Vector Packing Solver (VPSolver)

Filipe Brandão
fdabrandao@dcc.fc.up.pt

March 3, 2014

1 p-dimensional vector packing instances

The p-dimensional vector packing problem, also called general assignment problem, is a generalization of bin packing with multiple constraints. In this problem, we are required to pack n items of m different types, represented by p-dimensional vectors, into as few bins as possible. In practice, this problem models, for example, static resource allocation problems where the minimum number of servers with known capacities is used to satisfy a set of services with known demands.

This p-dimensional vector packing solver accepts instances in the following format:

Format:

$$
\begin{array}{cccccc}
p & & & & & \\
\mathrm{W}^1 & \dots & \mathrm{W}^d & \dots & \mathrm{W}^p & \\
m & & & & & \\
w_1^1 & \dots & w_1^d & \dots & w_1^p & b_1 \\
\vdots & & \vdots & & \vdots & \vdots \\
w_i^1 & \dots & w_i^d & \dots & w_i^p & b_i \\
\vdots & & \vdots & & \vdots & \vdots \\
w_m^1 & \dots & w_m^d & \dots & w_m^p & b_m
\end{array}
$$

Description:

- p - number of dimensions;
- W^d - capacity of the d-th dimension;
- m - number of different items;
- b_i - demand of the i-th item;
- w_i^d - weight of the i-th item on the d-th dimension.

2 Components

- **vpsolver**
  - Description: solves vector packing instances using the general arc-flow formulation proposed in Brandão and Pedroso (2013a).
  - Requirements: Gurobi 5.0.0 or superior.
  - Compile:
    ```
    make bin/vpsolver
    ```
  - Usage:
    ```
    bin/vpsolver instance.vbp [method:-2] [binary:0] [vtype:I]
    ```
- **vbp2afg**
  - Description: builds an arc-flow graph containing every valid packing pattern for a given vector packing instance (using the algorithms proposed in Brandão 2012 and Brandão and Pedroso 2013a).
  - Compile:
    ```
    make bin/vbp2afg
    ```
  - Usage:
    ```
    bin/vbp2afg instance.vbp graph.afg [method:-2] [binary:0] [vtype:I]
    ```

- **afg2mps**
  - Description: converts arc-flow graphs into .mps models.
  - Compile:
    ```
    make bin/afg2mps
    ```
  - Usage:
    ```
    bin/afg2mps graph.afg model.mps
    ```
- **afg2lp**
  - Description: converts arc-flow graphs into .lp models.
  - Compile:
    ```
    make bin/afg2lp
    ```
  - Usage:
    ```
    bin/afg2lp graph.afg model.lp
    ```
- **solve_gurobi**
  - Description: solves .mps/.lp models using `Gurobi`.
  - Requirements: Gurobi 5.0.0 or superior.
  - Compile:
    ```
    make bin/solve_gurobi
    ```
  - Usage:
    ```
    bin/solve_gurobi model.mps$|$model.lp [vars.sol]
    ```
- **solve_glpk**
  - Description: solves .mps/.lp models using `GLPK`.
  - Requirements: GLPK 4.45 or superior.
  - Compile:
    ```
    make bin/solve_glpk
    ```
  - Usage:
    ```
    bin/solve_glpk model.mps$|$model.lp [vars.sol]
    ```
- **vbpsol**
  - Description: converts integer arc-flow solutions into vector packing solutions (using the algorithm proposed in Brandão 2012).
  - Compile:
    ```
    make bin/vbpsol
    ```
  - Usage:
    ```
    bin/vbpsol graph.afg vars.sol [print_instance:0]
    ```
- **gg_afg**
  - Description: Gilmore-Gomory's column generation approach. The multi-constraint knapsack sub-problems are solved using arc-flow graphs (see Brandão and Pedroso 2013a). If the parameter htlimit (heuristic time limit in seconds) is nonzero, Gurobi tries to find heuristic solutions considering only the restricted set of variables created during the column-generation procedure.
  - Requirements: Gurobi 5.0.0 or superior.
  - Compile:
    ```
    make bin/gg_afg
    ```
  - Usage:
    ```
    bin/gg_afg instance.vbp [method:-2] [binary:0] [tlimit:0]
    ```

3 Parameters

- **method**:
  - method=-2 (default)
    - Builds the Step-4' graph using the method proposed in Brandão and Pedroso (2013a).
  - method=-1
    - Builds the Step-3' graph using the method proposed in Brandão and Pedroso (2013a) without applying the final compression step.
  - method=0
    - Builds the Step-1 graph (Brandão 2012).
  - method=1
    - Builds the Step-4 using the graph compression algorithm proposed in Brandão (2012).
- **binary**:
  - binary=0 (dafault)
  - binary=1
    - Binary patterns: each pattern can contain at most one item of each type.
    - The binary constraints are introduced using the method proposed in Brandão and Pedroso (2013b).
- **vtype**:
  - vtype=I (Integer variables)
  - vtype=C (Continuous variable - linear relaxation only)

4 Scripts

VPSolver does not explicitly require any MIP solver in particular, though a good MIP solver may be necessary for solving large models. VPSolver includes several scripts for solving vector packing instances using different solvers:

- vpsolver_gurobi.sh
- vpsolver_cplex.sh
- vpsolver_coinor.sh
- vpsolver_glpk.sh
- vpsolver_lpsolve.sh
- vpsolver_scip.sh

These scripts can be used as follows:

- Solve a vector packing instance using solver X.

  ```
  scripts/vpsolver_X.sh --vbp instance.vbp
  ```

- Solve a .mps/.lp arc-flow model using solver X.

  ```
  scripts/vpsolver_X.sh --mps/--lp model.mps/.lp
  ```

- Solve a .mps/.lp arc-flow model using solver X and extract the solution (graph.afg must the underlying arc-flow graph of model model.mps/.lp).

  ```
  scripts/vpsolver_X.sh --mps/--lp model.mps/.lp --afg graph.afg
  ```

5 Examples

Solve a vector packing instance using Gurobi:

```
bin/vpsolver example.vbp
```

Solve a vector packing instance using Gurobi (step-by-step):

```
bin/vbp2afg example.vbp graph.afg   # 1. builds the arc-flow graph (graph.afg)
bin/afg2mps graph.afg model.mps     # 2. converts the arc-flow graph
                                    #    into a .mps file (model.mps)
bin/solve_gurobi model.mps vars.sol # 3. solves the MIP model and stores the
                                    #    solution in vars.sol
bin/vbpsol graph.afg vars.sol       # 4. outputs the vector packing solution
```

Solve a vector packing instance using GLPK (step-by-step):

```
bin/vbp2afg example.vbp graph.afg   # 1. builds the arc-flow graph (graph.afg)
bin/afg2lp graph.afg model.lp       # 2. converts the arc-flow graph
                                    #    into a .lp file (model.lp)
bin/solve_glpk model.lp vars.sol    # 3. solves the MIP model and stores the
                                    #    solution in vars.sol
bin/vbpsol graph.afg vars.sol       # 4. outputs the vector packing solution
```

Solve a vector packing instance using the COIN-OR script:

```
scripts/vpsolver_coinor.sh --vbp example.vbp
```

References

Brandão, F. (2012). Bin Packing and Related Problems: Pattern-Based Approaches. Master's thesis, Faculdade de Ciências da Universidade do Porto, Portugal.

Brandão, F. and Pedroso, J. P. (2013a). Bin Packing and Related Problems: General Arc-flow Formulation with Graph Compression. Technical Report DCC-2013-08, Faculdade de Ciências da Universidade do Porto, Portugal.

Brandão, F. and Pedroso, J. P. (2013b). Cutting Stock with Binary Patterns: Arc-flow Formulation with Graph Compression. Technical Report DCC-2013-09, Faculdade de Ciências da Universidade do Porto, Portugal.