Package ‘gbp’

January 28, 2017

Type Package

Title A Bin Packing Problem Solver

Version 0.1.0.4

Author Guang Yang

Maintainer Guang Yang <gyang274@gmail.com>

Description Basic infrastructure and several algorithms for 1d-4d bin packing problem. This package provides a set of c-level classes and solvers for 1d-4d bin packing problem, and an r-level solver for 4d bin packing problem, which is a wrapper over the c-level 4d bin packing problem solver.
The 4d bin packing problem solver aims to solve bin packing problem, a.k.a container loading problem, with an additional constraint on weight.
Given a set of rectangular-shaped items, and a set of rectangular-shaped bins with weight limit, the solver looks for an orthogonal packing solution such that minimizes the number of bins and maximize volume utilization.
Each rectangular-shaped item i = 1, .. , n is characterized by length l_i, depth d_i, height h_i, and weight w_i, and each rectangular-shaped bin j = 1, .. , m is specified similarly by length l_j, depth d_j, height h_j, and weight limit w_j.
The item can be rotated into any orthogonal direction, and no further restrictions implied.

License MIT + file LICENSE

Depends R (>= 3.0.0), magrittr, data.table

Imports methods, rgl, Rcpp (>= 0.12.7)

Suggests testthat, knitr, rmarkdown

LinkingTo Rcpp, RcppArmadillo

SystemRequirements C++11, GNU make

LazyData TRUE

Encoding UTF-8

RoxygenNote 5.0.1

VignetteBuilder knitr

URL https://github.com/gyang274/gbp

BugReports https://github.com/gyang274/gbp/issues

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-01-28 17:31:25

R topics documented:

`bppSgl` *bppSgl*

Description

bpp solution of a single one order or multiple order

Usage

```
bppSgl
```

Format

An object of class `C++Class` of length 1.

Details

packing it into multiple bn w.r.t bn size and weight limit while select bn as small as possible

a bppSgl class instance has 6 fields:

- id: order id <integer>

list - should sorted or at least grouped w.r.t order id

- it: it position and scale <matrix>

- x, y, z, w it position and w in the bin <numeric> (w hold in bn when fit it in bn)

- l, d, h, w it scale along x, y, z and w <numeric> (w of it itself)

- bn: bn scale <vector>

- l, d, h, w bn scale along x, y, z and w <numeric>

- k: ticket id indicator 0 (if cannot fit into any bin), 1, 2, 3, 4, ... <vector>

- kb: ticket bn id indicator - which bn to use for packing each ticket <vector>

- ok: ok a quick indicator of any it can not fit into any bn? <bool>

bpp_solver *bpp_solver*

Description

bpp single or multiple order packing solver

Usage

```
bpp_solver(it, bn)
```

Arguments

`it`	it item <data.table> - oid: order id <integer> - sku: stock keeping unit as it id <character> - l: it length which scale will be placed along x-coordinate <numeric> - d: it depth which scale will be placed along y-coordinate <numeric> - h: it height which scale will be placed along z-coordinate <numeric> - w: it weight optional which scale will be used restriction <integer>
`bn`	bn bins <data.table> - id: bn id <character> - l: bn length limit along x-coordinate <numeric> - d: bn depth limit along y-coordinate <numeric> - h: bn height limit along z-coordinate <numeric> - w: bn weight limit along w - a separate single dimension <numeric> - l, d, h will be sorted to have l >= d >= h within solver

Details

bpp solver is designed to solve packing in warehouse

bpp solver digest input it as a list of order (oid) and each row contain one sku (sku) in an order with length (l), depth (d), height (h) and weight (w) and aims to pack it list into one or more bin from a given list of bin that bin length (l), depth (d), height (h), and a single weight limit (wlmt).

bn list must be sorted by volume so that the smaller the eariler and preferred, and each bn must be sorted so that l >= d >= h

bpp solver would call bpp_solver_dpp_wrapper and aims to find a packing schema such that: use as small number of bin as possible, and use small bin whenever possible, w.r.t the 3d none overlap constraint and weight limit constraint.

Value

sn

sn bpp_solution <list>

- it item <data.table>

- oid: order id <integer>

- sku: stock keeping unit as it id <character>

- tid: ticket id - an unique id within oid <integer>

- otid: order id x ticket id - an unique indentifier indicate it with same tid can be packed into one bin <character>

- bid: bn id <integer>

- x, y, z it position in bid bin <numeric>

- l, d, h it scale along x, y, z <numeric>

- w it weight <numeric>

- bn bins <data.table>

- id bn id <character>

- l bn length limit along x-coordinate <numeric>

- d bn depth limit along y-coordinate <numeric>

- h bn height limit along z-coordinate <numeric>

- w bn weight limit along w - a separate single dimension <numeric>

Note

bpp_solver is an r-level wrapper over c-level bpp_solver_dpp_wrapper, add otid as an unique indentifier.

`bpp_solver_dpp` *bpp_solver_dpp*

Description

main solver of e-commerce warehouse packing algorithm

Usage

```
bpp_solver_dpp(id, ldhw, m)
```

Arguments

`id`	<vector> id order id <integer> vector - should sorted or at least grouped w.r.t order id
`ldhw`	<matrix> it order list - l, d, h, w it scale along x, y, z and w <numeric> it columns should corresponding to id
`m`	<matrix> m a bin list - l, d, h, w bn scale along x, y, z and w <numeric> m should sorted w.r.t preference

Details

bpp init a list of order on sku in data.frame it - oid, sku, l, d, h, w: order id oid, stock keeping unit sku, length l, depth d, height h and weight w,

and also a list of available bn in data.frame bn - id, l, d, h, w: bn id, length l, depth d, height h, and weight limit w, sorted by peference often smaller prefered,

and a single weight limit wlmt applied on all bin.

bpp solver would solve

select least number of bn for packing each order w.r.t bn size and weight limit and make sure the bn selected are as small as possible.

Value

bppSgl

See Also

Other bpp_solver_dpp: `bpp_solver_dpp_wrapper`, `bpp_solver_sgl`

---

bpp_solver_dpp_wrapper

bpp_solver_dpp_wrapper

Description

a wrapper over bpp_solver_dpp and expose an nicer r interface

Usage

```
bpp_solver_dpp_wrapper(it, bn)
```

Arguments

`it`

<data.frame>

it order itemSKU list

- oid: order id <integer>

- sku: stock keeping unit - it id <character>

- l, d, h, w it scale along x, y, z and w <numeric>

- w will be used as constraint while l, d, h will be used as both constraint and objective

it must be sorted w.r.t oid

`bn`

<data.frame>

bn a bin list

- id: bin id <character>

- l, d, h, w bn scale along x, y, z and w <numeric>

bn must be sorted w.r.t preference and have l >= d >= h

Value

sn <list>

sn solution - it order itemSKU list with tid, bid, and x, y, z <data.frame>

- oid: order id inherited from it <character>

- tid: ticket id implied one order can be packed using several ticket id <character>

each ticket id corresponding to a bid bin id which indicates which bin to use for packing

- bid: bin id which bn in bn list should be used in pakcing <character>

- sku: stock keeping unit it id <character>

- x, y, z it position in the bin <numeric>

- l, d, h it scale along x, y, z <numeric>

l, d, h is not inherited from it as it can be rotated to different orientation for packing

- w it weight scale inherited from it <numeric>

See Also

Other bpp_solver_dpp: `bpp_solver_dpp`, `bpp_solver_sgl`

`bpp_solver_sgl` *bpp_solver_sgl*

Description

subroutine of bpp_solver_dpp

Usage

```
bpp_solver_sgl(ldhw, m)
```

Arguments

`ldhw`	<matrix> it order list - l, d, h, w it scale along x, y, z and w <numeric> it columns should corresponding to id
`m`	<matrix> m a bin list - l, d, h, w bn scale along x, y, z and w <numeric> m should sorted w.r.t preference

Details

fit a single order into bn list, call gbp4d_solver_dpp_filt() as main solver.

Value

bppSgl

See Also

Other bpp_solver_dpp: `bpp_solver_dpp_wrapper`, `bpp_solver_dpp`

`bpp_viewer` *bpp_viewer*

Description

bpp single or multiple order packing solution viewer

Usage

```
bpp_viewer(sn, title = NULL, subtitle = NULL)
```

Arguments

`sn`	sn bpp_solution from bpp_solver <list> - it item <data.table> - oid: order id <integer> - sku: stock keeping unit as it id <character> - tid: ticket id - an unique id within oid <integer> - otid: order id x ticket id - an unique indentifier indicate it with same tid can be packed into one bin <character>

- bid: bn id <integer>
- x, y, z it position in bid bin <numeric>
- l, d, h it scale along x, y, z <numeric>
- w it weight <numeric>
- bn bins <data.table>
- id bn id <character>
- l bn length limit along x-coordinate <numeric>
- d bn depth limit along y-coordinate <numeric>
- h bn height limit along z-coordinate <numeric>
- w bn weight limit along w - a separate single dimension <numeric>

`title` title <character>

`subtitle` subtitle <character>

See Also

Other bpp_viewer: `bpp_viewer_single`

`bpp_viewer_single` *bpp_viewer_single*

Description

bpp solution viewer on single bin all item

Usage

```
bpp_viewer_single(it, bn, title = NULL, subtitle = NULL,
  it_rgl_control = NULL, bn_rgl_control = NULL, label_it = TRUE,
  label_bn = TRUE)
```

Arguments

`it` it item <data.table>
- id it id <integer>
- x, y, z it position w.r.t bins <numeric>
- l, d, h it scale along x, y, z <numeric>
- w it weight <numeric>
- auto: cc, wd, txt point and lines color, size, legend <numeric/character, numeric, character>

`bn` bn bins <data.table>
- id bn id <integer>
- l, d, h bn scale <numeric>
- w bn weight limit <numeric>
- auto: cc, wd, txt point and lines color, size, legend <numeric/character, numeric, character>

`title`	title <character>
`subtitle`	subtitle <character>
`it_rgl_control`	control the color of it in rgl
`bn_rgl_control`	control the color of bn in rgl
`label_it`	label text on it corner or not
`label_bn`	label text on bn corner or not

See Also

Other bpp_viewer: `bpp_viewer`

`create_bn_rgl_control` *create_bn_rgl_control*

Description

subroutine of bpp_viewer_single

Usage

```
create_bn_rgl_control()
```

`create_it_cube3d` *create_it_cube3d*

Description

subroutine of bpp_viewer_single

Usage

```
create_it_cube3d(id, x, y, z, l, d, h, cc, wd, txt, itxt = TRUE)
```

Arguments

`id`	id
`x`	x-coordinate
`y`	y-coordinate
`z`	z-coordinate
`l`	length along x-coordinate
`d`	depth along y-coordinate
`h`	height along z-coordinate
`cc`	color
`wd`	width
`txt`	text
`itxt`	plot text or not

Details

add it or bn on current rgl device

`create_it_rgl_control` *create_it_rgl_control*

Description

subroutine of bpp_viewer_single

Usage

```
create_it_rgl_control()
```

`gbp` *gbp*

Description

a collection of 1d, 2d, 3d and 4d bin packing problem solver

solver

r-level:

wrapper over c-level function aims solving e-commerce bin packing problem

bpp_solver

c-level:

core class and solver on 1d, 2d, 3d and 4d bpp

gbp1d_solver_dpp

gbp2d_solver_dpp

gbp2d_solver_dpp_filt

gbp3d_solver_dpp

gbp3d_solver_dpp_filt

gbp4d_solver_dpp

gbp4d_solver_dpp_filt

bpp_solver_sgl

bpp_solver_dpp

optimizer

TODO: implementing a bin-shuffing optimizer?

viewer

rgl 3d show packing obtained via bpp_solver

bpp_viewer

gbp1d *gbp1d*

Description

generalized bin packing problem in 1 dimension, a.k.a knapsack 0-1 problem.

Usage

```
gbp1d
```

Format

An object of class `C++Class` of length 1.

Details

gbp1d init a profit vector p, a weight vector w, and a weight constraint c, gbp1d solver would solve

maximize sum_j=1^n p_j x_j

subject to sum_j=1^n w_j x_j leq c x_j in 0, 1, j = 1, ..., n

and instantiate a gbp1d object with a selectin vector x and an objective z.

gbp1d is implemented as rcpp class, an instantiate can be solved by calling gbp1d_solver_dpp(p, w, c) and gbp1d_solver_min(p, w, c)

See Also

Other gbp1d: `gbp1d_solver_dpp`

gbp1d_solver_dpp *gbp1d_solver_dpp*

Description

solve gbp1d via dynamic programming simple - adagio::knapsnak()

Usage

```
gbp1d_solver_dpp(p, w, c)
```

Arguments

`p`	p profit <vector>::<numeric>
`w`	w weight <vector>::<integer>
`c`	c constraint on weight <integer>

Details

a dynamic programming solver on gbp1d instantiate - knapsack 0-1 problem, see gbp1d.

gbp1d init a profit vector p, a weight vector w, and a weight constraint c, gbp1d solver would solve

maximize sum_j=1^n p_j x_j

subject to sum_j=1^n w_j x_j leq c x_j in 0, 1, j = 1, ..., n

and instantiate a gbp1d object with a selectin vector x and an objective z.

gbp1d is implemented as rcpp class, an instantiate can be solved by calling gbp1d_solver_dpp(p, w, c) and gbp1d_solver_min(p, w, c)

Value

gbp1d a gbp1d instantiate with p profit, w weight, c constraint on weight, k selection, o objective, and ok an indicator of all fit or not.

See Also

Other gbp1d: `gbp1d`

`gbp2d` *gbp2d*

Description

generalized bin packing problem in 2 dimension, a.k.a rectangle fill.

Usage

```
gbp2d
```

Format

An object of class `C++Class` of length 1.

Details

gbp2d init a profit vector p, a length vector l, a depth vector d, a length constraint ml, and a depth constraint md on l x d rectangle with geometry intepretation.

gbp2d solver would solve

maximize sum_j=1^n p_j k_j

subject to fit (l_j, d_j) at coordinate (x_j, y_j) such that no overlap in ml x md, j = 1, ..., n

and instantiate a gbp2d object with a x-axis coordinate vector x, a y-axis coordinate vector y, a selection vector k, and an objective o.

a gbp2d class instance has 6 fields:

- p: profit of it fit into bn <vector>

created via cluster max(l, d) and min(l, d) via gbp2d_solver_dpp_prep_create_p()

- it: it position and scale <matrix>

- x, y it position in the bin <numeric>

- l, d it scale along x and y <numeric>

- bn: bn scale <vector>

- l, d bn scale along x and y <numeric>

- k: selection indicator 0, 1 <vector>

- o: objective achivement volumn fit in over volumn overall <numeric>

- ok: a quick indicator of all it fit into bn? <bool>

Note

p is a proxy of ranking on rectangle fit difficulty, often a function w.r.t max(l, d) and l x d

See Also

Other gbp2d: `gbp2d_checkr`, `gbp2d_solver_dpp`

`gbp2d_checkr` *gbp2d_checkr*

Description

auxilium of gbp2d and gbp2d_solver_dpp

Usage

```
gbp2d_checkr(sn)
```

Arguments

`sn`	<gbp2d> gbp2d object from gbp2d_solver_dpp() solution.

Details

check fit solution is valid: no conflict between item and bin, and no conflict between each pair of item.

Value

okfit? <bool>

See Also

Other gbp2d: `gbp2d_solver_dpp`, `gbp2d`

`gbp2d_it_create_ktlist`

gbp2d_it_create_ktlist

Description

create ktlist from itlist

Usage

```
gbp2d_it_create_ktlist(bn, it, xp, ktinit, nlmt)
```

Arguments

`bn` bn scale <vector> - l, d bn scale along x and y <numeric>

`it` it position and scale <matrix> - x, y it position in the bin <numeric> - l, d it scale along x and y <numeric>

`xp` xp extreme point position and residual space scale <matrix> - x, y xp position in the bin <numeric> - l, d xp residual space scale along x and y <numeric>

`ktinit` kt candidate scale without position <matrix> - l, d kt scale along x and y which open to orientation <numeric>

`nlmt` nlmt: limit on ktlist n max-value

Details

core function in gbp2d_solver_dpp select highest profitable it not yet fit into bn and return all possbile fit w.r.t xp and orientation

Value

Ktlist2d

Note

should make sure it kt can be fit in bin outside

internal function use in gbp2d_solver_dpp() for creating Ktlist2d object for fit.

See Also

Other gbp2d_it: `Ktlist2d`

`gbp2d_solver_dpp` *gbp2d_solver_dpp*

Description

solve gbp2d via extreme point heuristic and best information score fit strategy.

Usage

```
gbp2d_solver_dpp(p, ld, m)
```

Arguments

`p` p profit of it fit into bn <vector> - cluster max(l, d) and min(l, d) via gbp2d_solver_dpp_prep_create_p()

`ld` it position and scale <matrix> - l, d it scale along x and y, subject to orientation rotation <numeric>

`m` bn scale <vector> - l, d bn scale along x and y <numeric>

Details

gbp2d init a profit vector p, a length vector l, a depth vector d, a length constraint ml, and a depth constraint md on l x d rectangle with geometry intepretation.

gbp2d solver would solve

maximize sum_j=1^n p_j k_j

subject to fit (l_j, d_j) at coordinate (x_j, y_j) such that no overlap in ml x md, j = 1, ..., n

and instantiate a gbp2d object with a x-axis coordinate vector x, a y-axis coordinate vector y, a selection vector k, and an objective o.

Value

gbp2d a gbp2d instantiate with p profit, it item (x, y, l, d) position scale matrix, bn bin (l, d) scale vector, k selection, o objective, and ok an indicator of all fit or not.

See Also

Other gbp2d: `gbp2d_checkr`, `gbp2d`

`gbp2d_solver_dpp_filt` *gbp2d_solver_dpp_filt*

Description

solve gbp2d w.r.t select most preferable often smallest bin from bn list

Usage

```
gbp2d_solver_dpp_filt(ld, m)
```

Arguments

`ld`	it scale <matrix> - l, d it scale along x and y <numeric>
`m`	bn scale <matrix> - l, d bn scale along x and y <numeric> - l, d in row and each col is a single bn should make sure bn list are sorted via volume so that the first col is the most prefered smallest bn, and also the last col is the least prefered largest and often dominant bn should make sure no X in front of Y if bnX dominant bnY, bnX dominant bnY if all(X(l, d) > Y(l, d)) and should always prefer Y. should make sure bn such that l >= d or vice versa.

Details

gbp2d_solver_dpp_filt is built on top of gbp2d_solver_dpp aims to select the most preferable bn from a list of bn that can fit all or most it

gbp2d_solver_dpp()'s objective is fit all or most it into a single given bn (l, d)

gbp2d_solver_dpp_filt()'s objective is select the most preferable given a list of bn where bn list is specified in 2xN matrix that the earlier column the more preferable

gbp2d_solver_dpp_filt() use an approx binary search and determine f w.r.t bn.n_cols where f = 1 indicate the bn being selected and only one of 1 in result returned.

ok = true if any bin can fit all it and algorithm will select smallest bn can fit all otherwise ok = false and algorithm will select a bn can maximize volume of fitted it

often recommend to make the last and least preferable bn dominate all other bn in list when design bn list, bnX dominant bnY if all(X(l, d) > Y(l, d)).

Value

gbp2q a gbp2q instantiate with p profit, it item (x, y, l, d) position scale matrix, bn bin (l, d) scale matrix, k it selection, o objective, f bn selection, and ok an indicator of all fit or not.

See Also

Other gbp2q: `gbp2q_checkr`, `gbp2q`

gbp2d_solver_dpp_prep_create_p

gbp2d_solver_dpp_prep_create_p

Description

auxilium of gbp2d_solver_dpp

Usage

```
gbp2d_solver_dpp_prep_create_p(ld, m)
```

Arguments

`ld`	2xN matrix of l, d of it
`m`	2x1 vector of l, d of bn

Details

create p via ld and m via cluster max(l, d) and min(l, d) strategy

Value

p

gbp2d_viewer

gbp2d_viewer

Description

gbp2d solution viewer

Usage

```
gbp2d_viewer(sn, title = NULL, subtitle = NULL)
```

Arguments

`sn`	sn gbp2d object, solution from gbp2d_solver_dpp, see gbp2d.
`title`	title <character>
`subtitle`	subtitle <character>

gbp2q *gbp2q*

Description

generalized bin packing problem in 2 dimension, a.k.a rectangle fill.

Usage

```
gbp2q
```

Format

An object of class `C++Class` of length 1.

Details

gbp2d init a profit vector p, a length vector l, a depth vector d, a length constraint ml, and a depth constraint md on l x d rectangle with geometry intepretation.

gbp2d solver would solve

maximize sum_j=1^n p_j k_j

subject to fit (l_j, d_j) at coordinate (x_j, y_j) such that no overlap in ml x md, j = 1, ..., n

and instantiate a gbp2d object with a x-axis coordinate vector x, a y-axis coordinate vector y, a selection vector k, and an objective o.

gbp2q solver would also select the most preferred often smallest m from a list of m(l, d) after determine all or the higest volume set of ld can fit into one m(l, d).

a gbp2q class instance has 7 fields:

- p: profit of it fit into bn <vector>

created via cluster max(l, d) and min(l, d) via gbp2d_solver_dpp_prep_create_p()

- it: it position and scale <matrix>

- x, y it position in the bin <numeric>

- l, d it scale along x and y <numeric>

- bn: bn scale <matrix>

- l, d bn scale along x and y <numeric>

matrix of 2 rows and each column is a single bn

should make sure bn list are sorted via volume so that the first col is the most prefered smallest bn, and also the last col is the least prefered largest and often dominant bn

should make sure no X in front of Y if bnX dominant bnY, bnX dominant bnY if all(X(l, d) > Y(l, d)) and should always prefer Y.

should make sure bn such that l >= d or vice versa.

- k: selection indicator 0, 1 on it <vector>

- f: selection indicator 0, 1, 2, 3 on bn <vector>

f in result should have no 0 and only one of 1

- o: objective achivement volumn fit in over volumn overall <numeric>

- ok: a quick indicator of all it fit into bn? <bool>

See Also

Other gbp2q: `gbp2d_solver_dpp_filt`, `gbp2q_checkr`

`gbp2q_checkr` *gbp2q_checkr*

Description

auxilium of gbp2q and gbp2d_solver_dpp_filt

Usage

```
gbp2q_checkr(sn)
```

Arguments

`sn`	<gbp2q> gbp2q object from gbp2d_solver_dpp_filt() solution.

Details

check fit solution is valid: no conflict between item and bin, and no conflict between each pair of item.

Value

okfit? <bool>

See Also

Other gbp2q: `gbp2d_solver_dpp_filt`, `gbp2q`

gbp2q_viewer *gbp2q_viewer*

Description

gbp2q solution viewer

Usage

```
gbp2q_viewer(sn, title = NULL, subtitle = NULL)
```

Arguments

`sn`	sn gbp2q object, solution from gbp2d_solver_dpp_filt, see gbp2q.
`title`	title <character>
`subtitle`	subtitle <character>

gbp3d *gbp3d*

Description

generalized bin packing problem in 3 dimension, a.k.a bin packing problem.

Usage

```
gbp3d
```

Format

An object of class `C++Class` of length 1.

Details

gbp3d init a profit vector p, a length vector l, a depth vector d, a height vector h, and also a length constraint ml, a depth constraint md, and a height constraint mh on l x d x h cuboid with geometry intepretation.

gbp3d solver would solve

maximize sum_j=1^n p_j k_j

subject to fit (l_j, d_j, h_j) at coordinate (x_j, y_j, z_j) such that no overlap in ml x md x mh cuboid, j = 1, ..., n

and instantiate a gbp3d object with a x-axis coordinate vector x, a y-axis coordinate vector y, a z-axis coordinate vector z, a selection vector k, and an objective o.

a gbp3d class instance has 6 fields:

- p: profit of it fit into bn <vector>

created via cluster max(l, d, h) and area via gbp3d_solver_dpp_main_create_p()

- it: it position and scale <matrix>

- x, y, z it position in the bin <numeric>

- l, d, h it scale along x, y, z <numeric>

- bn: bn scale <vector>

- l, d, h bn scale along x, y, z <numeric>

- k: selection indicator 0, 1 <vector>

- o: objective achivement volumn fit in over volumn overall <numeric>

- ok: a quick indicator of all it fit into bn? <bool>

Note

p is a proxy of ranking on cuboid fit difficulty, often a func of max(l, d, h), surface, volume and solver would often maximize sum_j=1^n v_j k_j instead of sum_j=1^n p_j k_j

See Also

Other gbp3d: `gbp3d_checkr`, `gbp3d_solver_dpp`

`gbp3d_checkr` *gbp3d_checkr*

Description

auxilium of gbp3d_solver_dpp

Usage

```
gbp3d_checkr(sn)
```

Arguments

`sn`	<gbp3d> gbp3d object from gbp3d_solver_dpp() solution.

Details

check fit solution is valid: no conflict between item and bin, and no conflict between each pair of item.

Value

okfit? <bool>

See Also

Other gbp3d: `gbp3d_solver_dpp`, `gbp3d` gbp3d_it_create_ktlist

gbp3d_it_create_ktlist

Description

create ktlist from itlist

Usage

```
gbp3d_it_create_ktlist(bn, it, xp, ktinit, nlmt)
```

Arguments

`bn`	bn scale <vector> - l, d, h bn scale along x, y, z <numeric>
`it`	it position and scale <matrix> - x, y, z it position in the bin <numeric> - l, d, h it scale along x, y, z <numeric>
`xp`	xp extreme point position and residual space scale <matrix> - x, y, z xp position in the bin <numeric> - l, d, h xp residual space scale along x, y, z <numeric>
`ktinit`	kt candidate scale without position <matrix> - l, d, h kt scale along x, y, z which open to orientation <numeric>
`nlmt`	nlmt: limit on ktlist n max-value

Details

core function in gbp3d_solver_dpp select highest profitable it not yet fit into bn and return all possbile fit w.r.t xp and orientation

Value

Ktlist3d

Note

should make sure it kt can be fit in bin outside

internal function use in gbp3d_solver_dpp() for creating Ktlist3d object for fit.

See Also

Other gbp3d_it: `Ktlist3d`

`gbp3d_solver_dpp` *gbp3d_solver_dpp*

Description

solve gbp3d via extreme point heuristic and best information score fit strategy.

Usage

```
gbp3d_solver_dpp(p, ldh, m)
```

Arguments

`p` p profit of it fit into bn <vector> - cluster max(l, d) and min(l, d) via gbp3d_solver_dpp_prep_create_p()

`ldh` it position and scale <matrix> - l, d, h it scale along x, y, z, subject to orientation rotation <numeric>

`m` bn scale <vector> - l, d, h bn scale along x, y, z <numeric>

Details

gbp3d init a profit vector p, a length vector l, a depth vector d, a height vector h, and also a length constraint ml, a depth constraint md, and a height constraint mh on l x d x h cuboid with geometry intepretation.

gbp3d solver would solve

maximize sum_j=1^n p_j k_j

subject to fit (l_j, d_j, h_j) at coordinate (x_j, y_j, z_j) such that no overlap in ml x md x mh cuboid, j = 1, ..., n

and instantiate a gbp3d object with a x-axis coordinate vector x, a y-axis coordinate vector y, a z-axis coordinate vector z, a selection vector k, and an objective o.

Value

gbp3d a gbp3d instantiate with p profit, it item (x, y, z, l, d, h) position scale matrix, bn bin (l, d, h) scale vector, k selection, o objective, and ok an indicator of all fit or not.

See Also

Other gbp3d: `gbp3d_checkr`, `gbp3d`

`gbp3d_solver_dpp_filt` *gbp3d_solver_dpp_filt*

Description

solve gbp3d w.r.t select most preferable often smallest bin from bn list

Usage

```
gbp3d_solver_dpp_filt(ldh, m)
```

Arguments

`ldh` it scale <matrix> - l, d, h it scale along x, y, z <numeric>

`m` bn scale <matrix> - l, d, h bn scale along x, y, z <numeric> - l, d, h in row and each col is a single bn

should make sure bn list are sorted via volume so that the first col is the most prefered smallest bn, and also the last col is the least prefered largest and often dominant bn

should make sure no X in front of Y if bnX dominant bnY, bnX dominant bnY if all(X(l, d, h) > Y(l, d, h)) and should always prefer Y.

should make sure bn such that l >= d >= h or vice versa.

Details

gbp3d_solver_dpp_filt is built on top of gbp3d_solver_dpp aims to select the most preferable bn from a list of bn that can fit all or most it

gbp3d_solver_dpp()'s objective is fit all or most it into a single given bn (l, d, h)

gbp3d_solver_dpp_filt()'s objective is select the most preferable given a list of bn where bn list is specified in 3xN matrix that the earlier column the more preferable

gbp3d_solver_dpp_filt() use an approx binary search and determine f w.r.t bn.n_cols where f = 1 indicate the bn being selected and only one of 1 in result returned.

ok = true if any bin can fit all it and algorithm will select smallest bn can fit all otherwise ok = false and algorithm will select a bn can maximize volume of fitted it

often recommend to make the last and least preferable bn dominate all other bn in list when design bn list, bnX dominant bnY if all(X(l, d, h) > Y(l, d, h)).

Value

gbp3q a gbp3q instantiate with p profit, it item (x, y, z, l, d, h) position scale matrix, bn bin (l, d, h) scale matrix, k it selection, o objective, f bn selection, and ok an indicator of all fit or not.

See Also

Other gbp3q: `gbp3q_checkr`, `gbp3q`

`gbp3d_solver_dpp_prep_create_p` *gbp3d_solver_dpp_prep_create_p*

Description

auxilium of gbp3d_solver_dpp

Usage

```
gbp3d_solver_dpp_prep_create_p(ldh, m)
```

Arguments

`ldh`	3xN matrix of l, d, h of it
`m`	3x1 vector of l, d, h of bn

Details

create p via ldh and m via cluster max(l, d, h) and area strategy

Value

p

`gbp3d_viewer` *gbp3d_viewer*

Description

gbp3d solution viewer

Usage

```
gbp3d_viewer(sn, title = NULL, subtitle = NULL)
```

Arguments

`sn`	sn gbp3d object, solution from gbp3d_solver_dpp, see gbp3d.
`title`	title <character>
`subtitle`	subtitle <character>

gbp3q *gbp3q*

Description

generalized bin packing problem in 3 dimension, a.k.a bin packing problem.

Usage

```
gbp3q
```

Format

An object of class `C++Class` of length 1.

Details

gbp3d init a profit vector p, a length vector l, a depth vector d, a height vector h, and also a length constraint ml, a depth constraint md, and a height constraint mh on l x d x h cuboid with geometry intepretation.

gbp3d solver would solve

maximize sum_j=1^n p_j k_j

subject to fit (l_j, d_j, h_j) at coordinate (x_j, y_j, z_j) such that no overlap in ml x md x mh cuboid, j = 1, ..., n

and instantiate a gbp3d object with a x-axis coordinate vector x, a y-axis coordinate vector y, a z-axis coordinate vector z, a selection vector k, and an objective o.

gbp3q solver would also select the most preferred often smallest m from a list of m(l, d, h) after determine all or the higest volume set of ld can fit into one m(l, d, h).

a gbp3q class instance has 7 fields:

- p: profit of it fit into bn <vector>

created via cluster max(l, d, h) and area via gbp3d_solver_dpp_main_create_p()

- it: it position and scale <matrix>

- x, y, z it position in the bin <numeric>

- l, d, h it scale along x, y, z <numeric>

- bn: bn scale <matrix>

- l, d, h bn scale along x, y, z <numeric>

matrix of 3 rows and each column is a single bn

should make sure bn list are sorted via volume so that the first col is the most prefered smallest bn, and also the last col is the least prefered largest and often dominant bn

should make sure no X in front of Y if bnX dominant bnY, bnX dominant bnY if all(X(l, d, h) > Y(l, d, h)) and should always prefer Y.

should make sure bn such that l >= d or vice versa.

- k: selection indicator 0, 1 on it <vector>

- f: selection indicator 0, 1, 2, 3 on bn <vector>

f in result should have no 0 and only one of 1

- o: objective achivement volumn fit in over volumn overall <numeric>

- ok: a quick indicator of all it fit into bn? <bool>

See Also

Other gbp3q: `gbp3d_solver_dpp_filt`, `gbp3q_checkr`

`gbp3q_checkr` *gbp3q_checkr*

Description

auxilium of gbp3d_solver_dpp_filt

Usage

```
gbp3q_checkr(sn)
```

Arguments

`sn`	<gbp3q> gbp3q object from gbp3d_solver_dpp_filt() solution.

Details

check fit solution is valid: no conflict between item and bin, and no conflict between each pair of item.

Value

okfit? <bool>

See Also

Other gbp3q: `gbp3d_solver_dpp_filt`, `gbp3q`

gbp3q_viewer *gbp3q_viewer*

Description

gbp3q solution viewer

Usage

```
gbp3q_viewer(sn, title = NULL, subtitle = NULL)
```

Arguments

`sn`	sn gbp3q object, solution from gbp3d_solver_dpp_filt, see gbp3q.
`title`	title <character>
`subtitle`	subtitle <character>

gbp4d *gbp4d*

Description

generalized bin packing problem in 4 dimension, a.k.a bin packing problem with weight limit.

Usage

```
gbp4d
```

Format

An object of class `C++Class` of length 1.

Details

gbp4d init a profit vector p, a length l, a depth d, a height h, and a weight w, along with associate constraints ml, md, mh and mw. gbp4d should fit it (l, d, h, w) into bn (ml, md, mh, mw) with w on weight limit constraint and l, d, h on geometry intepretation. gbp4d solver would solve

maximize sum_j=1^n p_j k_j

subject to sum_j=1^n w_j k_j leq mw and

fit (l_j, d_j, h_j) at coordinate (x_j, y_j, z_j) such that no overlap in ml x md x mh cuboid, j = 1, ..., n

and instantiate a gbp4d object with a x-axis coordinate vector x, a y-axis coordinate vector y, a z-axis coordinate vector z, a selection vector k, and an objective o.

a gbp4d class instance has 6 fields:

- p: profit of it fit into bn <vector>

created via cluster w via gbp1d, cluster max(l, d, h) and area via gbp4d_solver_dpp_main_create_p()

- it: it position and scale <matrix>

- x, y, z, w it position and w in the bin <numeric> (w hold in bn when fit it in bn)

- l, d, h, w it scale along x, y, z and w <numeric> (w of it itself)

- bn: bn scale <vector>

- l, d, h, w bn scale along x, y, z and w <numeric>

- k: selection indicator 0, 1 <vector>

- o: objective achivement volumn fit in over volumn overall <numeric>

- ok: a quick indicator of all it fit into bn? <bool>

Note

p is a proxy of ranking on cuboid fit difficulty, often a func of max(l, d, h), surface, volume and solver would often maximize sum_j=1^n v_j k_j instead of sum_j=1^n p_j k_j

See Also

Other gbp4d: `gbp4d_checkr`, `gbp4d_solver_dpp`

`gbp4d_checkr` *gbp4d_checkr*

Description

auxilium of gbp4d_solver_dpp

Usage

```
gbp4d_checkr(sn)
```

Arguments

`sn`	<gbp4d> gbp4d object from gbp4d_solver_dpp() solution.

Details

check fit solution is valid: no conflict between item and bin, and no conflict between each pair of item, and no conflict on weight limit.

Value

okfit? <bool>

See Also

Other gbp4d: `gbp4d_solver_dpp`, `gbp4d` `gbp4d_it_create_ktlist` *gbp4d_it_create_ktlist*

Description

create ktlist from itlist

Usage

```
gbp4d_it_create_ktlist(bn, it, xp, ktinit, nlmt)
```

Arguments

`bn` bn scale <vector> - l, d, h, w bn scale along x, y, z and w <numeric>

`it` it position and scale <matrix> - x, y, z, w it position and w in the bin <numeric> - l, d, h, w it scale along x, y, z and w <numeric>

`xp` xp extreme point position and residual space scale <matrix> - x, y, z, w xp position and w in the bin <numeric> - l, d, h, w xp residual space scale along x, y, z and w <numeric>

`ktinit` kt candidate scale without position <matrix> - l, d, h, w kt scale along x, y, z, w which open to orientation <numeric>

`nlmt` nlmt: limit on ktlist n max-value

Details

core function in gbp4d_solver_dpp select highest profitable it not yet fit into bn and return all possbile fit w.r.t xp and orientation

Value

Ktlist4d

Note

should make sure it kt can be fit in bin outside

internal function use in gbp4d_solver_dpp() for creating Ktlist4d object for fit.

See Also

Other gbp4d_it: `Ktlist4d`

gbp4d_solver_dpp *gbp4d_solver_dpp*

Description

solve gbp4d via extreme point heuristic and best information score fit strategy.

Usage

```
gbp4d_solver_dpp(p, ldhw, m)
```

Arguments

p	p profit of it fit into bn <vector> - cluster w via gbp1d, cluster max(l,d,h) and area via gbp4d_solver_dpp_main_create_p()
ldhw	it scales <matrix> - l, d, h, w it scale along x, y, z and w (weight on separate single dimension) <numeric>
m	bn scales <vector> - l, d, h, w bn scale along x, y, z and w (weight on separate single dimension) <numeric>

Details

gbp4d init a profit vector p, a length l, a depth d, a height h, and a weight w, along with associate constraints ml, md, mh and mw. gbp4d should fit it (l, d, h, w) into bn (ml, md, mh, mw) with w on weight limit constraint and l, d, h on geometry intepretation. gbp4d solver would solve

maximize sum_j=1^n p_j k_j

subject to sum_j=1^n w_j k_j leq mw and

fit (l_j, d_j, h_j) at coordinate (x_j, y_j, z_j) such that no overlap in ml x md x mh cuboid, j = 1, ..., n

and instantiate a gbp4d object with a x-axis coordinate vector x, a y-axis coordinate vector y, a z-axis coordinate vector z, a selection vector k, and an objective o.

Value

gbp4d a gbp4d instantiate with p profit, it item (x, y, z, w, l, d, h, w) position scale matrix, bn bin (l, d, h, w) scale vector, k selection, o objective, and ok an indicator of all fit or not.

See Also

Other gbp4d: gbp4d_checkr, gbp4d

gbp4d_solver_dpp_filt *gbp4d_solver_dpp_filt*

Description

solve gbp4d w.r.t select most preferable often smallest bin from bn list

Usage

```
gbp4d_solver_dpp_filt(ldhw, m)
```

Arguments

`ldhw`	it scale <matrix> - l, d, h, w it scale along x, y, z and w <numeric>
`m`	bn scale <matrix> - l, d, h, w bn scale along x, y, z and w <numeric> - l, d, h, w in row and each col is a single bn should make sure bn list are sorted via volume so that the first col is the most prefered smallest bn, and also the last col is the least prefered largest and often dominant bn should make sure no X in front of Y if bnX dominant bnY, bnX dominant bnY if all(X(l, d, h) > Y(l, d, h)) and should always prefer Y. should make sure bn such that l >= d >= h or vice versa.

Details

gbp4d_solver_dpp_filt is built on top of gbp4d_solver_dpp aims to select the most preferable bn from a list of bn that can fit all or most it

gbp4d_solver_dpp()'s objective is fit all or most it into a single given bn (l, d, h, w)

gbp4d_solver_dpp_filt()'s objective is select the most preferable given a list of bn where bn list is specified in 4xN matrix that the earlier column the more preferable

gbp4d_solver_dpp_filt() use an approx binary search and determine f w.r.t bn.n_cols where f = 1 indicate the bn being selected and only one of 1 in result returned.

ok = true if any bin can fit all it and algorithm will select smallest bn can fit all otherwise ok = false and algorithm will select a bn can maximize volume of fitted it

often recommend to make the last and least preferable bn dominate all other bn in list when design bn list, bnX dominant bnY if all(X(l, d, h) > Y(l, d, h)).

Value

gbp4q a gbp4q instantiate with p profit, it item (x, y, z, w, l, d, h, w) position scale matrix, bn bin (l, d, h, w) scale matrix, k it selection, o objective, f bn selection, and ok an indicator of all fit or not.

See Also

Other gbp4q: `gbp4q_checkr`, `gbp4q`

`gbp4d_solver_dpp_prep_create_p` *gbp4d_solver_dpp_prep_create_p*

Description

auxilium of gbp4d_solver_dpp

Usage

```
gbp4d_solver_dpp_prep_create_p(ldhw, m)
```

Arguments

`ldhw`	4xN matrix of l, d, h, w of it
`m`	4x1 vector of l, d, h, w of bn

Details

create p via ldhw and m via cluster w, cluster max(l, d, h) and area strategy

Value

p

`gbp4d_viewer` *gbp4d_viewer*

Description

gbp4d solution viewer

Usage

```
gbp4d_viewer(sn, title = NULL, subtitle = NULL)
```

Arguments

`sn`	sn gbp4d object, solution from gbp4d_solver_dpp, see gbp4d.
`title`	title <character>
`subtitle`	subtitle <character>

gbp4q *gbp4q*

Description

generalized bin packing problem in 4 dimension, a.k.a bin packing problem with weight limit.

Usage

```
gbp4q
```

Format

An object of class `C++Class` of length 1.

Details

gbp4d init a profit vector p, a length l, a depth d, a height h, and a weight w, along with associate constraints ml, md, mh and mw. gbp4d should fit it (l, d, h, w) into bn (ml, md, mh, mw) with w on weight limit constraint and l, d, h on geometry intepretation. gbp4d solver would solve

maximize sum_j=1^n p_j k_j

subject to sum_j=1^n w_j k_j leq mw and

fit (l_j, d_j, h_j) at coordinate (x_j, y_j, z_j) such that no overlap in ml x md x mh cuboid, j = 1, ..., n

and instantiate a gbp4d object with a x-axis coordinate vector x, a y-axis coordinate vector y, a z-axis coordinate vector z, a selection vector k, and an objective o.

gbp4q solver would also select the most preferred often smallest m from a list of m(l, d, h) after determine all or the higest volume set of ld can fit into one m(l, d, h) w.r.t the weight constraint.

a gbp4q class instance has 7 fields:

- p: profit of it fit into bn <vector>

created via cluster w via gbp1d, cluster max(l, d, h) and area via gbp4d_solver_dpp_main_create_p()

- it: it position and scale <matrix>

- x, y, z, w it position and w in the bin <numeric> (w hold in bn when fit it in bn)

- l, d, h, w it scale along x, y, z and w <numeric> (w of it itself)

- bn: bn scale <matrix>

- l, d, h, w bn scale along x, y, z and w <numeric>

matrix of 4 rows and each column is a single bn

should make sure bn list are sorted via volume so that the first col is the most prefered smallest bn, and also the last col is the least prefered largest and often dominant bn

should make sure no X in front of Y if bnX dominant bnY, bnX dominant bnY if all(X(l, d, h) > Y(l, d, h)) and should always prefer Y.

should make sure bn such that l >= d or vice versa.

- k: selection indicator 0, 1 on it <vector>

- f: selection indicator 0, 1, 2, 3 on bn <vector>

f in result should have no 0 and only one of 1

- o: objective achivement volumn fit in over volumn overall <numeric>

- ok: a quick indicator of all it fit into bn? <bool>

See Also

Other gbp4q: `gbp4d_solver_dpp_filt`, `gbp4q_checkr`

`gbp4q_checkr` *gbp4q_checkr*

Description

auxilium of gbp4d_solver_dpp_filt

Usage

```
gbp4q_checkr(sn)
```

Arguments

`sn`	<gbp4q> gbp4q object from gbp4d_solver_dpp_filt() solution.

Details

check fit solution is valid: no conflict between item and bin, and no conflict between each pair of item, and no conflict on weight limit.

Value

okfit? <bool>

See Also

Other gbp4q: `gbp4d_solver_dpp_filt`, `gbp4q`

gbp4q_viewer *gbp4q_viewer*

Description

gbp4q solution viewer

Usage

```
gbp4q_viewer(sn, title = NULL, subtitle = NULL)
```

Arguments

`sn`	sn gbp4q object, solution from gbp4d_solver_dpp_filt, see gbp4q.
`title`	title <character>
`subtitle`	subtitle <character>

Ktlist2d *Ktlist2d*

Description

Ktlist2d hold multiple kt for recursive fit

Usage

```
Ktlist2d
```

Format

An object of class `C++Class` of length 1.

Details

Ktlist2d hold multiple kt via consider all possible fit onto different xp and different rotation and nlimit

a Ktlist2d class instance has 4 fields:

- n: length of kt candidate position scale vector list

- kt: candidate (x, y, l, d) fit of it current investigating

- xp: candidate extreme point list after kt fit into each corresponding (x, y, l, d) position scale

- s: score of each kt fit: calculate overall extrem point residual space entropy as score, the smaller the better, since smaller entropy indicate concentrated residual space and less number of extreme point.

Note

internal cpp class use in gbp2d_solver_dpp()

See Also

Other gbp2d_it: `gbp2d_it_create_ktlist`

`Ktlist3d` *Ktlist3d*

Description

Ktlist3d hold multiple kt for recursive fit

Usage

```
Ktlist3d
```

Format

An object of class `C++Class` of length 1.

Details

Ktlist3d hold multiple kt via consider all possible fit onto different xp and different rotation and nlimit

a Ktlist3d class instance has 4 fields:

- n: length of kt candidate position scale vector list

- kt: candidate (x, y, z, l, d, h) fit of it current investigating

- xp: candidate extreme point list after kt fit into each corresponding (x, y, z, l, d, h) position scale

- s: score of each kt fit: calculate overall extrem point residual space entropy as score, the smaller the better, since smaller entropy indicate concentrated residual space and less number of extreme point.

Note

internal cpp class use in gbp3d_solver_dpp()

See Also

Other gbp3d_it: `gbp3d_it_create_ktlist`

`Ktlist4d` *Ktlist4d*

Description

Ktlist4d hold multiple kt for recursive fit

Usage

```
Ktlist4d
```

Format

An object of class `C++Class` of length 1.

Details

Ktlist4d hold multiple kt via consider all possible fit onto different xp and different rotation and nlimit

a Ktlist4d class instance has 4 fields

- n: length of kt candidate position scale vector list

- kt: candidate (x, y, z, w, l, d, h, w) fit of it current investigating

x, y, z, w - weight holding in bn when fit; l, d, h, w - weight of it itself

- xp: candidate extreme point list after kt fit into each corresponding (x, y, z, l, d, h) position scale

- s: score of each kt fit: calculate overall extrem point residual space entropy as score, the smaller the better, since smaller entropy indicate concentrated residual space and less number of extreme point.

Note

internal cpp class use in gbp4d_solver_dpp()

See Also

Other gbp4d_it: `gbp4d_it_create_ktlist`

Index