate all nondominated points; or
- to optimize a linear function over the set of efficient solutions (to an arbitrary optimality gap); or
- to compute the Nadir point; or
- to compute an arbitrary component of the Nadir point.

To compile the code, CPLEX (default version 12.6) must be installed on your computer. The default directory for CPLEX used is */opt /nuor/ilog/*. Changing the directory of CPLEX to your preferred directory can be done either in the Makefile or through Netbeans. If you would like to do it in the Makefile you should go to *nbproject/Makefile-Debug.mk* and *nbproject/Makefile-Release.mk* and change all instances of */opt /nuor/ilog/* to your preferred directory. If you would like to do it through Netbeans, you can open the project in Netbeans and right click on the name of the project and choose *Properties*. You should then change the directory in the *Include Directories* box which is located in the *C++ Compiler* sub-menu. Moreover, you should also change the directory in the *Libraries* box which is located in the *Linker* sub-menu.

2 Data Files

The instances (with three objective functions and $f(x) = -\sum_{j=1}^{p} z_j^a(x)$) used in the computational experiments discussed in the paper are available in the *instances* folder. As is mentioned in the paper, it is assumed that each objective function, i.e., $z_j(x)$ for each $i = 1, \cdots, p$, and the linear function, i.e., $f(x)$, must be *minimized*. Each data file should be written as a CPLEX LP file as follows,

$$
\begin{aligned}
&\min f(x) && \\
&z_j^a = 0 && \forall j \in \{1, \ldots, p\}, \\
&z_j^a = z_j(x) && \forall j \in \{1, \ldots, p\}, \\
&x \in \mathcal{X}, && \\
&z_j^a \in \mathbb{R} && \forall j \in \{1, \ldots, p\}.
\end{aligned}
$$

Note that if you want to generate all nondominated points, or the Nadir point, and/or any component of the Nadir point then you do not need $f(x)$. So, you can write any arbitrary objective function as $f(x)$, and the code will remove it. Moreover, in some cases, it may be more efficient to write $z_j^a \in \mathbb{Z}_{\geq}$ in LP file rather than $z_j^a \in \mathbb{R}$ for $j = 1, \cdots, p$ (if it is possible).

3 Compiling and Running

Compiling the project can be done by going to the directory in which the project is saved and typing "make clean" followed by "make" (you can also compile through Netbeans).

- If you want to compute all nondominated points then after compiling the project, an instance can be solved by typing

  ./DCM <address>/<instance><number_of_objectives><0><1>

  for example

  ./DCM instances/AP5.lp 3 0 1;

- If you want to optimize a linear function over the set of efficient solutions then after compiling the project, an instance can be solved by typing

  ./DCM <address>/<instance><number_of_objectives><1><1><relative_optimality_gap>

  for example

  ./DCM instances/AP5.lp 3 1 1 0.000001;

- If you want to compute the Nadir point then after compiling the project, an instance can be solved by typing

  ./DCM <address>/<instance><number_of_objectives><2><1><relative_optimality_gap>

  for example

  ./DCM instances/AP5.lp 3 2 1 0.000001;

- If you want to compute an arbitrary component of the Nadir point then after compiling the project, an instance can be solved by typing

  ./DCM <address>/<instance><number_of_objectives><3><1><relative_optimality_gap><Nadir_point_component>

  for example

  ./DCM instances/AP5.lp 3 3 1 0.000001 2.

  Note that the Nadir_point_component is $j-1$ where $j \in \{1, \cdots, p\}$.

4 Results

After solving an instance completely, the phrase "Finish!!!!" will be shown in your screen. The algorithm will output the obtained results (almost) every 300 seconds (unless CPLEX struggles to solve single-objective IPs). Detailed results can be found in *Results.txt*. It is clear that before showing the phrase "Finish!!!!" in your screen, the outputted results are not complete. Moreover, note the maximum runtime of the code is set to 86400 seconds. To change these parameters, refer to *Headers/extern_parameters.h*.