Instance Specification for Loading Problems

Jelke J. van Hoorn

May 23, 2018

1 Introduction

This document describes the structure of the instances that can be found on the website. Instances are provided in the JSON, XML and YAML file formats, with minimal differences across formats.

2 File Formats

The instances are specified in three different file formats: JSON, XML and YAML. The structure used in all formats is the same. Basically there are just a few types of constructs in these files:

- An object with properties, e.g., **instance**, **container**, **item** and **loadingspace**
- A property can be of different types:
  - A value, e.g., **name**, **length** and **weight**
  - A group of properties, e.g., **info**, **data** and **size**
  - A list of similar objects, e.g., **containers**, **items** and **loadingspaces**

2.1 The JSON file

JSON is a lightweight data-interchange format. The mapping between the instance structure and JSON is very straightforward:

- objects are objects
- properties are pairs
- a group of properties is an object
- a list is a list

The biggest difference with the other file formats is that the objects in a list do not have an own name, e.g., **instance**, **items** and **containers**.

2.2 The XML file

XML is an eXtensible Markup Language. The mapping between the instance structure and XML is again straightforward:

- objects are tags containing tags with different names
- properties are tags containing text
- a group of properties is a tag containing tags with different names
- a list is tag containing tags with the same name

One exception to this is the property **id** which is provided as attribute.

2.3 The YAML file

YAMLis a human friendly data serialization The mapping between the instance structure and YAML is once more straightforward:

- objects are represented by a mapping

Listing 2.1: A small JSON example

```
{
  "info": {
    "set": "Example",
    "name": "001"
  },
  "parameters": {
    "objectives": [
      {"name": "container_costs", "priority": 1, "weight": 1.0},
      {"name": "fill_rate",       "priority": 2, "weight": 1.0}],
    "constraints": [
      {"name": "maximum_weight"},
      {"name": "sequence"}]
    },
  "data": {
    "containers": [{
      "id": 1,
      "quantity": 9999,
      "loadingspaces": [{
        "id": 1,
        "size": {
          "length": 2000,
          "width": 1000,
          "height": 1000
        }
      }
      ],
      "maxWeight": 24000.0
    }
    ],
    "items": [{
        "id": 1,
        "quantity": 3,
        "size": {
          "length": 500,
          "width": 500,
          "height": 500
        },
        "orientations": "LWH,WLH",
        "weight": 100.0
      }, {
        "id": 2,
        "quantity": 1,
        "size": {
          "length": 750,
          "width": 750,
          "height": 500
        },
        "orientations": "LWH,WLH",
        "weight": 100.0
      }
    ]
  }
}
```

Listing 2.2: A small XML example

```
<instance>
  <info>
    <set>Example</set>
    <name>001</name>
  </info>
  <parameters />
  <data>
    <containers>
      <container id="1">
        <quantity>9999</quantity>
        <loadingspaces>
          <loadingspace id="1">
            <size>
              <length>2000</length>
              <width>1000</width>
              <height>1000</height>
            </size>
          </loadingspace>
        </loadingspaces>
        <maxWeight>24000.0</maxWeight>
      </container>
    </containers>
    <items>
      <item id="1">
        <quantity>3</quantity>
        <size>
          <length>500</length>
          <width>500</width>
          <height>500</height>
        </size>
        <orientations>LWH,WLH</orientations>
        <weight>100.0</weight>
      </item>
      <item id="2">
        <quantity>1</quantity>
        <size>
          <length>750</length>
          <width>750</width>
          <height>500</height>
        </size>
        <orientations>LWH,WLH</orientations>
        <weight>100.0</weight>
      </item>
    </items>
  </data>
</instance>
```

- properties are an entry in such mapping
- a group of properties also represented by a mapping
- a list is represented by a sequence of single entry mappings with equal names

Listing 2.3: A small YAML example

```yaml
instance:
  info:
    set: Example
    name: '001'
  parameters: ''
  data:
    containers:
    - container:
        id: 1
        quantity: 9999
        loadingspaces:
        - loadingspace:
            id: 1
            size:
              length: 2000
              width: 1000
              height: 1000
        maxWeight: 24000.0
    items:
    - item:
        id: 1
        quantity: 3
        size:
          length: 500
          width: 500
          height: 500
        orientations: LWH,WLH
        weight: 100.0
    - item:
        id: 2
        quantity: 1
        size:
          length: 750
          width: 750
          height: 500
        orientations: LWH,WLH
        weight: 100.0
```

3 Instance Structure

The structure of an instance has very few required elements, a lot of elements are optional and more will be added later. Optional fields typically become required in the presence of certain objectives or constraints, although a default is defined for most. The order of fields within an object or group, as well as the order within a list is flexible. It is advised to order lists by id and start with the required fields and order the rest alphabetically. This chapter follows that order.

3.1 Instance

info information about the instance.

required

Type: group

Fields:

set name of a set of instances

required

Type: string

name name of the instance

required

Type: string

parameters parameters of the instance, such as constraints and objectives.

required

Type: group

Fields: The details and the fields can be found in section 3.2.

data the data of the instance.

required

Type: group

Fields:

containers the containers to be filled

required

Type: list of containers. Description of a container can be found in section 3.3.

items the items to be loaded

required

Type: list of items. Description of an item can be found in section 3.5.

3.2 Parameters

constraints constraints of the instance.

optional

Type: list of constraints. Description of a constraint can be found in section 3.6.

objectives objectives of the instance.

required

Type: list of objectives. Description of an objective can be found in section 3.7.

3.3 Container

id id, unique within the containers.

required

Type: integer

quantity number of containers available.

required

Type: integer

loadingspaces the available spaces within a container

required

Type: list of loadingspaces. Description of a loadingspace can be found in section 3.4.

cost total cost of usage of this container.

optional

Type: float

maxWeight maximum allowed weight to load.

optional

Type: float

3.4 Loadingspace

id id, unique within the loadingspaces

required

Type: integer

size space to load

required

Type: group

Fields:

length length of the space

required

Type: integer

width width of the space

required

Type: integer

height height of the space

required

Type: integer

3.5 Item

id id, unique within the items

required

Type: integer

quantity number of this item to load

required

Type: integer

size size of the item

required

Type: group

Fields:

length length of the item

required

Type: integer

width width of the item

required

Type: integer

height height of the item

required

Type: integer

orientations allowed orientations

required

Type: string, a comma-separated list describing the allowed orientations of the items relative to the orientation (length-width-height) of the space, containing the following possible entries:

- HLW (height-length-width)
- HWL (height-width-length)
- LHW (length-height-width)
- LWH (length-width-height)
- WHL (width-height-length)
- WLH (width-length-height)

weight weight of the item

optional

Type: float

group group id of the item (groups are shipped together)

optional

Type: integer

3.6 Constraint

name name, unique within the constraints. A more detailed description can be found in section 4.

required

Type: string, should be one of: `maximum_weight`, `ship_together`, or `weight_distribution`

3.7 Objective

name name of the objective. A more detailed description can be found in 5.

required

Type: string, should be one of: `container_costs`, `fill_rate`, or `weight_distribution`

priority objectives are compared lexicographically in increasing order with respect to their priority.

required

Type: integer

weight the weighted averages of objectives with equal priority are taken with these weights.

required

Type: float

4 Constraints

This section lists the constraints that are currently available, along with a description of their purpose. More constraints may be added in the future. The optional fields noted in chapter 3 typically become required in the presence of certain constraints or objectives. This dependency on data fields is referred to as an 'existence requirement' of certain data fields. In addition to existence requirements, constraints can have propositional requirements, which state certain propositions that the data fields should adhere to. Typically propositional requirements state properties such as that a certain data field should be positive or non-negative, but there is generally no limit to their genericity.

The only constraints that required for each solution are the non-overlapping and the orientation constraints. All other constraints are only required once they are specified.

4.1 Maximum Weight

This constraint requires the total summed weight of all placements (items, boxes, and pallets) in all loading spaces of a certain container to not exceed a certain threshold. The threshold value can differ per container type and is loaded from the optional `maxWeight` field from the container data. When no maximum weight is specified for a container, it will default to infinity (relaxing the maximum weight constraint for that particular container). Similarly, a weight value is required for each item, box, and pallet, corresponding to the `weight` field for the corresponding data objects. When no weight is specified for an item, box, or pallet, it will default to zero (relaxing the constraint with respect to that particular item, box, or pallet). This constraint can be imposed on an instance by adding a constraint with the name: `maximum_weight`.

4.2 Must Place

Specific items must be placed when this constraint is active. Each pallet, box, and item will get a `place` attribute attached, which indicates whether it must occur as a placement in the solution.

4.3 Support

This constraint requires the lower surface of a placement to be supported (to overlap with the top surfaces of other surfaces) by at least a certain percentage. The overlap percentage can differ per item, box, and pallet and is loaded from the optional `support` field from the respective data objects. When no support field is specified, it defaults to one, corresponding to a full support requirement. For the implementation details of support checking, refer to the "Ngoi matrix"

section of "Tools for Loadbuilding Benchmarks". This constraint can be imposed on an instance by adding a constraint with the name: `support`.

4.4 Scannable Item Labels

Each item has a label associated with it, indicating a face on the item where a label is present. That specific side must be visible from the outside of the pallet on which the item is placed.

5 Objectives

This section lists the objectives that are currently available, along with a description of their purpose. More objectives may be added in the future. As said before, the optional fields from chapter 3 typically become required in the presence of certain constraints or objectives.