Package 'warehouseTools'

June 22, 2025

Type Package

Title Heuristics for Solving the Traveling Salesman Problem in Warehouse Layouts

Version 0.1.3

Description Heuristic methods to solve the routing problems in a warehouse management. Package includes several heuristics such as the Midpoint, Return, S-Shape and Semi-Optimal Heuristics for designation of the picker's route in order picking. The heuristics aim to provide the acceptable travel distances while considering warehouse layout constraints such as aisles and shelves. It also includes implementation of the COPRAS (COmplex PRoportional ASsessment) method for supporting selection of locations to be visited by the picker in shared storage systems. The package is designed to facilitate more efficient warehouse routing and logistics operations.
see:
Bartholdi, J. J., Hackman, S. T. (2019). ``WAREHOUSE & DISTRIBUTION SCIENCE. Release 0.98.1."
The Supply Chain & Logistics Institute. H. Milton Stewart School of Industrial and Systems Engineering.
Georgia Institute of Technology.
<https://www.warehouse-science.com/book/editions/wh-sci-0.98.1.pdf>.

Depends R (>= 3.5.0)

License GPL-3

Imports dplyr,clusterSim

Suggests graphics,testthat (>= 2.1.0)

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Andrzej Dudek [aut, cre] (ORCID:
<https://orcid.org/0000-0002-4943-8703>),
Krzysztof Dmytrów [aut] (ORCID:
<https://orcid.org/0000-0001-7657-6063>)

Maintainer Andrzej Dudek <andrzej.dudek@ue.wroc.pl>

Repository CRAN

Date/Publication 2025-06-22 14:00:02 UTC

Contents

`chart_warehouse_route` *Plot Warehouse Route with Coordinates and Arcs usually returned by heuristic*

Description

This function visualizes a warehouse layout and the corresponding route with arcs representing connections between different coordinates (locations). It allows customization of point markers, line colors, line widths, and text labels.

Usage

```
chart_warehouse_route(
  coordinates,
  arcs,
  pch = 18,
  col_warehouse_lines = "gray",
  lwd_warehouse_lines = 1,
  col_arcs_lines = "darkblue",
  lwd_arcs_lines = 2.5,
  text_cex = 0.7,
  text_col = "darkorange",
  ...
)
```

Arguments

`coordinates`	A matrix or data frame representing the coordinates (x, y) of the warehouse locations.
`arcs`	A matrix or data frame representing the arcs (connections) between locations. Each row should contain four columns: x_from, y_from, x_to, and y_to, with an optional fifth column for arc labels.

pch	The plotting character to use for the points in the plot (default is 18).
col_warehouse_lines	The color of the warehouse lines (default is "gray").
lwd_warehouse_lines	The line width for the warehouse lines (default is 1).
col_arcs_lines	The color of the arcs that represent duplicate routes (default is "darkblue").
lwd_arcs_lines	The line width for the arcs representing duplicate routes (default is 2.5).
text_cex	Numeric value indicating the size of the text labels for arcs (default is 0.7).
text_col	The color of the text labels for arcs (default is "darkorange").
...	Additional graphical parameters passed to the underlying plot function.

Details

The function visualizes a route through a warehouse using arcs that connect different coordinates. It highlights duplicate arcs using different line colors and line widths. The function also supports text labels for arcs, with customizable text size and color. The user can pass additional graphical parameters through the '...' argument, which are forwarded to the 'plot' function.

#'

Value

The function produces a plot of the warehouse layout with arcs representing connections between locations. It does not return any value.

Author(s)

Krzysztof Dmytrów <krzysztof.dmytrow@usz.edu.pl> [aut] ORCID: 0000-0001-7657-6063

Andrzej Dudek <andrzej.dudek@ue.wroc.pl> [aut, cre] ORCID: 0000-0002-4943-8703

References

Le-Duc, T. (2005). *Design and Control of Efficient Order*. Erasmus Research Institute of Management (ERIM).

Ratliff, H. D., & Rosenthal, A. S. (1983). Order-Picking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem. *Operations Research, 31*(3), 507–521. doi:10.1287/opre.31.3.507

Examples

```
# Example usage
coordinates <- matrix(c(1, 1, 2, 2, 3, 3), ncol = 2)
arcs <- matrix(c(1, 1, 2, 2, 1, 2, 2, 3, 3, 2), ncol = 5, byrow = TRUE)
chart_warehouse_route(coordinates, arcs, pch = 19, col_warehouse_lines = "red",
lwd_warehouse_lines = 2)
```

`copras_assignment` *COPRAS Method for Selection of Locations in a Warehouse*

Description

This function allows for selection of locations in a warehouse to be visited by the picker during order picking using the COPRAS (COmplex PRoportional ASsessment) method. The method applies weighted criteria such as storage time, distance from the I/O (depot) point, degree of demand satisfaction, degree of demand satisfaction in full packages, and the number of other items on the pick list in the neighborhood of analyzed location to determine the selection of locations that satisfies the given take-out strategy.

Usage

```
copras_assignment(goods_and_locations, weights = c(rep(0.2, 5)))
```

Arguments

`goods_and_locations`	A data frame containing information about goods and their locations. It should include at least the following columns: ‘product‘, ‘location‘, ‘storage_time‘, ‘distance‘, ‘demand_ratio‘, ‘full_packages_demand_ratio‘, and ‘neighbors‘.
`weights`	A numeric vector of weights for the criteria used in the COPRAS method. The default is ‘c(rep(0.2, 5))‘, representing the weights for storage time, distance (distance from the I/O point), demand ratio (degree of demand satisfaction), full packages demand ratio (degree of demand satisfaction in full packages), and number of neighbors (number of other items on the pick list in the neighborhood of analyzed location), respectively.

Details

The function normalizes the criteria for each product and multiplies them by the provided weights. It then computes two scores, ‘si_plus‘ (positive criteria) and ‘si_minus‘ (negative criteria), which are used to calculate the COPRAS score (‘q‘). The function selects the best locations based on the highest COPRAS score while ensuring that the cumulative demand ratio for the selected locations meets or exceeds 1 (i.e., satisfies demand).

Value

A list of results for each product, where each result includes: - ‘normalized_and_weighted‘: A data frame containing the normalized and weighted values for the criteria, along with the COPRAS scores (‘q‘) and rankings (‘rank‘) for each location. - ‘copras‘: A data frame containing the selected locations for each product based on the COPRAS method, with columns ‘product‘, ‘location‘, ‘rank‘, and ‘demand_ratio‘.

Author(s)

Andrzej Dudek <andrzej.dudek@ue.wroc.pl> [aut, cre] ORCID: 0000-0002-4943-8703

Krzysztof Dmytrów <krzysztof.dmytrow@usz.edu.pl> [aut] ORCID: 0000-0001-7657-6063

References

Gudehus, T., & Kotzab, H. (2012). *Comprehensive Logistics*. Springer Berlin Heidelberg. doi:10.1007/9783642243677

Zavadskas, E. K., Kaklauskas, A., & Šarka, V. (1994). The new method of multicriteria complex proportional assessment projects. In E. K. Zavadskas & P. Linnert (Eds.), *Technological and economic development of economy. Volume 3. Business Management* (pp. 131–140). Vilnius: „Technika".

Examples

```
# Assuming `goods_and_locations` is a data frame with appropriate columns
scenario2 <- generate_sample_goods_and_locatons_scenario(warehouse_height = 10,
warehouse_width = 10,nr_goods = 5)
copras <- copras_assignment(scenario2)
```

`create_arcs` *Create Arcs from Coordinates*

Description

This internal function generates a graph representation from a set of warehouse coordinates by creating arcs (edges) between valid points.

Usage

```
create_arcs(coordinates)
```

Arguments

`coordinates`	A matrix or data frame with two columns representing the x and y coordinates of warehouse locations.

Details

The function identifies valid arcs between coordinate pairs based on specific rules: - Arcs are created only for points that are aligned either horizontally or vertically. - Horizontal arcs are added only if they are on the warehouse boundaries (e.g., ground level or the top level). - Vertical arcs are created if there are no intermediate points between the two coordinates in the same column. - The distance of each arc is calculated based on the Manhattan distance.

The output graph is represented as a matrix, where each row corresponds to an arc and contains: - `x_from`: x-coordinate of the starting point. - `y_from`: y-coordinate of the starting point. - `x_to`:

x-coordinate of the ending point. - `y_to`: y-coordinate of the ending point. - `distance`: Length of the arc.

Value

A matrix with columns `x_from`, `y_from`, `x_to`, `y_to`, and `distance`, representing the arcs of the graph.

`dd_heuristic` *Semi-Optimal Heuristic for the Picker's Route Designation*

Description

This function applies a semi-optimal heuristic for warehouse routing problems by optimizing aisle configurations and creating a feasible tour graph.

Usage

```
dd_heuristic(coordinates, arcs = NULL)
```

Arguments

`coordinates`	A data frame or matrix with columns 'x' and 'y' representing the coordinates of warehouse locations.
`arcs`	A matrix representing the initial arcs in the warehouse graph. If 'NULL', the arcs will be generated using the 'create_arcs' function.

Details

The heuristic optimizes aisle usage by identifying empty aisles and minimizing the loss associated with routing through the warehouse. It performs the following steps:

- Calculates the warehouse height and width based on coordinates.
- Identifies and isolates empty aisles.
- Evaluates configurations using a binary vector approach and selects the configuration with the minimum routing loss.
- Adjusts arcs to create a semi-optimal tour graph.

The algorithm returns an optimized graph that balances routing efficiency and warehouse constraints.

Value

A matrix representing the optimized warehouse graph, including the routing paths and IO points.

Author(s)

Krzysztof Dmytrów <krzysztof.dmytrow@usz.edu.pl> [aut] ORCID: 0000-0001-7657-6063

Andrzej Dudek <andrzej.dudek@ue.wroc.pl> [aut, cre] ORCID: 0000-0002-4943-8703

References

Dmytrów, K. (2022). Analytical and simulation determination of order picking time in a low storage warehouse for shared storage systems. Operations Research and Decisions, 32(2), 34–51. doi:10.37190/ord220203

Le-Duc, T. (2005). Design and Control of Efficient Order. Erasmus Research Institute of Management (ERIM).

Tarczyński, G. (2012). Analysis of the Impact of Storage Parameters and the Size of Orders on the Choice of the Method for Routing Order Picking. Operations Research and Decisions, 22(4), 105–120. doi:10.5277/ord120406

See Also

midpoint_heuristic, return_heuristic, sshape_heuristic

Examples

```
coordinates <- matrix(c(1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5,
                        0, 4, 11, 0, 10, 11, 0, 1, 5, 11, 0, 4, 11, 0, 4,
                        11), ncol = 2, byrow = FALSE,
                        dimnames = list(NULL, c("x", "y")))
dd_heuristic(coordinates)
```

---

generate_sample_goods_and_locatons_scenario

Generate Sample Goods and Locations Scenario for a Warehouse

Description

This function generates a sample scenario of goods and their locations in a warehouse, along with related data such as storage time, demand ratio, and full packages demand ratio.

Usage

```
generate_sample_goods_and_locatons_scenario(
  warehouse_height = 25,
  warehouse_width = 40,
  warehouse_size = warehouse_height * warehouse_height,
  nr_goods = 10,
  max_nr_locations = rep(10, nr_goods),
  max_storage_time = rep(30, nr_goods),
```

```
  max_demand = rep(200, nr_goods),
  actual_demand = rep(100, nr_goods),
  full_packaging_amount = rep(20, nr_goods)
)
```

Arguments

warehouse_height	An integer specifying the height of the warehouse (default is 25).
warehouse_width	An integer specifying the width of the warehouse (default is 40).
warehouse_size	The total size of the warehouse (height * width), default is calculated as 'warehouse_height * warehouse_height'.
nr_goods	An integer specifying the number of different goods/products in the warehouse (default is 10).
max_nr_locations	A vector specifying the maximum number of locations each product can occupy (default is 'rep(10, nr_goods)').
max_storage_time	A vector specifying the maximum storage time for each product (default is 'rep(30, nr_goods)').
max_demand	A vector specifying the maximum demand for each product (default is 'rep(200, nr_goods)').
actual_demand	A vector specifying the actual demand for each product (default is 'rep(100, nr_goods)').
full_packaging_amount	A vector specifying the full packaging amount for each product (default is 'rep(20, nr_goods)').

Details

The function randomly assigns goods to locations in the warehouse and calculates various parameters such as storage time, demand ratio, and the full packages demand ratio for each product. It also calculates the number of neighboring products at each location and the distance from the location to the I/O point.

Value

A data frame containing the generated scenario data with columns for:

- 'product': Product identifier.
- 'location': Location of the product in the warehouse.
- 'storage_time': Storage time of the product.
- 'demand_ratio': The ratio of demand relative to the actual demand.
- 'full_packages_demand_ratio': The ratio of full packages relative to the demand.
- 'distance': The calculated distance to the I/O point.
- 'neighbors': The number of neighboring products for each location.

Author(s)

Krzysztof Dmytrów <krzysztof.dmytrow@usz.edu.pl> [aut] ORCID: 0000-0001-7657-6063

Andrzej Dudek <andrzej.dudek@ue.wroc.pl> [aut, cre] ORCID: 0000-0002-4943-8703

References

Dmytrów, K. (2022). Analytical and simulation determination of order picking time in a low storage warehouse for shared storage systems. Operations Research and Decisions, 32(2), 34–51. doi:10.37190/ord220203

Examples

```
scenario <- generate_sample_goods_and_locatons_scenario()
scenario2 <- generate_sample_goods_and_locatons_scenario(warehouse_height = 10,
warehouse_width = 10,nr_goods = 5)
```

---

iopoint_distance *Calculate IO Point Distance*

Description

This function calculates the distance to the I/O point based on the provided position in the warehouse. The taxicab geometry is used to calculate the distance.

Usage

```
iopoint_distance(i, warehouse_height)
```

Arguments

i	Integer index representing the position.
warehouse_height	Integer representing the height of the warehouse (number of levels or rows).

Details

The function calculates the x and y coordinates of the position within the warehouse and returns the sum of x and y as the distance.

Value

A numeric value representing the calculated distance to the I/O point.

Author(s)

Krzysztof Dmytrów <krzysztof.dmytrow@usz.edu.pl> [aut] ORCID: 0000-0001-7657-6063

Andrzej Dudek <andrzej.dudek@ue.wroc.pl> [aut, cre] ORCID: 0000-0002-4943-8703

References

Dmytrów, K. (2022). Analytical and simulation determination of order picking time in a low storage warehouse for shared storage systems. Operations Research and Decisions, 32(2), 34–51. doi:10.37190/ord220203

Petrović, M., Malešević, B., Banjac, B., & Obradović, R. (2014). Geometry of some taxicab curves. *4th International Scientific Conference on Geometry and Graphics*. Serbian Society for Geometry and Graphics, University of Niš, Srbija.

Examples

```
iopoint_distance(i = 1,  warehouse_height = 10)
iopoint_distance(i = 15,  warehouse_height = 5)
```

`midpoint_heuristic` *Midpoint Heuristic for the Picker's Route Designation*

Description

This heuristic generates a return route solution for the picker's route designation in a warehouse by removing arcs at the top row of the warehouse and duplicating the remaining arcs to form a round trip.

Usage

```
midpoint_heuristic(arcs)
```

Arguments

`arcs`	A data frame or matrix representing the arcs (edges) in the warehouse. The arcs should have columns 'x_from', 'x_to', 'y_from', and 'y_to', which represent the coordinates of the connections between aisles and shelves.

Details

The heuristic first determines the warehouse height and the reduced width by examining the arcs. For each aisle, the edges are split and removed if they cross beyond the midpoint of the warehouse's height. Finally, I/O points are added, and any empty final aisles are deleted.

Value

A matrix of arcs after applying the midpoint heuristic, with edges that cross the warehouse's middle line excluded and I/O points added.

Author(s)

Krzysztof Dmytrów <krzysztof.dmytrow@usz.edu.pl> [aut] ORCID: 0000-0001-7657-6063

Andrzej Dudek <andrzej.dudek@ue.wroc.pl> [aut, cre] ORCID: 0000-0002-4943-8703

References

Dmytrów, K. (2022). Analytical and simulation determination of order picking time in a low storage warehouse for shared storage systems. Operations Research and Decisions, 32(2), 34–51. doi:10.37190/ord220203

Le-Duc, T. (2005). Design and Control of Efficient Order. Erasmus Research Institute of Management (ERIM).

Tarczyński, G. (2012). Analysis of the Impact of Storage Parameters and the Size of Orders on the Choice of the Method for Routing Order Picking. Operations Research and Decisions, 22(4), 105–120. doi:10.5277/ord120406

See Also

return_heuristic, sshape_heuristic

Examples

```
coordinates <- matrix(c(1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5,
                        0, 4, 11, 0, 10, 11, 0, 1, 5, 11, 0, 4, 11, 0, 4,
                        11), ncol = 2, byrow = FALSE,
                        dimnames = list(NULL, c("x", "y")))
midpoint_heuristic(create_arcs(coordinates))
```

---

nr_neighbors *Calculate the Number of Neighbors for a Warehouse Location*

Description

This function calculates the number of neighboring products for a specific product at a given location in the warehouse.

Usage

```
nr_neighbors(product, location, goods_and_locations, warehouse_height)
```

Arguments

product	The product for which neighbors are being calculated.
location	The specific location of the product in the warehouse.
goods_and_locations	A data frame containing the products and their corresponding locations in the warehouse. It should have at least two columns: ‘product‘ and ‘location‘.

`warehouse_height`
: warehouse height

Details

The function calculates how many products share a similar location pattern in the warehouse by grouping locations and checking whether they fall within the same row or column as the provided location. The function returns the total number of products in the same general area, excluding the current product.

Value

An integer representing the number of neighboring products at the specified location.

Author(s)

Krzysztof Dmytrów <krzysztof.dmytrow@usz.edu.pl> [aut] ORCID: 0000-0001-7657-6063

Andrzej Dudek <andrzej.dudek@ue.wroc.pl> [aut, cre] ORCID: 0000-0002-4943-8703

References

Dmytrów, K. (2022). Analytical and simulation determination of order picking time in a low storage warehouse for shared storage systems. Operations Research and Decisions, 32(2), 34–51. doi:10.37190/ord220203

Examples

```
scenario <- generate_sample_goods_and_locatons_scenario()
nr <- nr_neighbors(1,1,scenario,25)
```

---

rectangular_warehouse_from_copras_locations

Create a Rectangular Warehouse Layout from COPRAS Locations

Description

This function generates a rectangular warehouse layout matrix based on COPRAS-assigned locations, given the warehouse dimensions.

Usage

```
rectangular_warehouse_from_copras_locations(
  locations,
  warehouse_width,
  warehouse_height
)
```

Arguments

`locations`	A numeric vector representing the COPRAS-assigned locations of items in the warehouse.
`warehouse_width`	An integer specifying the width of the warehouse.
`warehouse_height`	An integer specifying the height of the warehouse.

Details

The function maps the given COPRAS-assigned item locations to a rectangular warehouse layout. The x and y coordinates are calculated based on the warehouse dimensions and adjusted for the warehouse structure. Additional rows are added for boundary positions at the top and bottom of the warehouse.

The resulting matrix contains the following columns: - 'x': The x-coordinate (column) of the warehouse. - 'y': The y-coordinate (row) of the warehouse.

The matrix is sorted by x and y coordinates and ensures unique rows in the layout.

Value

A matrix representing the rectangular warehouse layout. Each row corresponds to a coordinate in the warehouse.

Author(s)

Andrzej Dudek <andrzej.dudek@ue.wroc.pl> [aut, cre] ORCID: 0000-0002-4943-8703

Krzysztof Dmytrów <krzysztof.dmytrow@usz.edu.pl> [aut] ORCID: 0000-0001-7657-6063

scenario2 <- generate_sample_goods_and_locatons_scenario(warehouse_height = 10,warehouse_width = 10,nr_goods = 5) copras <- copras_assignment(scenario2) locations<-unique(unlist(lapply(copras, function(x){return(x$copras$location)}))) coordinates <- rectangular_warehouse_from_copras_locations(locations,warehou = 10,warehouse_width = 10)

References

Gudehus, T., & Kotzab, H. (2012). *Comprehensive Logistics*. Springer Berlin Heidelberg. doi:10.1007/9783642243677

Zavadskas, E. K., Kaklauskas, A., & Šarka, V. (1994). The new method of multicriteria complex proportional assessment projects. In E. K. Zavadskas & P. Linnert (Eds.), *Technological and economic development of economy. Volume 3. Business Management* (pp. 131–140). Vilnius: „Technika".

Examples

```
# Example usage
locations <- c(1, 5, 10, 15)
warehouse_width <- 8
warehouse_height <- 4
rectangular_warehouse_from_copras_locations(locations, warehouse_width, warehouse_height)
```

`return_heuristic` *Return Heuristic for the Picker's Route Designation*

Description

This heuristic generates a return route solution for the Traveling Salesman Problem (TSP) in a warehouse by removing arcs at the top row of the warehouse and duplicating the remaining arcs to form a round trip.

Usage

```
return_heuristic(arcs)
```

Arguments

`arcs`	A data frame or matrix representing the arcs (edges) in the warehouse.

Details

The heuristic first removes the arcs corresponding to the top row of the warehouse. Then, the remaining arcs are duplicated to simulate a return trip, and I/O points are added.

Value

A matrix of arcs after applying the return heuristic, where edges at the top row are removed and remaining arcs are duplicated to form a return route. I/O points are added at the end.

Author(s)

Krzysztof Dmytrów <krzysztof.dmytrow@usz.edu.pl> [aut] ORCID: 0000-0001-7657-6063

Andrzej Dudek <andrzej.dudek@ue.wroc.pl> [aut, cre] ORCID: 0000-0002-4943-8703

References

Dmytrów, K. (2022). Analytical and simulation determination of order picking time in a low storage warehouse for shared storage systems. Operations Research and Decisions, 32(2), 34–51. doi:10.37190/ord220203

Le-Duc, T. (2005). Design and Control of Efficient Order. Erasmus Research Institute of Management (ERIM).

Tarczyński, G. (2012). Analysis of the Impact of Storage Parameters and the Size of Orders on the Choice of the Method for Routing Order Picking. Operations Research and Decisions, 22(4), 105–120. doi:10.5277/ord120406

See Also

`midpoint_heuristic`, `sshape_heuristic`

Examples

```
coordinates <- matrix(c(1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5,
                        0, 4, 11, 0, 10, 11, 0, 1, 5, 11, 0, 4, 11, 0, 4,
                        11), ncol = 2, byrow = FALSE,
                        dimnames = list(NULL, c("x", "y")))
return_heuristic(create_arcs(coordinates))
```

---

sshape_heuristic *S-Shape Heuristic for the Picker's Route Designation*

Description

This heuristic generates an S-shape traversal solution for the picker's route designation in a warehouse by alternating traversal directions across aisles and ensuring no crossovers between top and bottom rows.

Usage

```
sshape_heuristic(arcs)
```

Arguments

arcs	A data frame or matrix representing the arcs (edges) in the warehouse.

Details

The S-shape heuristic removes unnecessary arcs and ensures a consistent traversal direction across the aisles. If an odd number of aisles is found, the last aisle is treated separately, and I/O points are added.

Value

A matrix of arcs after applying the S-shape heuristic, where traversal alternates between aisles, and I/O points are added.

Author(s)

Krzysztof Dmytrów <krzysztof.dmytrow@usz.edu.pl> [aut] ORCID: 0000-0001-7657-6063

Andrzej Dudek <andrzej.dudek@ue.wroc.pl> [aut, cre] ORCID: 0000-0002-4943-8703

References

Dmytrów, K. (2022). Analytical and simulation determination of order picking time in a low storage warehouse for shared storage systems. Operations Research and Decisions, 32(2), 34–51. doi:10.37190/ord220203

Le-Duc, T. (2005). Design and Control of Efficient Order. Erasmus Research Institute of Management (ERIM).

Tarczyński, G. (2012). Analysis of the Impact of Storage Parameters and the Size of Orders on the Choice of the Method for Routing Order Picking. Operations Research and Decisions, 22(4), 105–120. doi:10.5277/ord120406

See Also

midpoint_heuristic, return_heuristic

Examples

```
coordinates <- matrix(c(1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5,
                        0, 4, 11, 0, 10, 11, 0, 1, 5, 11, 0, 4, 11, 0, 4,
                        11), ncol = 2, byrow = FALSE,
                        dimnames = list(NULL, c("x", "y")))
sshape_heuristic(create_arcs(coordinates))
```

Index