

Package: CoopPostman (via r-universe)

October 15, 2026

Type Package

Title Optimal Routing and Cost Allocation for the Chinese Postman Game

Version 0.1.0

Description Solves the Chinese Postman Problem (CPP) using Dijkstra's algorithm for shortest paths, Edmonds' Blossom algorithm (1965) for minimum weight perfect matching, and Fleury's algorithm for Eulerian circuit generation. Additionally, computes the cooperative gamma-rule cost allocation proposed by Hamers, Klijn, Suijs, and Tijs (1999) <doi:10.1016/S0377-2217(98)00346-6> for weakly Eulerian networks.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Imports igraph, graphics

Suggests ggplot2

NeedsCompilation no

Author Maria Gabriela Perez [aut, cre]

Maintainer Maria Gabriela Perez <maria.g.perezs@udc.es>

Repository <https://cran.r-universe.dev>

Date/Publication 2026-10-15 17:10:16 UTC

RemoteUrl <https://github.com/cran/CoopPostman>

RemoteRef HEAD

RemoteSha d9f39d87e030ab3f3bb4209c97461df5b8d4c82d

Contents

calcular_gamma	2
duplicate_shortest_path	2
fleury_path	3
get_dijkstra_matrix	3

get_optimal_matching	4
odd_vertices	4
plot_matching	5
solve_cpp	5
Index	7

calcular_gamma	<i>Cooperative Gamma-Rule Cost Allocation (Hamers et al., 1999)</i>
----------------	---

Description

Calculates the cooperative game cost allocation vector gamma for weakly Eulerian networks.

Usage

calcular_gamma(aristas, nodos, deposito)

Arguments

- aristas A data.frame with columns from, to, and weight.
- nodos A data.frame with a column name.
- deposito Identifier of the depot vertex.

Value

A list containing gamma allocation and bridge followers.

duplicate_shortest_path	<i>Duplicate Shortest Path Edges in the Graph</i>
-------------------------	---

Description

Duplicates edges along the shortest path between two vertices to make the graph Eulerian.

Usage

duplicate_shortest_path(grafo, from, to)

Arguments

- grafo An igraph object representing the network.
- from Starting vertex name or index.
- to Destination vertex name or index.

Value

An updated igraph object containing duplicated edges.

fleury_path	<i>Fleury's Algorithm for Eulerian Circuit Generation</i>
-------------	---

Description

Computes an Eulerian circuit on an Eulerian multigraph using Fleury's algorithm.

Usage

```
fleury_path(grafo)
```

Arguments

grafo An Eulerian igraph object.

Value

A character vector of visited vertices in order.

get_dijkstra_matrix	<i>Shortest Path Distance Matrix for Odd Vertices</i>
---------------------	---

Description

Computes the all-pairs shortest path distance matrix between odd-degree vertices using Dijkstra's algorithm.

Usage

```
get_dijkstra_matrix(grafo, nodos_impares)
```

Arguments

grafo An igraph object representing the network.
nodos_impares A character or numeric vector of odd-degree vertices.

Value

A symmetric numeric distance matrix.

get_optimal_matching	<i>Minimum Weight Perfect Matching via Edmonds' Blossom Algorithm</i>
----------------------	---

Description

En R implementation of Edmonds' Blossom algorithm (1965) for minimum weight perfect matching on complete graphs. Uses primal-dual potential adjustments, LCA detection, and blossom contraction/expansion.

Usage

```
get_optimal_matching(matriz_dist)
```

Arguments

matriz_dist A symmetric numeric matrix of shortest path distances.

Value

A data.frame with columns n1, n2, and coste.

odd_vertices	<i>Identify Odd Degree Vertices</i>
--------------	-------------------------------------

Description

Finds all vertices with an odd degree in a given undirected graph.

Usage

```
odd_vertices(grafo)
```

Arguments

grafo An igraph object representing the network.

Value

A character vector with the names of vertices with odd degree.

plot_matching

Visualize Complete Distance Graph and Blossom Matching

Description

Draws the complete graph of shortest-path distances between odd vertices.

Usage

```
plot_matching(matriz_dist, matching_df, main = "Optimal Blossom Matching")
```

Arguments

matriz_dist	A square distance matrix.
matching_df	A data.frame from get_optimal_matching().
main	Optional title.

Value

Invisibly returns the complete igraph object.

solve_cpp

Solve the Chinese Postman Problem (CPP)

Description

Master pipeline that executes full CPP resolution: odd-degree detection, shortest-path distance graph creation, minimum-weight matching, edge duplication, and Fleury Eulerian tour construction.

Usage

```
solve_cpp(grafo)
```

Arguments

grafo	An undirected weighted igraph object.
-------	---------------------------------------

Value

A list with the optimization results, including odd vertices, repair cost, total cost, matching pairs, and Eulerian route.

Examples

```
g <- igraph::make_ring(4)
igraph::E(g)$weight <- c(10, 20, 15, 25)
g <- igraph::add_edges(g, c(1, 3), attr = list(weight = 5))
solve_cpp(g)
```

Index

calcular_gamma, [2](#)
duplicate_shortest_path, [2](#)
fleury_path, [3](#)
get_dijkstra_matrix, [3](#)
get_optimal_matching, [4](#)
odd_vertices, [4](#)
plot_matching, [5](#)
solve_cpp, [5](#)