Package: heumilkr (via r-universe)

August 29, 2024

Title Heuristic Capacitated Vehicle Routing Problem Solver

Version 0.2.0.9000

Description Implements the Clarke-Wright algorithm to find a quasi-optimal solution to the Capacitated Vehicle Routing Problem. See Clarke, G. and Wright, J.R. (1964) <doi:10.1287/opre.12.4.568> for details. The implementation is accompanied by helper functions to inspect its solution.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

LinkingTo cpp11

SystemRequirements C++17

URL `https://github.com/lschneiderbauer/heumilkr`, `https://lschneiderbauer.github.io/heumilkr/`

BugReports `https://github.com/lschneiderbauer/heumilkr/issues`

Imports rlang (>= 1.1.0), cli (>= 3.6.0), xml2 (>= 1.3.0), ggplot2 (>= 3.4.0)

Suggests testthat (>= 3.0.0), hedgehog (>= 0.1), curl (>= 5.2.0), ggExtra (>= 0.10.0), scales (>= 1.3.0), knitr, rmarkdown

Config/testthat/edition 3

Depends R (>= 3.5.0)

LazyData true

VignetteBuilder knitr

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

RemoteUrl https://github.com/lschneiderbauer/heumilkr

RemoteRef HEAD

RemoteSha fd1f5e571ba4a4645baff5af94ce2f5e91722be2

Contents

`clarke_wright` *Clarke-Wright algorithm, a Capacitated Vehicle Routing Problem solver*

Description

Finds a quasi-optimal solution to the Capacitated Vehicle Routing Problem (CVRP). It is assumed that all demands will be satisfied by a single source.

Usage

```
clarke_wright(demand, distances, vehicles, restrictions = NULL)
```

Arguments

`demand` A numeric vector consisting of "demands" indexed by sites. The `i`th entry refers to the demand of site `i` (and the length of the vector equals the number of sites `N` with demands). The units of demand values need to match the units of vehicle capacity values. `NA` values are not allowed.

`distances` An object of class `dist`, created by `stats::dist()`, with `(N + 1)` locations describing the distances between individual sites. The first index refers to the source site. The `(i+1)`th index refers to site `i` (as defined by `demand`).

`vehicles` A `data.frame()` describing available vehicle types and their respective capacities. One row per vehicle type. The data frame is expected to have two columns:

- `n` - Number of available vehicles. This can be set to `NA` if the number is "infinite" (i.e. effectively the maximal integer value on your machine.). It is recommended to keep at least one vehicle type as "infinite", otherwise the solver might raise a run time error due to initially not having enough vehicles available (even though the final solution might satisfy the availability restrictions).
- `caps` - The vehicle capacity in same units as `demand`.

The order of the `data.frame()` is relevant and determines the prioritization of vehicle assignments to runs (in case two or more vehicle types are eligible for assignment the "first" vehicle is chosen). In a typical scenario "more expensive" vehicles should be further down in the list (so the cheaper one is chosen in case there is doubt). Since higher capacity vehicles usually involve higher costs sorting the data frame by capacity is usually a good rule of thumb.

`restrictions` An optional `data.frame()` that allows to define vehicle type restrictions for particular sites in the form of a blacklist. The data frame is expected to have two columns:

- `vehicle` - The vehicle type index.
- `site` - The site index (i.e. the index of the `demand` vector)

Each row defines a restriction: vehicle type `vehicle` can not approach site `site`. Defaults to `NULL`, i.e. no restrictions are enforced.

Details

See the original paper, Clarke, G. and Wright, J.R. (1964) doi:10.1287/opre.12.4.568, for a detailed explanation of the Clarke-Wright algorithm.

Value

Returns a `"heumilkr_solution"` object, a `data.frame()` with one row per site-run combination bestowed with additional attributes. Its columns consist of:

- `site` - The site index (i.e. the index of the `demand` vector) associated to the run.
- `run` - Identifies the run the site is assigned to.
- `order` - Integer values providing the visiting order within each run.
- `vehicle` - The vehicle type index (as provided in `vehicles`) associated to the run.
- `load` - The actual load in units of `demand` on the particular run.
- `distance` - The travel distance of the particular run.

Unless a site demand exceeds the vehicle capacities it is always assigned to only a single run.

Examples

```
demand <- c(3, 2, 4, 2)

positions <-
  data.frame(
    pos_x = c(0, 1, -1, 2, 3),
    pos_y = c(0, 1, 1, 2, 3)
  )

clarke_wright(
  demand,
  dist(positions),
  data.frame(n = NA_integer_, caps = 6)
)
```

`clarke_wright_cvrplib` *Applying `clarke_wright()` to CVRPLIB data*

Description

Applying `clarke_wright()` to CVRPLIB data

Usage

```
clarke_wright_cvrplib(instance)
```

Arguments

`instance`	A "`heumilkr_solution`". See `cvrplib_download()` or bundled CVRPLIB data like cvrplib_A.

Value

A "`heumilkr_solution`" object. See `clarke_wright()`.

See Also

Other cvrplib: `cvrplib_download()`, `cvrplib_ls()`

`cvrplib_A` *CVRP instance data by Augerat, 1995*

Description

A collection of CVRP instances by Augerat, 1995, provided courtesy of CVRPLIB. See CVRPLIB for visualizations of the instances and their solutions as well as a multitude of alternative instance data.

Usage

```
cvrplib_A
```

Format

`cvrplib_A`:
A list of CVRP instances as "`cvrplib_instance`" objects. The instances can be directly fed into solver algorithm, e.g. via `clarke_wright_cvrplib()`.

Source

`http://vrp.atd-lab.inf.puc-rio.br`

`cvrplib_B` *CVRP instance data by Augerat, 1995*

Description

A collection of CVRP instances by Augerat, 1995, provided courtesy of CVRPLIB. See CVRPLIB for visualizations of the instances and their solutions as well as a multitude of alternative instance data.

Usage

```
cvrplib_B
```

Format

`cvrplib_B`**:**

A list of CVRP instances as "`cvrplib_instance`" objects. The instances can be directly fed into solver algorithm, e.g. via `clarke_wright_cvrplib()`.

Source

`http://vrp.atd-lab.inf.puc-rio.br`

`cvrplib_download` *CVRPLIB problem instance downloader*

Description

CVRLIB offers a selection of CVRP problem instances. This function downloads the instance data and conveniently makes it available to be fed into solver functions, e.g. with `clarke_wright_cvrplib()`. The primary purpose for those instances is benchmarking / comparing speed as well as performance of solvers.

Usage

```
cvrplib_download(qualifier)
```

Arguments

`qualifier`	The qualifier of the problem instance. E.g. "tai/tai150d". This can either be inferred directly from the website or by the output of `cvrplib_ls()`.

Value

Returns a "`cvrplib_instance`" object which contains CVRPLIB problem instance data.

See Also

Other cvrplib: `clarke_wright_cvrplib()`, `cvrplib_ls()`

`cvrplib_E` *CVRP instance data by Christofides and Eilon, 1969*

Description

A collection of CVRP instances by Christofides and Eilon, 1969, provided courtesy of CVRPLIB. See CVRPLIB for visualizations of the instances and their solutions as well as a multitude of alternative instance data.

Usage

```
cvrplib_E
```

Format

`cvrplib_E`:
A list of CVRP instances as "`cvrplib_instance`" objects. The instances can be directly fed into solver algorithm, e.g. via `clarke_wright_cvrplib()`.

Source

`http://vrp.atd-lab.inf.puc-rio.br`

`cvrplib_F` *CVRP instance data by Fisher, 1994*

Description

A collection of CVRP instances by Fisher, 1994, provided courtesy of CVRPLIB. See CVRPLIB for visualizations of the instances and their solutions as well as a multitude of alternative instance data.

Usage

```
cvrplib_F
```

Format

`cvrplib_F`:
A list of CVRP instances as "`cvrplib_instance`" objects. The instances can be directly fed into solver algorithm, e.g. via `clarke_wright_cvrplib()`.

Source

`http://vrp.atd-lab.inf.puc-rio.br`

`cvrplib_ls` *List available CVRPLIB online data*

Description

Scrapes the CVRPLIB website to look for available data sets. This function call can take some time.

Usage

```
cvrplib_ls()
```

Value

A vector of data set qualifiers which can be used with `cvrplib_download()`.

See Also

Other cvrplib: `clarke_wright_cvrplib()`, `cvrplib_download()`

`cvrplib_Tai` *CVRP instance data by Rochat and Taillard, 1995*

Description

A collection of CVRP instances by Rochat and Taillard, 1995, provided courtesy of CVRPLIB. See CVRPLIB for visualizations of the instances and their solutions as well as a multitude of alternative instance data.

Usage

```
cvrplib_Tai
```

Format

`cvrplib_Tai`**:**

A list of CVRP instances as "`cvrplib_instance`" objects. The instances can be directly fed into solver algorithm, e.g. via `clarke_wright_cvrplib()`.

Source

`http://vrp.atd-lab.inf.puc-rio.br`

milkr_cost *Vehicle runs cost / distance*

Description

Calculates the total distance associated to a `clarke_wright()` result. This is the measure that the corresponding Capacitated Vehicle Routing Problem minimizes.

Usage

```
milkr_cost(solution)
```

Arguments

solution A "heumilkr_solution" object, typically obtained by `clarke_wright()`.

Value

The total traveled distance.

Examples

```
demand <- c(3, 2, 4, 2)

positions <-
  data.frame(
    pos_x = c(0, 1, -1, 2, 3),
    pos_y = c(0, 1, 1, 2, 3)
  )

solution <- clarke_wright(
  demand,
  dist(positions),
  data.frame(n = NA_integer_, caps = 6)
)

milkr_cost(solution)
```

milkr_saving *Vehicle run saving*

Description

Measures the saving that was achieved by the heuristic optimization algorithm `clarke_wright()` compared to the naive vehicle run assignment, i.e. one run per site.

Usage

```
milkr_saving(solution, relative = FALSE)
```

Arguments

`solution`	A "`heumilkr_solution`" object, typically obtained by `clarke_wright()`.
`relative`	Should the saving be given as dimensionful value (in units of distance as provided to `clarke_wright()`), or as percentage relative to the naive costs. Defaults to `FALSE`, i.e. a dimensionful value.

Value

The savings either as dimensionful value or as percentage relative to the naive costs, depending on `relative`.

Examples

```
demand <- c(3, 2, 4, 2)

positions <-
  data.frame(
    pos_x = c(0, 1, -1, 2, 3),
    pos_y = c(0, 1, 1, 2, 3)
  )

solution <- clarke_wright(
  demand,
  dist(positions),
  data.frame(n = NA_integer_, caps = 6)
)

print(milkr_saving(solution))
print(milkr_saving(solution, relative = TRUE))
```

Index