Package: enum (via r-universe)

October 31, 2026

Title Enumerate Partitions of Grid Graphs

Version 0.0.1

Description Enumerates polyomino tilings of grid graphs. Partitions an m by n grid into k connected pieces where each piece has a size within given bounds, under rook or queen contiguity rules. A translation of the Julia 'enumerator' package by Schutzman (2019) <doi:10.5281/zenodo.3467675>.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/Needs/website christopherkenny/ctktemplate

LinkingTo cli

Imports cli, igraph

Depends R (>= 4.2.0)

Language en-US

URL http://christophertkenny.com/enum/

Config/pak/sysreqs libglpk-dev libxml2-dev

Repository https://christopherkenny.r-universe.dev

Date/Publication 2026-05-01 16:15:02 UTC

RemoteUrl https://github.com/christopherkenny/enum

RemoteRef HEAD

RemoteSha ac1768987e10ed118f8060ac3a497c8e9a1f8433

Contents

enum_count_partitions 2
enum_count_partitions_graph 3
enum_partitions 4
enum_partitions_graph 5
enum_read_partitions 6

Index **8**

`enum_count_partitions` *Count grid partitions*

Description

Count the number of valid partitions of an `nrow` by `ncol` grid into `num_parts` connected pieces. Part sizes can be constrained either as a range via `min_size` and `max_size`, or as an explicit set of allowed sizes via `exact_sizes`. Prefer this over `enum_partitions()` when only the count is needed, as it avoids storing all partitions in memory.

Usage

```
enum_count_partitions(
  nrow,
  ncol,
  num_parts,
  min_size = NULL,
  max_size = NULL,
  exact_sizes = NULL,
  contiguity = c("rook", "queen"),
  progress = TRUE
)
```

Arguments

`nrow`	Integer. Number of rows in the grid.
`ncol`	Integer. Number of columns in the grid.
`num_parts`	Integer. Number of parts to partition the grid into.
`min_size`	Integer or `NULL`. Minimum number of cells per part. Must be supplied together with `max_size`; mutually exclusive with `exact_sizes`.
`max_size`	Integer or `NULL`. Maximum number of cells per part. Must be supplied together with `min_size`; mutually exclusive with `exact_sizes`.
`exact_sizes`	Integer vector or `NULL`. The exact set of allowed part sizes. For example, `exact_sizes = c(4, 8)` allows parts of size 4 or 8 only. Mutually exclusive with `min_size`/`max_size`.
`contiguity`	Character. Either `"rook"` (default) for edge-adjacency or `"queen"` for edge-and-corner adjacency.
`progress`	Logical. Whether to report enumeration progress. Default `TRUE`.

Value

A single integer giving the number of valid partitions.

Examples

```
# Count partitions of a 2x3 grid into 2 rook-connected parts of size 3
enum_count_partitions(2, 3, num_parts = 2, min_size = 3, max_size = 3)

# Count with exact sizes
enum_count_partitions(4, 4, num_parts = 3, exact_sizes = c(4, 8))
```

`enum_count_partitions_graph`
Count partitions of a graph

Description

Count the number of valid partitions of an arbitrary graph into `num_parts` connected subgraphs. Part sizes can be constrained either as a range via `min_size` and `max_size`, or as an explicit set of allowed sizes via `exact_sizes`. Prefer this over `enum_partitions_graph()` when only the count is needed, as it avoids storing all partitions in memory.

Usage

```
enum_count_partitions_graph(
  graph,
  num_parts,
  min_size = NULL,
  max_size = NULL,
  exact_sizes = NULL,
  progress = TRUE
)
```

Arguments

`graph`	An `igraph` or `adj` object. Must be undirected and connected. An `adj` object is a list where element `i` contains the integer indices (1-indexed) of vertices adjacent to vertex `i`.
`num_parts`	Integer. Number of parts to partition the graph into.
`min_size`	Integer or `NULL`. Minimum number of vertices per part. Must be supplied together with `max_size`; mutually exclusive with `exact_sizes`.
`max_size`	Integer or `NULL`. Maximum number of vertices per part. Must be supplied together with `min_size`; mutually exclusive with `exact_sizes`.
`exact_sizes`	Integer vector or `NULL`. The exact set of allowed part sizes. Mutually exclusive with `min_size`/`max_size`.
`progress`	Logical. Whether to report enumeration progress. Default `TRUE`.

Value

A single integer giving the number of valid partitions.

Examples

```
g <- igraph::make_ring(6)
enum_count_partitions_graph(g, num_parts = 2, min_size = 3, max_size = 3)
```

`enum_partitions` *Enumerate grid partitions*

Description

Enumerate all partitions of an `nrow` by `ncol` grid into `num_parts` connected pieces. Part sizes can be constrained either as a range via `min_size` and `max_size`, or as an explicit set of allowed sizes via `exact_sizes`. Exactly one of these two forms must be supplied.

Usage

```
enum_partitions(
  nrow,
  ncol,
  num_parts,
  min_size = NULL,
  max_size = NULL,
  exact_sizes = NULL,
  contiguity = c("rook", "queen"),
  file = NULL,
  progress = TRUE
)
```

Arguments

`nrow`	Integer. Number of rows in the grid.
`ncol`	Integer. Number of columns in the grid.
`num_parts`	Integer. Number of parts to partition the grid into.
`min_size`	Integer or `NULL`. Minimum number of cells per part. Must be supplied together with `max_size`; mutually exclusive with `exact_sizes`.
`max_size`	Integer or `NULL`. Maximum number of cells per part. Must be supplied together with `min_size`; mutually exclusive with `exact_sizes`.
`exact_sizes`	Integer vector or `NULL`. The exact set of allowed part sizes. For example, `exact_sizes = c(4, 8)` allows parts of size 4 or 8 only. Mutually exclusive with `min_size`/`max_size`.
`contiguity`	Character. Either `"rook"` (default) for edge-adjacency or `"queen"` for edge-and-corner adjacency.

file	Character or NULL. If a file path is provided, partitions are written to a binary file instead of returned as a matrix. Each partition is stored as nrow * ncol consecutive 32-bit integers. Use enum_read_partitions() to read the file back into R. When file is not NULL, the function returns the partition count invisibly.
progress	Logical. Whether to report enumeration progress. Default TRUE.

Value

When file is NULL (default), an integer matrix where each column is a partition and each row corresponds to a cell of the grid in row-major order. Cell values are integers from 1 to num_parts indicating part membership. When file is a path, returns the number of partitions written, invisibly.

Examples

```
# Partition a 2x3 grid into 2 rook-connected parts of size 3
enum_partitions(2, 3, num_parts = 2, min_size = 3, max_size = 3)

# Allow only parts of size 4 or 8 (exact sizes)
enum_partitions(4, 4, num_parts = 3, exact_sizes = c(4, 8))
```

---

enum_partitions_graph *Enumerate partitions of a graph*

Description

Enumerate all partitions of an arbitrary graph into num_parts connected subgraphs. Part sizes can be constrained either as a range via min_size and max_size, or as an explicit set of allowed sizes via exact_sizes. Accepts igraph objects and adj objects (1-indexed adjacency lists).

Usage

```
enum_partitions_graph(
  graph,
  num_parts,
  min_size = NULL,
  max_size = NULL,
  exact_sizes = NULL,
  file = NULL,
  progress = TRUE
)
```

Arguments

graph	An igraph or adj object. Must be undirected and connected. An adj object is a list where element i contains the integer indices (1-indexed) of vertices adjacent to vertex i.
num_parts	Integer. Number of parts to partition the graph into.

min_size	Integer or NULL. Minimum number of vertices per part. Must be supplied together with max_size; mutually exclusive with exact_sizes.
max_size	Integer or NULL. Maximum number of vertices per part. Must be supplied together with min_size; mutually exclusive with exact_sizes.
exact_sizes	Integer vector or NULL. The exact set of allowed part sizes. Mutually exclusive with min_size/max_size.
file	Character or NULL. If a file path is provided, partitions are written to a binary file instead of returned as a matrix. Each partition is stored as one 32-bit integer per vertex. Use enum_read_partitions() to read the file back into R. When file is not NULL, the function returns the partition count invisibly.
progress	Logical. Whether to report enumeration progress. Default TRUE.

Value

When file is NULL (default), an integer matrix where each column is a partition and each row corresponds to a vertex (in igraph vertex order). Cell values are integers from 1 to num_parts indicating part membership. When file is a path, returns the number of partitions written, invisibly.

Examples

```
g <- igraph::make_ring(6)
enum_partitions_graph(g, num_parts = 2, min_size = 3, max_size = 3)
```

---

enum_read_partitions *Read partitions from a binary file*

Description

Read the binary file produced by enum_partitions() or enum_partitions_graph() when called with the file argument.

Usage

```
enum_read_partitions(file, skip = 0L, n = NULL)
```

Arguments

file	Character. Path to the binary file.
skip	Integer. Number of partitions to skip before reading. Defaults to 0.
n	Integer or NULL. Number of partitions to read. If NULL (default), reads all remaining partitions.

Value

An integer matrix with one row per cell/vertex and one column per partition read. Cell values are integers from 1 to the number of parts indicating part membership.

Examples

```
tmp <- tempfile()
enum_partitions(3, 3, num_parts = 3, min_size = 3, max_size = 3, file = tmp)
enum_read_partitions(tmp)
enum_read_partitions(tmp, skip = 5)
enum_read_partitions(tmp, skip = 2, n = 3)
```

Index

enum_count_partitions, 2
enum_count_partitions_graph, 3
enum_partitions, 4
enum_partitions(), *2*, *6*
enum_partitions_graph, 5
enum_partitions_graph(), *3*, *6*
enum_read_partitions, 6
enum_read_partitions(), *5, 6*