Package 'dress.graph'

May 8, 2026

Type Package

Title DRESS - A Continuous Framework for Structural Graph Refinement

Version 0.8.3

Date 2026-05-04

Maintainer Eduar Castrillo Velilla <velicast@outlook.com>

Description DRESS is a deterministic, parameter-free framework for continuous structural graph refinement. It iterates a nonlinear dynamical system on real-valued edge similarities and produces a graph fingerprint as a sorted edge-value vector once the iteration reaches a prescribed stopping criterion. The resulting
fingerprint is self-contained, isomorphism-invariant by construction, reproducible across vertex labelings under the reference implementation, numerically robust in practice, and efficient to compute with straightforward parallelization and distribution.

License MIT + file LICENSE

URL https://github.com/velicast/dress-graph,
https://velicast.github.io/dress-graph/

BugReports https://github.com/velicast/dress-graph/issues

Encoding UTF-8

Depends R (>= 3.5.0)

NeedsCompilation yes

SystemRequirements OpenMP, CUDA Toolkit (optional, for GPU acceleration)

Author Eduar Castrillo Velilla [aut, cre] (ORCID:
<https://orcid.org/0009-0005-2492-0957>)

Repository CRAN

Date/Publication 2026-05-05 11:20:17 UTC

Contents

---

cuda *CUDA-accelerated DRESS functions.*

Description

An environment containing GPU-accelerated versions of `fit`, `delta_fit`, and `nabla_fit` with identical signatures. Switch from CPU to GPU by prefixing calls with `cuda$`.

Details

The `cuda` environment provides:

`cuda$fit(...)` GPU-accelerated `fit`. Same arguments and return value.

`cuda$delta_fit(...)` GPU-accelerated `delta_fit`. Same arguments and return value.

`cuda$nabla_fit(...)` GPU-accelerated `nabla_fit`. Same arguments and return value.

CUDA support requires rebuilding the package with `DRESS_CUDA=1`. If CUDA is not available, calling either function raises an error.

Examples

```
## Not run:
  # CPU
  r1 <- fit(4L, c(0L,1L,2L,2L), c(1L,2L,0L,3L))

  # CUDA -- same signature
  r2 <- cuda$fit(4L, c(0L,1L,2L,2L), c(1L,2L,0L,3L))

## End(Not run)
```

delta_fit *Compute the Delta-k-DRESS Histogram*

Description

Compute the Δ^k-DRESS histogram by exhaustively removing all k-vertex subsets and measuring the change in edge similarity values. The result is a sparse exact histogram of edge DRESS values pooled across all $\binom{N}{k}$ subgraphs.

Usage

```
delta_fit(n_vertices, sources, targets, weights = NULL,
                vertex_weights = NULL, k = 0L, variant = DRESS_UNDIRECTED,
                max_iterations = 100L, epsilon = 1e-6,
                n_samples = 0L, seed = 0L,
                precompute = FALSE,
                keep_multisets = FALSE, compute_histogram = TRUE)
```

Arguments

n_vertices	Integer. Number of vertices (vertex ids must be in 0 .. n_vertices - 1).
sources	Integer vector of length E – edge source endpoints (0-based).
targets	Integer vector of length E – edge target endpoints (0-based).
weights	Numeric vector of length E – edge weights, or NULL for unweighted (all ones). Default NULL.
vertex_weights	Numeric vector of length N – vertex weights, or NULL (all ones). Default NULL.
k	Integer. Number of vertices to remove per subset (0 = original graph, default 0).
variant	Graph variant (default DRESS_UNDIRECTED). One of DRESS_UNDIRECTED (0), DRESS_DIRECTED (1), DRESS_FORWARD (2), DRESS_BACKWARD (3).
max_iterations	Maximum number of fitting iterations per subgraph (default 100).
epsilon	Convergence threshold (default 1e-6).
n_samples	Integer. Number of random subgraphs to sample (0 = exhaustive enumeration, default 0).
seed	Integer. Random seed for sampling (default 0).
precompute	Logical. Pre-compute common-neighbor intercept index for faster iteration at the cost of more memory (default FALSE).
keep_multisets	Logical. If TRUE, return per-subgraph edge DRESS values in a C(N,k) x E matrix (NaN for removed edges). Default FALSE.
compute_histogram	Logical. If FALSE, skip histogram construction (default TRUE).

Value

A list with components:

histogram	Data frame with columns value and count, containing the exact sparse histogram entries sorted by value.
multisets	Matrix (C(N,k) x E) of per-subgraph edge DRESS values (only present when keep_multisets = TRUE; NaN = removed edge).
num_subgraphs	Integer – C(N,k) (only present when keep_multisets = TRUE).

See Also

fit for the standard DRESS computation.

Examples

```
# Triangle K3, delta-1: remove 1 vertex at a time
res <- delta_fit(3L, c(0L, 1L, 2L), c(1L, 2L, 0L), k = 1L)
res$histogram

# K4, delta-0 (original graph)
res0 <- delta_fit(4L, c(0L,0L,0L,1L,1L,2L), c(1L,2L,3L,2L,3L,3L))
sum(res0$histogram$count)  # 6 edge values
```

DRESS *Persistent DRESS Graph Object*

Description

Create a persistent DRESS graph that stays alive across multiple fit / get (virtual-edge query) calls without rebuilding the graph each time.

Usage

```
DRESS(n_vertices, sources, targets, weights = NULL,
      vertex_weights = NULL, variant = DRESS_UNDIRECTED,
      precompute_intercepts = FALSE)
```

Arguments

n_vertices	Integer. Number of vertices (vertex ids must be in 0 .. n_vertices - 1).
sources	Integer vector of length E – edge source endpoints (0-based).
targets	Integer vector of length E – edge target endpoints (0-based).
weights	Optional numeric vector of length E – per-edge weights. NULL (default) gives every edge weight 1.
vertex_weights	Optional numeric vector of length N – per-vertex weights. NULL (default) gives every vertex weight 1.

`variant` Graph variant (default `DRESS_UNDIRECTED`). One of `DRESS_UNDIRECTED` (0), `DRESS_DIRECTED` (1), `DRESS_FORWARD` (2), `DRESS_BACKWARD` (3).

`precompute_intercepts`
Logical. Pre-compute common-neighbor index for faster iteration at the cost of more memory (default `FALSE`).

Value

An environment of class `"DRESS"` with the following methods:

`$fit(max_iterations = 100L, epsilon = 1e-6)`
Run iterative fitting. Returns a list with `iterations` and `delta`.

`$get(u, v, max_iterations = 100L, epsilon = 1e-6, edge_weight = 1.0)`
Query the DRESS value for an existing or virtual edge between vertices `u` and `v`.

`$result()` Extract current results as a list with `sources`, `targets`, `edge_dress`, `edge_weight`, and `vertex_dress`.

`$close()` Explicitly free the underlying C graph. Called automatically by the garbage collector if not invoked manually.

References

E. Castrillo, E. Leon, J. Gomez. Dynamic Structural Similarity on Graphs. arXiv:1805.01419, 2018.

Examples

```
g <- DRESS(4L, c(0L,1L,2L,2L), c(1L,2L,0L,3L))
g$fit(100L, 1e-6)
g$get(0L, 3L, 100L, 1e-6, 1.0)
r <- g$result()
g$close()
```

---

dress_version *DRESS Library Version*

Description

Return the version string of the underlying DRESS C library.

Usage

```
dress_version()
```

Value

A character scalar, e.g. `"0.7.0"`.

Examples

```
dress_version()
```

`fit` *Compute DRESS Edge Similarity on Graphs*

Description

Build a DRESS graph from an edge list and run iterative fitting to compute per-edge structural similarity values.

Usage

```
fit(n_vertices, sources, targets, weights = NULL,
          vertex_weights = NULL, variant = DRESS_UNDIRECTED,
          max_iterations = 100L, epsilon = 1e-6,
          precompute_intercepts = FALSE)


DRESS_UNDIRECTED
DRESS_DIRECTED
DRESS_FORWARD
DRESS_BACKWARD
```

Arguments

`n_vertices`	Integer. Number of vertices (vertex ids must be in `0 .. n_vertices - 1`).
`sources`	Integer vector of length E – edge source endpoints (0-based).
`targets`	Integer vector of length E – edge target endpoints (0-based).
`weights`	Optional numeric vector of length E – per-edge weights. `NULL` (default) gives every edge weight 1.
`vertex_weights`	Optional numeric vector of length N – per-vertex weights. `NULL` (default) gives every vertex weight 1.
`variant`	Graph variant (default `DRESS_UNDIRECTED`). One of `DRESS_UNDIRECTED` (0), `DRESS_DIRECTED` (1), `DRESS_FORWARD` (2), `DRESS_BACKWARD` (3).
`max_iterations`	Maximum number of fitting iterations (default 100).
`epsilon`	Convergence threshold – stop when the max per-edge change falls below this value (default 1e-6).
`precompute_intercepts`	Logical. Pre-compute common-neighbor index for faster iteration at the cost of more memory (default `FALSE`).

Value

A list with components:

`sources`	Integer vector [E] – edge source endpoints (0-based).
`targets`	Integer vector [E] – edge target endpoints (0-based).
`edge_dress`	Numeric vector [E] – DRESS similarity per edge.
`edge_weight`	Numeric vector [E] – variant-specific weight.
`vertex_dress`	Numeric vector [N] – per-vertex norm.
`iterations`	Integer – number of iterations performed.
`delta`	Numeric – final max per-edge change.
`vertex_weights`	Numeric vector [N] – vertex weights (if provided).

References

E. Castrillo, E. Leon, J. Gomez. Dynamic Structural Similarity on Graphs. arXiv:1805.01419, 2018.

Examples

```
# Triangle + pendant: 0-1, 1-2, 2-0, 2-3
res <- fit(4L, c(0L,1L,2L,2L), c(1L,2L,0L,3L))
res$edge_dress
```

`mpi` *MPI-distributed DRESS functions.*

Description

An environment containing MPI-distributed versions of `delta_fit` and `nabla_fit`. Switch from CPU to MPI by prefixing calls with `mpi$`.

Details

The `mpi` environment provides:

`mpi$delta_fit(...)` MPI-distributed `delta_fit` (CPU backend). Same arguments plus `comm_f`.

`mpi$nabla_fit(...)` MPI-distributed `nabla_fit` (CPU backend). Same arguments plus `comm_f`.

`mpi$cuda$delta_fit(...)` MPI-distributed `delta_fit` (CUDA backend). Each rank runs GPU-accelerated DRESS.

`mpi$cuda$nabla_fit(...)` MPI-distributed `nabla_fit` (CUDA backend).

`mpi$omp$delta_fit(...)` MPI-distributed `delta_fit` (OpenMP backend).

`mpi$omp$nabla_fit(...)` MPI-distributed `nabla_fit` (OpenMP backend).

MPI support requires rebuilding the package with `DRESS_MPI` (auto-detected when `mpicc` is available).

Examples

```
## Not run:
  # CPU
  r1 <- delta_fit(4L, c(0L,1L,2L,2L), c(1L,2L,0L,3L), k = 1L)

  # MPI -- same signature, distributed
  r2 <- mpi$delta_fit(4L, c(0L,1L,2L,2L), c(1L,2L,0L,3L), k = 1L)

  # MPI + CUDA
  r3 <- mpi$cuda$delta_fit(4L, c(0L,1L,2L,2L), c(1L,2L,0L,3L), k = 1L)

## End(Not run)
```

`nabla_fit` *Compute the Nabla-k-DRESS Histogram*

Description

Compute the ∇^k-DRESS histogram by enumerating all $P(N, k)$ ordered k-tuples of vertices, marking each tuple with distinct generic vertex weights, running DRESS on each marked graph, and pooling all converged edge values into a sparse exact histogram.

Usage

```
nabla_fit(n_vertices, sources, targets, weights = NULL,
                vertex_weights = NULL, k = 0L, variant = DRESS_UNDIRECTED,
                max_iterations = 100L, epsilon = 1e-6,
                n_samples = 0L, seed = 0L,
                precompute = FALSE,
                keep_multisets = FALSE, compute_histogram = TRUE)
```

Arguments

`n_vertices`	Integer. Number of vertices (vertex ids must be in `0 .. n_vertices - 1`).
`sources`	Integer vector of length E – edge source endpoints (0-based).
`targets`	Integer vector of length E – edge target endpoints (0-based).
`weights`	Numeric vector of length E – edge weights, or `NULL` for unweighted (all ones). Default `NULL`.
`vertex_weights`	Numeric vector of length N – vertex weights, or `NULL` (all ones). Default `NULL`.
`k`	Integer. Number of vertices to individualize per tuple (0 = original graph, default 0).
`variant`	Graph variant (default `DRESS_UNDIRECTED`). One of `DRESS_UNDIRECTED` (0), `DRESS_DIRECTED` (1), `DRESS_FORWARD` (2), `DRESS_BACKWARD` (3).
`max_iterations`	Maximum number of fitting iterations per marked graph (default 100).
`epsilon`	Convergence threshold (default 1e-6).

n_samples	Integer. Number of random tuples to sample (0 = exhaustive enumeration, default 0).
seed	Integer. Random seed for sampling (default 0).
precompute	Logical. Pre-compute common-neighbor intercept index for faster iteration at the cost of more memory (default FALSE).
keep_multisets	Logical. If TRUE, return per-tuple edge DRESS values in a P(N,k) x E matrix. Default FALSE.
compute_histogram	Logical. If FALSE, skip histogram construction (default TRUE).

Value

A list with components:

histogram	Data frame with columns value and count, containing the exact sparse histogram entries sorted by value.
multisets	Matrix (P(N,k) x E) of per-tuple edge DRESS values (only present when keep_multisets = TRUE).
num_tuples	Integer – P(N,k) (only present when keep_multisets = TRUE).

See Also

fit for the standard DRESS computation. delta_fit for the vertex-deletion variant.

Examples

```
# Triangle K3, nabla-1: individualize 1 vertex at a time
res <- nabla_fit(3L, c(0L, 1L, 2L), c(1L, 2L, 0L), k = 1L)
res$histogram

# K4, nabla-0 (original graph, no individualization)
res0 <- nabla_fit(4L, c(0L,0L,0L,1L,1L,2L), c(1L,2L,3L,2L,3L,3L))
sum(res0$histogram$count)  # 6 edge values
```

---

omp *OpenMP-Parallel DRESS Backend*

Description

An environment providing OpenMP-parallel versions of DRESS functions. Use omp$fit(...), omp$delta_fit(...), and omp$nabla_fit(...) for OpenMP edge-parallel fitting, subgraph-parallel delta fitting, and tuple-parallel nabla fitting.

Same parameters as fit, delta_fit, and nabla_fit.

Usage

```
omp
```

Examples

```
## Not run:
  r <- omp$fit(4L, c(0L,1L,2L,2L), c(1L,2L,0L,3L))
  d <- omp$delta_fit(3L, c(0L,1L,0L), c(1L,2L,2L), k = 1L)
  n <- omp$nabla_fit(3L, c(0L,1L,0L), c(1L,2L,2L), k = 1L)

## End(Not run)
```

Index