Multi-Level Bin Packing Problem

CSE3000 Research Project Q4 2021/2022

Matthias Horn

Algorithmics group, Delft University of Technology, The Netherlands,
m.g.horn@tudelft.nl

April 3, 2022

Multi-Level Bin Packing Problem

Problem Description

Given

- set of items $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{n^0}\}$
  - with size $s(\mathcal{I}_j) \in \mathbb{N}_{>0}$, $1 \leq j \leq n^0$
- set of bins $\mathcal{B}^i = \{\mathcal{B}^i_1, \mathcal{B}^i_2, \ldots, \mathcal{B}^i_{n^i}\}$, $1 \leq i \leq m$, m levels
  - with size $s(\mathcal{B}^i_k) \in \mathbb{N}_{>0}$, capacity $w(\mathcal{B}^i_k) \in \mathbb{N}_{>0}$, and cost $c(\mathcal{B}^i_k) \in \mathbb{N}_{>0}$
  - $1 \leq k \leq n^i$, $1 \leq i \leq m$

Task

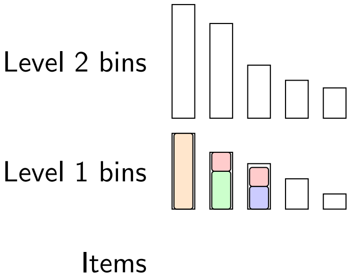
- insert each item into a bin of level 1
- insert each used bin of level i, $1 \leq i < m$ into a bin of level $i + 1$
- the capacity of a bin must not exceeded
- minimise total cost of used bins

Multi-Level Bin Packing Problem

Example with $m = 2$ levels

Multi-Level Bin Packing Problem

Example with $m = 2$ levels



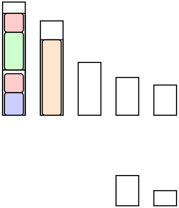
Multi-Level Bin Packing Problem

Example with $m = 2$ levels

Level 2 bins

Level 1 bins

Items



Class Constrained MLBP (CCMLBP)

In addition to MLBP

- Each item $\mathcal{I}_j \in \mathcal{I}$ belongs to a class $\kappa(\mathcal{I}_j) \in \{1, 2, \ldots, q\}$ where q is a positive integer.
- Each level $i \in M$ is associated with a bound $Q^i \in \mathbb{N}_{\geq 0}$ for the number of different classes within a bin $\mathcal{B}^i_k \in \mathcal{B}^i$.

Task

Find a packing that minimises the total cost of the used bins such that every bin at level $i \in M$ satisfies the capacity constraints and contains items from no more than Q^i classes.

MLBP with Conflict Constraints (MLBPCC)

In addition to MLBP

- Given: a conflict graph $G = (\mathcal{I}, E)$ where each node corresponds to an item
- An edge $(\mathcal{I}_j, \mathcal{I}_q) \in E$ indicates a conflict between items $\mathcal{I}_j$ and $\mathcal{I}_q$

Task

Find a packing that minimises the total cost of the used bins such that every bin satisfies the capacity constraints and each bin contains no items that are in conflict with each other.

MLBP with Partial Orders (MLBPPO)

In addition to MLBP

- Given: precedence constraints, denoted by $\prec$, among items $\mathcal{I}$
- Precedence refers to the relative ordering of top level bins in $\mathcal{B}^m$
- if there is a precedence $\mathcal{I}_j \prec \mathcal{I}_q$ between items $\mathcal{I}_j, \mathcal{I}_q \in \mathcal{I}$
  - then if $\mathcal{I}_j$ and $\mathcal{I}_q$ are packed into $\mathcal{B}_k^m$ and $\mathcal{B}_l^m$, respectively $\Rightarrow k < l$

Task

Find a packing that minimises the total cost of the used bins such that every bin satisfies the capacity constraints and all items satisfy their precedence constraints.

MLBP with Time Windows (MLBPTW)

In addition to MLBP

- Given: Time window (TW) $t_j = [e_j,\, l_j]$, $\mathcal{I}_j \in \mathcal{I}$ and penalty factor p
- Item $\mathcal{I}_j$ must be packed as early as possible within its earliest packing time e_j and its latest packing time l_j
- Items can only be packed into the same bin if their TWs overlap

Task

- let $u(\mathcal{I}_j)$ be the earliest time item $\mathcal{I}_j$ is packed into a bin
- in addition to the total bin costs, a penalty value $(u(\mathcal{I}_j) - e_j)\, p$ is added for each item $\mathcal{I}_j$

Find a packing that minimises the changed objective function such that every bin satisfies the capacity constraints and contains no items with non-overlapping time windows.

MLBP with Fragmentation Constraints (MLBPFC)

In addition to MLBP

- Each item $\mathcal{I}_j \in \mathcal{I}$ belongs to a group $g(\mathcal{I}_j) \in G = \{1, 2, \ldots, q\}$, $q \in \mathbb{N}_{>0}$
- Items of the same group should be packed into the same top level bin $\mathcal{B}^m$

Task

- let n_r be the number of top level bins that contains items of group $r \in G$
- in addition to the total bin costs, a penalty value $(n_r - 1)\,p$ is added for each group

Find a packing that minimises the changed objective function such that every bin satisfies the capacity constraints.

C++ Framework

- Provides
  - C++ classes to read instances
  - simple command line argument parser
  - MIP solver class
    - uses CPLEX to solve MIPs
    - add your decision variables, constraints, and objective function
  - sample case for the standard bin packing problem
- It is not mandatory to use the C++ Framework!
  - You can use other MIP solvers and/or other programming languages.

C++ Framework

Setup Framework - Linux

- download and install CPLEX Optimization Studio
  - `https://www.ibm.com/academic/topic/data-science`
  - you will find Optimization Studio under software
  - free for students, but you have to register
- set system environment variable CPLEXDIR=<directory of cplex>
  - e.g., export CPLEXDIR=/opt/ibm/ILOG/CPLEX_Studio201
- run make

- download and install CPLEX Optimization Studio
  - `https://www.ibm.com/academic/topic/data-science`
  - you will find Optimization Studio under software
  - free for students, but you have to register
- download and install Visual Studio with C++
  - `https://visualstudio.microsoft.com/vs/features/cplusplus/`

C++ Framework

Setup Visual Studio - Windows

- click on the project name in solution explorer tab and select properties
- under C/C++ →general →additional include directories →add:
  - <cplex directory>\CPLEX_StudioXXX\concert\include
  - <cplex directory>\CPLEX_StudioXXX\cplex\include
- under C/C++ →preprocessors →preprocessor definitions→add:
  - NDEBUG
  - _CONSOLE
  - IL_STD

C++ Framework

Setup Visual Studio - Windows

- under C/C++ →code generation →runtime library
  - set to “multi-threaded DLL (/MD)”

- under Linker →Input →additional dependencies→add:
  - <cplex directory>\CPLEX_StudioXXX\cplex\lib\x64_windows_vsXXXX\stat_mda\cplexXXX.lib
  - <cplex directory>\CPLEX_StudioXXX\cplex\lib\x64_windows_vsXXXX\stat_mda\ilocplex.lib
  - <cplex directory>\CPLEX_StudioXXX\concert\lib\x64_windows_vsXXXX\stat_mda\concert.lib

- under general →C++ language standard
  - set to “ISO C++ 17 Standard” or to “ISO C++ 20 Standard”