Vector Packing Solver (VPSolver)

Filipe Brandão
fdabrandao@dcc.fc.up.pt

February 17, 2016

1 Vector packing instances

The p-dimensional vector bin packing problem (VBP), 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. The multiple-choice vector bin packing problem (MVP) is a variant of VBP in which bins have several types (i.e., sizes and costs) and items have several incarnations (i.e., will take one of several possible sizes).

This multiple-choice p-dimensional vector packing solver accepts instances in the following formats:

VBP Format (`.vbp` files):

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

Description:

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

MVP Format (`.mvp` files):

$$
\begin{array}{ccccc}
p & & & & \\
q & & & & \\
\mathrm{W}_1^1 & \dots & \mathrm{W}_1^p & \mathrm{C}_1 & \mathrm{Q}_1 \\
\vdots & & \vdots & \vdots & \vdots \\
\mathrm{W}_t^1 & \dots & \mathrm{W}_t^p & \mathrm{C}_t & \mathrm{Q}_t \\
m & & & & \\
t_1 & b_1 & & & \\
w_{1,1}^1 & \dots & w_{1,1}^p & & \\
\vdots & & \vdots & & \\
w_{1,t_1}^1 & \dots & w_{1,t_1}^p & & \\
 & \dots & & & \\
t_m & b_m & & & \\
w_{m,1}^1 & \dots & w_{m,1}^p & & \\
\vdots & & \vdots & & \\
w_{m,t_m}^1 & \dots & w_{m,t_m}^p & &
\end{array}
$$

Description:

- p - number of dimensions;
- q - number of different bin types;
- W_j^d - capacity on dimension d on bins of type j;
- C_j - cost per bin of type j;
- Q_j - number of bins of type j available;
- m - number of different item types;
- b_i - demand for items of type i;
- t_i - number of different incarnations of items of type i;
- w_{ij}^d - weight on dimension d of incarnation j of items of type i.

2 Components

- **vpsolver**
  - Description: solves multiple-choice vector packing instances using the method proposed in Brandão (2016), which is an extension of general arc-flow formulation proposed in Brandão and Pedroso (2016).
  - Requirements: Gurobi 5.0.0 or superior.
  - Usage:
    ```
    bin/vpsolver instance.vbp/.mvp [method:-2] [binary:0] [vtype:I]
    ```
- **vbp2afg**
  - Description: builds an arc-flow graph containing every valid packing pattern for a given multiple-choice vector packing instance (using the algorithm proposed in Brandão 2016).
  - Usage:
    ```
    bin/vbp2afg instance.vbp/.mvp graph.afg [method:-2] [binary:0] [vtype:I]
    ```
- **afg2mps**
  - Description: converts arc-flow graphs into .mps models.
  - Usage:
    ```
    bin/afg2mps graph.afg model.mps
    ```
- **afg2lp**
  - Description: converts arc-flow graphs into .lp models.
  - Usage:
    ```
    bin/afg2lp graph.afg model.lp
    ```
- **vbpsol**
  - Description: converts integer arc-flow solutions into vector packing solutions (using the algorithm proposed in Brandão 2012).
  - Usage:
    ```
    bin/vbpsol graph.afg vars.sol [print_instance:0]
    ```

3 Parameters

- **method**:
  - method=-2 (default)
    - Builds the Step-4' graph using the method proposed in Brandão (2016).
- **binary**:
  - binary=0 (dafault)
  - binary=1
    - Binary patterns: each pattern can contain at most one item of each type.
    - The binary constraints on the patterns are introduced using the method proposed in Brandão and Pedroso (2013).
- **vtype**:
  - vtype=I (Integer variables)
  - vtype=C (Continuous variables - linear relaxation)

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**, and **vpsolver_scip.sh**.

These scripts can be used as follows:

- Solve a vector packing instance using solver X.

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

- 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
  ```

- Solve a .mps/.lp model using solver X and write the solution to a file (vars.sol).

  ```
  scripts/vpsolver_X.sh --mps/--lp model.mps/.lp --wsol var.sol
  ```

5 Examples

Solve a vector packing instance using Gurobi:

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

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

```
# 1. Build the arc-flow graph (graph.afg):
$ bin/vbp2afg example.vbp graph.afg
# 2. Convert the arc-flow graph into a .mps model (model.mps):
$ bin/afg2mps graph.afg model.mps
# 3. Solve the MIP model and store the solution in vars.sol:
$ scritps/vpsolver_gurobi.sh --mps  model.mps --wsol vars.sol
# 4. Output the vector packing solution:
$ bin/vbpsol graph.afg vars.sol
```

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

```
# 1. Build the arc-flow graph (graph.afg):
$ bin/vbp2afg example.vbp graph.afg
# 2. Convert the arc-flow graph into a .lp model (model.lp):
$ bin/afg2lp graph.afg model.lp
# 3. Solve the MIP model and store the solution in vars.sol:
$ scritps/vpsolver_glpk.sh --lp  model.lp --wsol vars.sol
# 4. Output the vector packing solution:
$ bin/vbpsol graph.afg vars.sol
```

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. (2016). VPSolver 3: Multiple-choice Vector Packing Solver. arXiv:1602.04876.

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

Brandão, F. and Pedroso, J. P. (2016). Bin packing and related problems: General arc-flow formulation with graph compression. *Computers & Operations Research*, 69:56 – 67.