VehicleRoutingSynPre

Payakorn Saksuriya

August 1, 2022

Import package

```julia
using VehicleRoutingSynPre
using PrettyTables
```

---

1 Benchmark

Benchmark name is in the form, for exmaple, "ins10-1.jld2" is the instance number 1 with 10 nodes (not include depot).

To load the instance data use

```julia
name = "ins10-1"
num_node, num_vehi, num_serv, mind, maxd, a, r, d, p, e, l = load_data(name);
```

where

- `num_node`: total number of nodes (including depot)
- `num_vehi`: total number of vehicles
- `num_serv`: total number of services
- `mind`: minumum different of starting time between two services of each node
- `maxd`: maximum different of starting time between two services of each node
- `a`: compatibility matrix a[i,j] = 1 if vehicle i can process service j
- `r`: requiment matrix r[i,j] = 1 if node i requires service j
- `d`: distance matrix
- `p`: processing time matrix, p[i, j, k] = processing time of service j of vehicle i on node k
- `e`: earilest start time of each node
- `l`: latest start time of each node

Generate random particle (solution)

```
particle = generate_particles(name);
```

The particle is `struct` of `Particle` with fields

```
dump(Particle)
```

```
Particle <: Any
  route::Vector{Vector{Vector{Int64}}}
  starttime::Dict{Int64, Array{Float64}}
  slot::Dict{Int64, Vector{Int64}}
  serv_a::Tuple
  serv_r::Dict{Int64, Vector{Int64}}
  num_node::Int64
  num_vehi::Int64
  num_serv::Int64
  mind::Vector{Float64}
  maxd::Vector{Float64}
  a::Array{Int64}
  r::Array{Int64}
  d::Matrix{Float64}
  p::Array{Float64}
  e::Vector{Int64}
  l::Vector{Int64}
  PRE::Vector{Tuple}
  SYN::Vector{Tuple}
```

```
particle.route, particle.slot = example();
particle.starttime = find_starttime(particle)
```

```
Dict{Int64, Matrix{Float64}} with 12 entries:
  5  => [0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0]
  7  => [0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0; 0.0 0.0 … 224.083 0.0]
  8  => [0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0]
  1  => [480.527 0.0 … 0.0 0.0; 73.0384 0.0 … 0.0 0.0; 480.16 0.0 … 0.0 0.0
]
  0  => [0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0]
  4  => [0.0 247.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0]
  6  => [0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0]
  2  => [0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0]
  10 => [356.043 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0]
  11 => [0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 159.161]
  9  => [0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 46.0; 0.0 0.0 … 46.0 0.0]
  3  => [0.0 0.0 … 0.0 0.0; 0.0 0.0 … 0.0 0.0; 0.0 0.0 … 291.121 0.0]
```

```
particle.route
```

```
3-element Vector{Vector{Vector{Int64}}}:
 [[11, 3], [4, 2], [6, 3], [10, 1], [8, 3], [1, 1]]
 [[9, 6], [1, 1]]
 [[9, 5], [11, 6], [7, 5], [3, 5], [2, 4], [10, 4], [5, 4], [1, 1]]
```

where route start and end from depot node `1`

For example,

in row 1 first element [11, 3] represents route of vehicle 1 from node 1 to node 11 and process service 3 in row 1 second element [4, 2] represents route of vehicle 1 from node 11 to node 4 and process service 2

---

slot is the service sequence of each node

```
particle.slot

Dict{Int64, Vector{Int64}} with 10 entries:
  5  => [4]
  4  => [2]
  6  => [3]
  7  => [5]
  2  => [4]
  10 => [1, 4]
  11 => [3, 6]
  9  => [5, 6]
  8  => [3]
  3  => [5]
```

The starting time at each node

```
for i in 1:11
    println("Start time of node $i")
    pretty_table(particle.starttime[i])
end

Start time of node 1

  Col. 1   Col. 2   Col. 3   Col. 4   Col. 5   Col. 6

 480.527      0.0      0.0      0.0      0.0      0.0
 73.0384      0.0      0.0      0.0      0.0      0.0
  480.16      0.0      0.0      0.0      0.0      0.0

Start time of node 2

 Col. 1   Col. 2   Col. 3   Col. 4   Col. 5   Col. 6

    0.0      0.0      0.0      0.0      0.0      0.0
    0.0      0.0      0.0      0.0      0.0      0.0
    0.0      0.0      0.0    345.0      0.0      0.0

Start time of node 3

 Col. 1   Col. 2   Col. 3   Col. 4    Col. 5   Col. 6

    0.0      0.0      0.0      0.0       0.0      0.0
    0.0      0.0      0.0      0.0       0.0      0.0
    0.0      0.0      0.0      0.0   291.121      0.0

Start time of node 4

 Col. 1   Col. 2   Col. 3   Col. 4   Col. 5   Col. 6

    0.0    247.0      0.0      0.0      0.0      0.0
    0.0      0.0      0.0      0.0      0.0      0.0
    0.0      0.0      0.0      0.0      0.0      0.0
```

```
Start time of node 5

 Col. 1   Col. 2   Col. 3   Col. 4   Col. 5   Col. 6

    0.0      0.0      0.0      0.0      0.0      0.0
    0.0      0.0      0.0      0.0      0.0      0.0
    0.0      0.0      0.0   458.88      0.0      0.0

Start time of node 6

 Col. 1   Col. 2    Col. 3   Col. 4   Col. 5   Col. 6

    0.0      0.0   314.151      0.0      0.0      0.0
    0.0      0.0       0.0      0.0      0.0      0.0
    0.0      0.0       0.0      0.0      0.0      0.0

Start time of node 7

 Col. 1   Col. 2   Col. 3   Col. 4    Col. 5   Col. 6

    0.0      0.0      0.0      0.0       0.0      0.0
    0.0      0.0      0.0      0.0       0.0      0.0
    0.0      0.0      0.0      0.0   224.083      0.0

Start time of node 8

 Col. 1   Col. 2   Col. 3   Col. 4   Col. 5   Col. 6

    0.0      0.0    434.0      0.0      0.0      0.0
    0.0      0.0      0.0      0.0      0.0      0.0
    0.0      0.0      0.0      0.0      0.0      0.0

Start time of node 9

 Col. 1   Col. 2   Col. 3   Col. 4   Col. 5   Col. 6

    0.0      0.0      0.0      0.0      0.0      0.0
    0.0      0.0      0.0      0.0      0.0     46.0
    0.0      0.0      0.0      0.0     46.0      0.0

Start time of node 10

  Col. 1   Col. 2   Col. 3    Col. 4   Col. 5   Col. 6

 356.043      0.0      0.0       0.0      0.0      0.0
     0.0      0.0      0.0       0.0      0.0      0.0
     0.0      0.0      0.0   416.454      0.0      0.0

Start time of node 11

 Col. 1   Col. 2   Col. 3   Col. 4   Col. 5    Col. 6

    0.0      0.0    148.0      0.0      0.0       0.0
    0.0      0.0      0.0      0.0      0.0       0.0
    0.0      0.0      0.0      0.0      0.0   159.161
```

Distance matrix

```
pretty_table(particle.d, tf=tf_html_matrix, show_row_number=true)
```