An application of IP to software engineering

Worst-Case Execution Time of a program
Worst-Case Execution Time

- Worst-case execution time (WCET) of a program: maximum time to run a program when the input changes.

- Its knowledge is useful for the development of real-time systems: for example, a computer that controls the operation of a vehicle needs to respond to an input within a given lapse of time.

- It is usually difficult to exactly evaluate it or to estimate: you cannot always evaluate it simply by running the program for any possible input, and it also depends in part on the platform.
The worst-case execution time can be estimated (upper bounded) by using the Control Flow Graph and Integer Linear Programming.

Control flow graph: representation, using graph notation, of all paths that might be traversed through a program during its execution.
x=1; y=0; z=0;  
while x<10:  
   if x>5:  
      y=y+x  
   else  
      z=z+x  
   x=x+1  
print('...')

• The control flow graph nodes represent blocks (maximal) of consecutive instructions, not interrupted by control instructions (while, if, etc.)
• Consecutively executable blocks are connected by oriented edges.
• The labeled edges correspond to the different possible outcomes of the execution of a control statement.
Control flow graph

if \( x > 0 \):
    \( y = 1 \)
else
    \( y = 2 \)

if \( x > 1 \):
    \( z = 10 \)
else
    \( z = 20 \)

... 

- The path actually followed and the overall execution time of the program depend on the input.
- Not all paths are possible ("feasible"): indeed, in the example in the figure, if \( x > 0 \) is, then, \( x > 1 \) is false too.
IP can be used to provide an upper bound of the worst-case execution time of a program, taking into account the resulting information from the structure of the control flow graph and the knowledge of infeasible paths in the control flow graph.

In order to apply the IP, it is necessary to identify decision variables, constraints and objective function for the specific problem.
We introduce the following decision variables:

1. A decision variable $N_i$ for every block $i$ of instructions.
   - $N_i$ represents the number of times the block $i$ is executed.

2. A decision variable $E_{i,j}$ for each pair of blocks $(i,j)$ connected by an oriented arc in the control flow graph.
   - $E_{i,j}$ represents the number of times the arc $(i,j)$ is traveled during the execution of the program.

All the decision variables are non-negative integers.
The control flow graph provides constraints that link decision variables. These constraints are due to the structure of the programme. Example:

```
Block
```

- The number of times that the block in the figure is executed is equal to the number of times that incoming arcs are travelled, and also equal to the number of times that the outgoing arcs are traversed.
- The identified constraint does not depend on the particular input of the program.
Objective function

- The single execution of the block \( i \) of instructions takes time of execution equal to \( c_i \) (it is known).
- If the block \( i \) is executed \( N_i \) times, then the total runtime of that block is \( c_i \cdot N_i \).
- For \( N \) blocks of instructions, the objective function is:

\[
z = \sum_{i=1}^{N} c_i N_i = \text{“execution time of the program”}
\]

- The objective function must be maximized, as we are interested in the determination of the execution time (or an upper bound) in the worst-case scenario.
sum=0
for i in range(10):
    if i%2==0:
        sum+=1
    if sum<0:
        break
return sum

- % is the modulo operation (it finds the remainder after division).
Associated Integer Linear Programming Problem

\[
\max \ \text{Time} = c_1 N_1 + c_2 N_2 + c_3 N_3 + c_4 N_4 + c_5 N_5 + c_6 N_6 + c_7 N_7 + c_8 N_8
\]

such that

\[
\begin{align*}
1 &= N_1 = E_{12} \\
E_{62} + E_{12} &= N_2 = E_{23} + E_{27} \\
E_{23} &= N_3 = E_{34} + E_{35} \\
E_{34} &= N_4 = E_{45} \\
E_{35} + E_{45} &= N_5 = E_{56} + E_{58} \\
E_{56} &= N_6 = E_{67} \\
E_{58} &= N_8 = E_{87} \\
E_{87} + E_{27} &= N_7 = 1 \\
E_{62} &\leq 10 \\
N_8 &\leq 1
\end{align*}
\]

- The constraint \( E_{62} \leq 10 \) (loop bound) derives from the statements \( i = 0, \ i < 10? \) and \( i++ \) contained in blocks 1, 2 and 6, respectively.
- The constraint \( N_8 \leq 1 \) derives from the fact that the “break instruction can be performed at most once.
• It is also possible to integrate in the formulation of IP problem part of the information resulting from knowledge of “infeasible” paths.

• In the example in the figure, it is assumed that the path indicated in red is infeasible and that there is a loop bound of 100. You can take account of this information (even if approximate) adding the constraint $N_2 + N_6 \leq 100$. 

Worst-case execution time and Integer Linear Programming: