Package ‘momst’

June 22, 2026

Type Package

Title Multi-Objective Minimum Spanning Tree via NSGA-II with Local Search

Version 0.1.1

Description Solves the Multi-Criteria Minimum Spanning Tree (mc-MST) problem on complete weighted graphs by combining the Non-dominated Sorting Genetic Algorithm II (NSGA-II) with optional Pareto local search operators. Chromosomes are represented as Prufer sequences so that every random individual decodes to a valid spanning tree (Cayley's theorem), avoiding repair operators. Four solver variants are provided: pure NSGA-II (``base"), Path Relinking (``PR"), Pareto Local Search (``PLS"), and Tabu Search (``TS"). The package supports 2 and 3 objective formulations and provides convenience functions to plot Pareto fronts and best-compromise spanning trees. This package is the reference implementation of the method described in Parraga-Alava, Inostroza-Ponta and Dorn (2017) <doi:10.1109/CEC.2017.7969432>.

License GPL (>= 3)

URL `https://github.com/jorgeklz/momst`, `https://jorgeklz.github.io/momst/`

BugReports `https://github.com/jorgeklz/momst/issues`

Encoding UTF-8

Depends R (>= 4.0)

Imports stats, utils, graphics

Suggests igraph, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Jorge A. Parraga-Alava [aut, cre] (ORCID: <`https://orcid.org/0000-0001-8558-9122`>)

Maintainer Jorge A. Parraga-Alava <`jorge.parraga@utm.edu.ec`>

Repository CRAN

Date/Publication 2026-06-22 14:40:15 UTC

Contents

`momst-package` *momst: Multi-Objective Minimum Spanning Tree via NSGA-II with Local Search*

Description

The **momst** package implements the Multi-Criteria Minimum Spanning Tree (mc-MST) algorithm using the NSGA-II (Non-dominated Sorting Genetic Algorithm II). Four variants are supported depending on the local search operator applied after each generation:

`"base"` Pure NSGA-II without local search.

`"PR"` NSGA-II plus Path Relinking.

`"PLS"` NSGA-II plus Pareto Local Search.

`"TS"` NSGA-II plus Tabu Search.

Details

Chromosomes are encoded as Prufer sequences, taking advantage of Cayley's theorem (every sequence of length n-2 with values in {1,...,n} decodes to a unique spanning tree of n nodes). This bijection makes every random chromosome a valid solution, avoiding repair operators.

The main entry point is `run_momst`.

Reference

This package is the reference implementation of the method described in:

Parraga-Alava, J., Inostroza-Ponta, M., & Dorn, M. (2017). Using local search strategies to improve the performance of NSGA-II for the Multi-Criteria Minimum Spanning Tree problem. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1818-1825). IEEE. doi:10.1109/CEC.2017.7969432

Author(s)

Jorge A. Parraga-Alava <`jorge.parraga@utm.edu.ec`>

See Also

Useful links:

- `https://github.com/jorgeklz/momst`
- `https://jorgeklz.github.io/momst/`
- Report bugs at `https://github.com/jorgeklz/momst/issues`

`apply_local_search` *Apply the Configured Local-Search Variant*

Description

Apply the Configured Local-Search Variant

Usage

```
apply_local_search(
  instance,
  pareto_pop,
  num_obj,
  n,
  variant,
  pop_size,
  lookup = NULL,
  verbose = FALSE
)
```

Arguments

`instance`	Edge-list `data.frame`.
`pareto_pop`	Integer matrix.
`num_obj`	Integer.
`n`	Integer.

`variant`	One of `"base"`, `"PR"`, `"PLS"`, `"TS"`.
`pop_size`	Integer.
`lookup`	Optional lookup.
`verbose`	Logical.

Value

data.frame.

`build_weight_lookup` *Pre-build Edge-Weight Lookup Matrices*

Description

Constructs one `n x n` matrix per objective so that the weight of any edge `(i, j)` can be queried in O(1) via standard matrix indexing.

Usage

```
build_weight_lookup(instance, n, num_obj)
```

Arguments

`instance`	`data.frame` returned by `generate_instance` or one with columns `from`, `to`, `weight_1`, `weight_2` (and optionally `weight_3`).
`n`	Integer. Number of nodes.
`num_obj`	Integer. Number of objectives (2 or 3).

Value

A list of length `num_obj`; element `k` is the symmetric `n x n` weight matrix of objective `k`.

Examples

```
inst <- generate_instance(10, 2, seed = 1)
L <- build_weight_lookup(inst, 10, 2)
L[[1]][1, 2]
```

`compute_objectives` *Compute Multi-Objective Costs for a Population*

Description

Compute Multi-Objective Costs for a Population

Usage

```
compute_objectives(instance, chromosomes, num_obj, lookup = NULL)
```

Arguments

`instance`	Edge-list `data.frame` (only used when `lookup` = `NULL`).
`chromosomes`	Numeric matrix `[pop_size` x `(n - 2)]`.
`num_obj`	Integer.
`lookup`	Optional list returned by `build_weight_lookup`.

Value

Numeric matrix `[pop_size` x `(n - 2 + num_obj)]`.

Examples

```
inst <- generate_instance(10, 2, seed = 1)
lk   <- build_weight_lookup(inst, 10, 2)
pop  <- generate_prufer_population(10, 5)
compute_objectives(inst, pop, 2, lk)
```

`decode_prufer` *Decode a Prufer Sequence to its Spanning Tree (Linear Time)*

Description

Returns the edge list of the spanning tree encoded by a Prufer sequence of length `n - 2`. Uses Wang's pointer-based linear-time algorithm, replacing the classic O(n^2) "search smallest leaf each step" approach.

Usage

```
decode_prufer(seq_prufer, n)
```

Arguments

`seq_prufer`	Integer vector of length `n - 2` with values in {1, ..., `n`}.
`n`	Integer. Number of nodes of the underlying graph.

Value

Integer matrix of dimensions `(n - 1) x 2`; each row is an undirected edge `(from, to)`.

Examples

```
decode_prufer(c(3, 1, 5), n = 5)
```

`generate_instance` *Generate a Complete-Graph Instance for MO-MST*

Description

Produces a complete undirected weighted graph with random multi-objective edge weights. Each edge gets two or three independent uniform weights, one per objective.

Usage

```
generate_instance(
  n,
  num_obj,
  range_a = c(10, 100),
  range_b = c(10, 50),
  range_c = c(30, 200),
  seed = NULL
)
```

Arguments

`n`	Integer. Number of nodes of the graph (must be at least 3).
`num_obj`	Integer in {2, 3}. Number of objectives.
`range_a`	Numeric vector `c(min, max)` for weights of objective 1.
`range_b`	Numeric vector `c(min, max)` for weights of objective 2.
`range_c`	Numeric vector `c(min, max)` for weights of objective 3. Ignored when `num_obj == 2`.
`seed`	Optional integer. If supplied, the RNG seed is fixed before sampling and restored afterwards, leaving the global RNG untouched.

Value

A `data.frame` with `n*(n-1)/2` rows and columns `from`, `to`, `weight_1`, `weight_2` (and `weight_3` when `num_obj == 3`).

Examples

```
inst <- generate_instance(n = 10, num_obj = 2, seed = 12345)
head(inst)
```

`generate_prufer_population`

Generate an Initial Prufer-Encoded Population

Description

Generate an Initial Prufer-Encoded Population

Usage

```
generate_prufer_population(n, pop_size)
```

Arguments

`n`	Integer. Number of nodes.
`pop_size`	Integer. Number of individuals to generate.

Value

Integer matrix of dimensions `pop_size x (n - 2)`.

Examples

```
generate_prufer_population(n = 6, pop_size = 4)
```

`non_dominated_crowding`

Assign Pareto Rank and Crowding Distance

Description

Assign Pareto Rank and Crowding Distance

Usage

```
non_dominated_crowding(population, num_obj)
```

Arguments

`population`	Numeric matrix `[N x (vars + num_obj)]`.
`num_obj`	Integer.

Value

Matrix with extra columns `rankingIndex` and `density`.

`pareto_local_search` *Pareto Local Search*

Description

Pareto Local Search

Usage

```
pareto_local_search(
  instance,
  pareto_pop,
  num_obj,
  n,
  neighbour_frac = 0.1,
  pop_size,
  lookup = NULL,
  verbose = FALSE
)
```

Arguments

`instance`	Edge-list `data.frame`.
`pareto_pop`	Integer matrix `[k x (n - 2)]`.
`num_obj`	Integer.
`n`	Integer.
`neighbour_frac`	Numeric.
`pop_size`	Integer.
`lookup`	Optional lookup.
`verbose`	Logical.

Value

data.frame of chromosomes.

`path_relinking` *Path Relinking on the Current Pareto Front*

Description

Path Relinking on the Current Pareto Front

Usage

```
path_relinking(
  instance,
  pareto_pop,
  num_obj,
  n,
  pop_size,
  lookup = NULL,
  verbose = FALSE
)
```

Arguments

`instance`	Edge-list `data.frame`.
`pareto_pop`	Integer matrix `[k x (n - 2)]`.
`num_obj`	Integer.
`n`	Integer.
`pop_size`	Integer.
`lookup`	Optional lookup.
`verbose`	Logical.

Value

data.frame of non-dominated chromosomes.

`plot_best_tree` *Plot the Best-Compromise Spanning Tree*

Description

Plot the Best-Compromise Spanning Tree

Usage

```
plot_best_tree(result, n)
```

Arguments

`result`	List returned by `run_momst`.
`n`	Integer.

Value

Invisible NULL.

`plot_pareto_front` *Plot a Pareto Front (2-objective case)*

Description

Plot a Pareto Front (2-objective case)

Usage

```
plot_pareto_front(result, show_dominated = FALSE)
```

Arguments

`result`	List returned by `run_momst`.
`show_dominated`	Logical.

Value

Invisible NULL.

`random_mutation` *Random Mutation on Prufer Sequences*

Description

Random Mutation on Prufer Sequences

Usage

```
random_mutation(population, pop_size, mut_rate)
```

Arguments

`population`	Numeric matrix `[pop_size x (n - 2)]`.
`pop_size`	Integer.
`mut_rate`	Numeric in \[0, 1\].

Value

Integer matrix `[pop_size x (n - 2)]`.

`run_momst` *Run the MO-MST NSGA-II Solver*

Description

Single entry point that replaces the original `main.R` script.

Usage

```
run_momst(
  instance = NULL,
  instance_file = NULL,
  n = 10L,
  num_obj = 2L,
  variant = c("base", "PR", "PLS", "TS"),
  iterations = 10L,
  pop_size = 50L,
  tour_size = 2L,
  cross_rate = 0.8,
  mut_rate = 0.05,
  max_generations = 100L,
  convergence_window = 10L,
  range_a = c(10, 100),
  range_b = c(10, 50),
  range_c = c(30, 200),
  save_dir = NULL,
  verbose = TRUE,
  seed = NULL
)
```

Arguments

`instance`	Optional `data.frame`.
`instance_file`	Optional path.
`n`	Integer.
`num_obj`	Integer in {2, 3}.
`variant`	One of `"base"`, `"PR"`, `"PLS"`, `"TS"`.
`iterations`	Integer or two-length integer `c(min_iter, max_iter)`.
`pop_size`	Integer (must be even).
`tour_size`	Integer.
`cross_rate`	Numeric in \[0, 1\].
`mut_rate`	Numeric in \[0, 1\].
`max_generations`	Integer.

`convergence_window`	Integer.
`range_a`, `range_b`, `range_c`	Weight ranges for instance generation.
`save_dir`	Optional directory for per-iteration result files.
`verbose`	Logical.
`seed`	Optional integer.

Value

Invisible list with the solution data.

References

Parraga-Alava, J., Inostroza-Ponta, M., & Dorn, M. (2017). Using local search strategies to improve the performance of NSGA-II for the Multi-Criteria Minimum Spanning Tree problem. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1818-1825). IEEE. doi:10.1109/CEC.2017.7969432

Examples

```
res <- run_momst(n = 10, num_obj = 2, iterations = 3,
                 pop_size = 20, max_generations = 30,
                 variant = "base", seed = 1)
head(res$global_pareto)
```

`tabu_search` *Tabu Search on the Current Pareto Front*

Description

Tabu Search on the Current Pareto Front

Usage

```
tabu_search(
  instance,
  pareto_pop,
  num_obj,
  n,
  neighbour_frac = 0.05,
  pop_size,
  lookup = NULL,
  verbose = FALSE
)
```

Arguments

`instance`	Edge-list `data.frame`.
`pareto_pop`	Integer matrix `[k x (n - 2)]`.
`num_obj`	Integer.
`n`	Integer.
`neighbour_frac`	Numeric.
`pop_size`	Integer.
`lookup`	Optional lookup.
`verbose`	Logical.

Value

data.frame of non-dominated chromosomes.

`tournament_selection` *Tournament Selection*

Description

Tournament Selection

Usage

```
tournament_selection(population, pop_size, tour_size)
```

Arguments

`population`	Matrix with last two columns being `rankingIndex` and `density`.
`pop_size`	Integer.
`tour_size`	Integer.

Value

Selected subpopulation matrix.

`uniform_crossover` *Uniform Crossover for Prufer Sequences*

Description

Uniform Crossover for Prufer Sequences

Usage

```
uniform_crossover(pool, pop_size, cross_rate)
```

Arguments

`pool`	Numeric matrix `[pop_size x (n - 2)]`.
`pop_size`	Integer (must be even).
`cross_rate`	Numeric in \[0, 1\].

Value

Integer matrix `[pop_size x (n - 2)]`.

Index