The TEGP generator

LARS RELUND NIELSEN*
Research Unit of Statistics and Decision Analysis, Research Centre Foulum, P.O. Box 50, DK-8830 Tjele, Denmark

Version 1.5 - Jan. 2006

Abstract

This manual provides documentation of the *Time-Expanded Generator with Peaks* (*TEGP*) which generates instances of stochastic time-dependent networks. The program includes several features inspired by typical aspects of road networks (congestion effects, waiting, random perturbations etc.).

Keywords: stochastic time-dependent networks, testing, generator.

Contents

*e-mail: lars@relund.dk

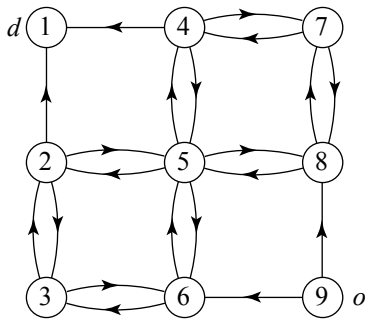


Figure 1: A topological grid network (3×3 grid).

1 Introduction

The *Time-Expanded Generator with Peaks* (*TEGP*) generates instances of *stochastic time-dependent networks* (*STD networks*). The program includes several features inspired by typical aspects of road networks (congestion effects, waiting, random perturbations etc.). Note that the TEGP generator like other generators only models a fraction of a real network. However, it provides alternative choices that may affect the behavior of the algorithms. For an short introduction to STD networks and the notation used see Appendix A.

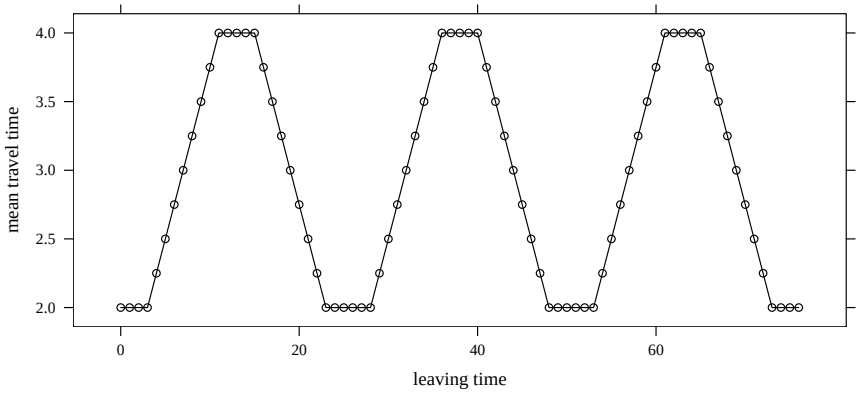


Figure 2: Mean travel time for an arc ($\psi = 100\%$).

A topological grid network G of *base* b and *height* h is assumed, and we search for optimal strategies from the bottom-right corner node (origin o) to the upper left corner node (destination d). This choice is motivated by the fact that each origin-destination path has at least $b + h - 2$ arcs, and the number of such paths grows exponential with the size of G. A topological grid network G of base 3 and height 3 is shown in Figure 1. Note that no arcs enter the origin and no arcs exit the destination.

The generator considers *cyclic* time periods (e.g. a day) and the time horizon H is the (finite) number of time instances in a cycle multiplied by the number of cycles. In each cyclic period there are some *peak periods* (e.g. rush hours). Each peak consists of three parts; a *transient* part where the traffic increases, a *pure peak* part where the traffic stays the same and a transient part where the traffic decreases again. Peaks are placed at the same time in each cycle. If no peaks are wanted then the length of the peak may be set to zero.

2 Generating travel time distributions

The travel time distributions are found by first generating the *off-peak* mean travel times $\mu(u, v) \in \{lb_T, ..., ub_T\}$ for all arcs (u, v).

Let $\mu_{uv}(t)$ denote the mean travel time for each possible leaving time $t \in L(u, v)$. The mean $\mu_{uv}(t)$ follows a pattern like the line in Figure 2 where the mean for each leaving time is shown with a circle. In off-peaks $\mu_{uv}(t) = \mu(u, v)$, i.e. the constant mean travel time in off-peaks. At the beginning of a peak, $\mu_{uv}(t)$ increases from $\mu(u, v)$ to $\mu(u, v)(1 + \psi)$, where ψ denotes the *peak increase parameter*, then stays the same during the pure peak period, and then decreases to $\mu(u, v)$ again.

Let $X(u, v, t)$ denote the travel time to node v when leaving node u at time t along arc (u, v). Given the mean travel time $\mu_{uv}(t)$ the travel time distribution is found as follows.

1. Find $\sigma_{uv}(t) = \rho\mu_{uv}(t)$ where ρ is the *standard deviation mean ratio*.
2. The set of possible travel times is then $\{t_1, ..., t_{\kappa(u,v,t)}\} = \{\lfloor \mu_{uv}(t) - \sigma_{uv}(t) \rfloor, ..., \mu_{uv}(t), ..., \lceil \mu_{uv}(t) + \sigma_{uv}(t) \rceil\}$. Moreover, only positive travel times in $\{t_1, ..., t_{\kappa(u,v,t)}\}$ are considered.
3. Note $X(u, v, t) = t_1 + Y(u, v, t)$ where Y is a discrete random variable taking the values $\{0, ..., \kappa(u, v, t) - 1\}$. Given the positive travel times $\{t_1, ..., t_{\kappa(u,v,t)}\}$ we now set $Pr(X = t_i) = Pr(Y = i - 1)$ where $Y \sim bi(q - 1, 1/2)$. As a result $E(X) = \mu_{uv}(t)$.

Note, using the setting above gives higher mean travel time and higher standard deviation in peaks. Moreover, only the interval of possible off-peak travel times $\{lb_T, ..., ub_T\}$ is given as an input parameter to the TEGP generator. The actual interval of possible travel times I_T depends on the peak increase parameter ψ and the standard deviation mean ratio ρ, i.e.

$$I_T = \{\lfloor (1-\rho)\, lb_T \rfloor, \ldots, \lceil (1+\psi)\,(1+\rho)\, ub_T \rceil\} \tag{1}$$

Finally, symmetric mean travel times can be used for the arcs in G. In this case $\mu_{uv}(t) = \mu_{vu}(t)$.

3 Generating costs

Three different types of costs are considered, namely, travel costs, waiting costs and penalty costs. Currently, negative costs are not accepted.

3.1 Generating travel costs

Two costs $c_i(u, v, t)$, $i = 1, 2$ for each arc (u, v) and leaving time $t \in H$ are generated, since we may need two costs if bicriterion route choice is considered. The way the costs are generated is controlled using two flags. The first flag, $flag_{cor}$, specify the correlation between the two off-peak costs $c_i(u, v)$, $i = 1, 2$. The second flag, $flag_C$, specify how the costs, given an arc, for different leaving times depend on each other.

The following values of $flag_{cor}$ are possible:

0 - both costs $c_i(u, v)$ are random in $\{lb_C, ..., ub_C\}$.

1 - $c_2(u, v) = ub_C - (c_1(u, v) - lb_C)$

2 - The costs are generated as follows:

$$c_1(u,v) < \frac{ub_C - lb_C}{2} \Rightarrow c_2(u,v) \in \{ub_C - (c_1(u,v) - lb_C), \ldots, ub_C\}$$
$$c_1(u,v) \geq \frac{ub_C - lb_C}{2} \Rightarrow c_2(u,v) \in \{lb_C, \ldots, lb_C + (ub_C - c_1(u,v))\}$$

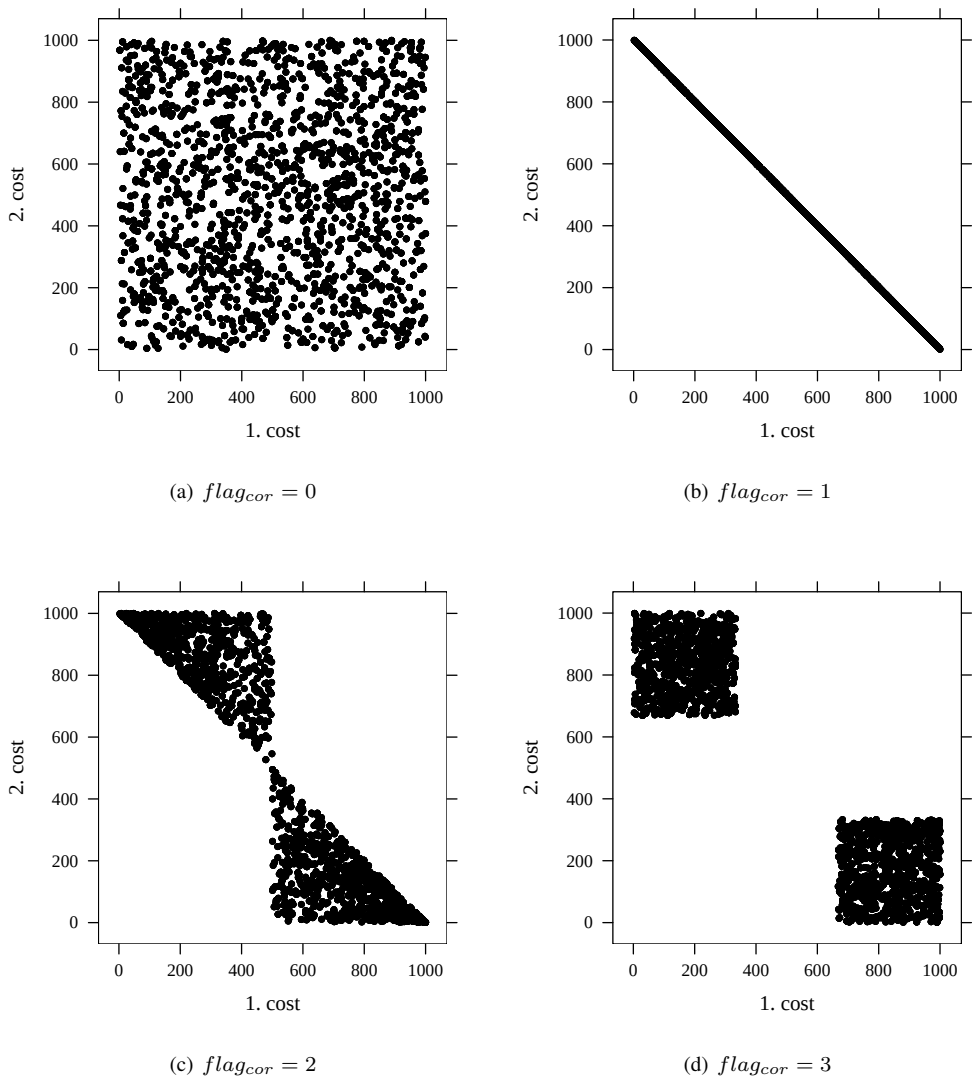
3 - NETMAKER costs see Skriver and Andersen [8]. "...if one cost is between 1 and 33, the other is between 67 and 100"

Note for $flag_{cor}$ equal 1, 2 and 3 the two costs are negatively correlated. This is a typical situation in hazardous material transportation, where travel cost and risk/exposure are conflicting.

Off-peak costs generated in the interval $[1, 1000] \times [1, 1000]$ for the four correlation types are shown in Figure 3.

Given costs $c_i(u, v)$ the generation of costs $c_i(u, v, t)$, $i = 1, 2$ may take three components into account: the off-peak cost, the peak effect, and a random perturbation. The *random perturbation* introduces small variations in the cost, due to other factors not intercepted by the peak, e.g. special information about the cost at exactly that leaving time. Given a cost c, we generate a perturbation $\xi \in [-r_\xi, r_\xi]$, where range r_ξ is a small percentage. Then, the cost after applying the random perturbation becomes $c(1+\xi)$. The following values of $flag_C$ are possible:

0 - Costs are generated independently for each leaving time and correlated as specified by $flag_{cor}$. That is, we have costs in the interval $\{lb_C, ..., ub_C\}$. The random perturbation is not applied. A plot of the costs for a specific arc is shown in Figure 4. In the left figure costs $c_i(u, v, t)$, $i = 1, 2$ are displayed



(a) $flag_{cor} = 0$

(b) $flag_{cor} = 1$

(c) $flag_{cor} = 2$

(d) $flag_{cor} = 3$

Figure 3: Off-peak costs for the different correlation types.

for each leaving time (1. cost is the solid line, 2. cost is the dotted line) and in the right figure the costs (c_1, c_2) are shown. Note that by using this setting, e.g. the costs of leaving u at time t may be 10 and the costs of leaving u at time $t+1$ may be 500. In a road network this is probably not realistic and peak dependent cost can be generated instead. However, random costs can be used to see if e.g. a priori algorithms are robust, see Nielsen, Pretolani, and Andersen [6].

1 - Consider node u having a west, north, east and south arc.[1] First the off-peak costs $c_i(u, v) \in \{lb_C, ..., ub_C\}$, $i = 1, 2$, are generated for the west and north arcs with a correlation as specified by $flag_{cor}$. Let $c_i^{\min}$ denote the minimum cost generated for criterion i of the north and west arcs. For the east and south arc off-peak costs $c_i(u, v) \in \{lb_C, ..., c_i^{\min}\}$, $i = 1, 2$, are now generated. In this way, east and south arcs are expected to have smaller costs than the corresponding west or north arcs. Finally, the costs $c_i\,(u, v, t)$, $i = 1, 2$ are found by applying the random perturbation. Note can only be used with non symmetric arcs. Costs of a west arc $(5, 2)$ and an east arc $(5, 8)$ (of the grid in

[1] Some arcs are not considered in output if the arc do not exist in the underlying grid.

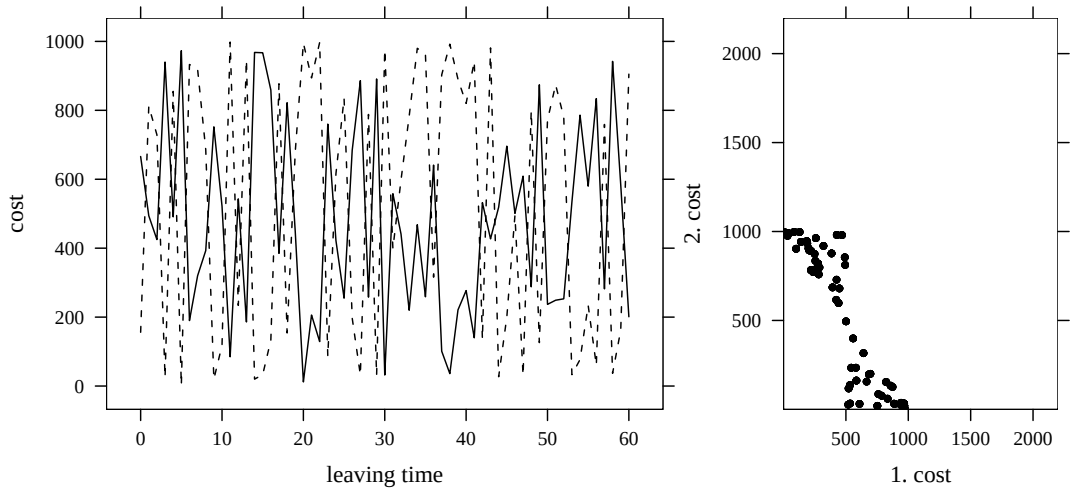


Figure 4: Travel costs for $flag_C = 0$ and $flag_{cor} = 2$.

Figure 1) are given in Figure 5. Note the only difference in e.g. $c_1(5, 2, t)$, $t = 1, ..., H$ is due to the random perturbation (equal 10%).

2 - First the off-peak costs $c_i(u, v) \in \{lb_C, ..., ub_C\}$, $i = 1, 2$, are generated with a correlation as specified by $flag_{cor}$. Next, the costs $c_i\,(u, v, t)$, $i = 1, 2$ are found by applying the random perturbation. A plot of the costs for a specific arc are given in Figure 6.

3 - Generation of *peak dependent costs*: First, the off-peaks cost $c_i\,(u, v) \in \{lb_C, ..., ub_C\}$, $i = 1, 2$, are generated with a correlation as specified by $flag_{cor}$. After generating the off-peak costs the peak effect is taken into account. For each arc (u, v), the costs, if leaving node u at an off-peak time, are $\hat{c}_i\,(u, v, t) = c_i\,(u, v)$, $i = 1, 2$. At the beginning of a peak, the costs $\hat{c}_i\,(u, v, t)$ increase from $c_i\,(u, v)$ to $c_i\,(u, v)\,(1 + \psi)$, then stays the same during the pure peak period, and then decrease to $c_i\,(u, v)$ again. Finally, the costs $c_i\,(u, v, t)$, $i = 1, 2$ are found by applying the random perturbation to $\hat{c}_i\,(u, v, t)$. A plot of the costs for a specific arc are given in Figure 7.

4 - Peak dependent costs: The first cost act like for $flag_C = 3$. But the second cost decrease in peaks instead of increasing. A plot of the costs for a specific arc is given in Figure 8.

5 - Peak dependent costs: The off-peak costs $c_i\,(u, v)$ are generated as under $flag_C = 3$. If $c_1\,(u, v) < c_2\,(u, v)$ then a peak cost $c_1^p(u, v)$ is generated randomly in $\{c_1\,(u, v), \ldots, c_2\,(u, v)\}$ and a peak cost $c_2^p(u, v,)$ is generated randomly in $\{c_2\,(u, v), \ldots, ub_C\}$. As a result $c_1\,(u, v) \leq c_1^p(u, v,) \leq c_2\,(u, v) \leq c_2^p(u, v)$. Similar is the case $c_1\,(u, v) \geq c_2\,(u, v)$. Next, the costs $c_i\,(u, v, t)$, $i = 1, 2$ are found by applying the random perturbation to $c_i(u, v)$ in off-peaks and $c_i^p(u, v)$ in peaks. A plot of the costs for a specific arc is given in Figure 9.

6 - Peak dependent costs as if $flag_C = 3$ for horizontal arcs and costs are generated as for $flag_C = 2$ for vertical arcs. A plot of the costs for a vertical and a horizontal arc are given in Figure 10.

Symmetric travel costs may be used. However, note the costs are symmetric before the random perturbation is applied, i.e. $\hat{c}_i\,(u, v, t) = \hat{c}_i\,(v, u, t)$, $i = 1, 2$ and then the random perturbation is applied to obtain $c_i\,(u, v, t)$. As a result we do not have exact symmetry, costs may vary a bit. If you want exact symmetry you may set the range r_ξ of the random perturbation to zero. Symmetric travel costs are controlled with the $flag_{sym}$ flag. Possible options are:

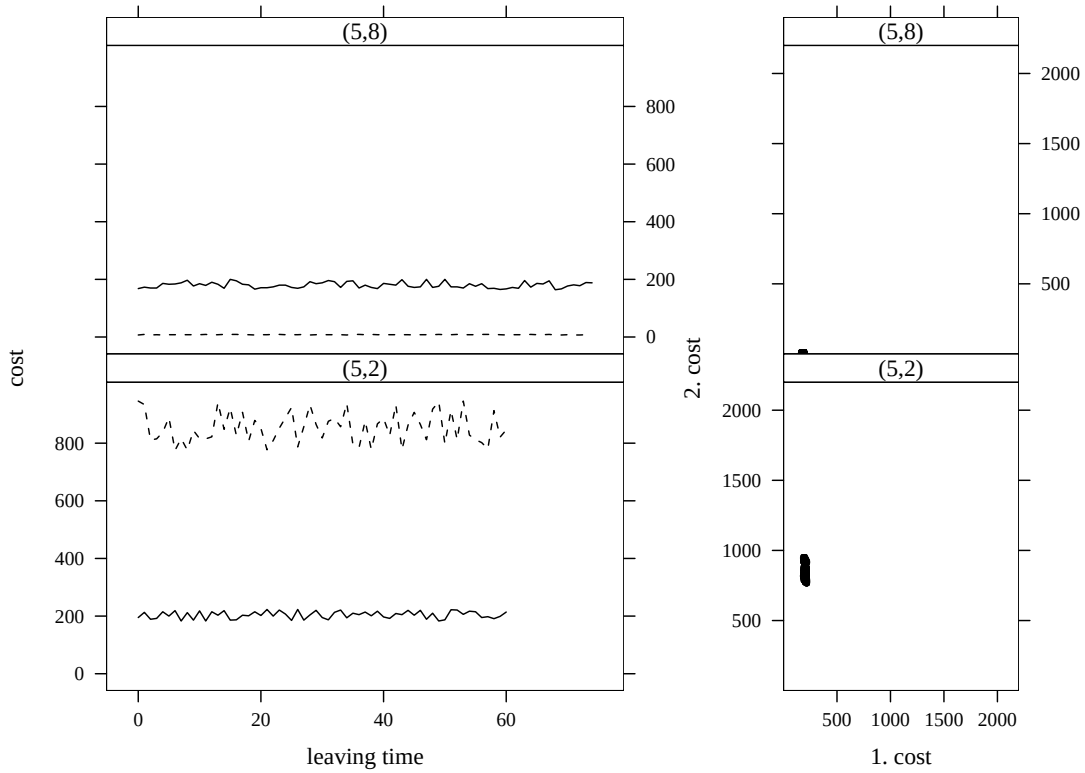


Figure 5: Travel costs for $flag_C = 1$ and $flag_{cor} = 2$.

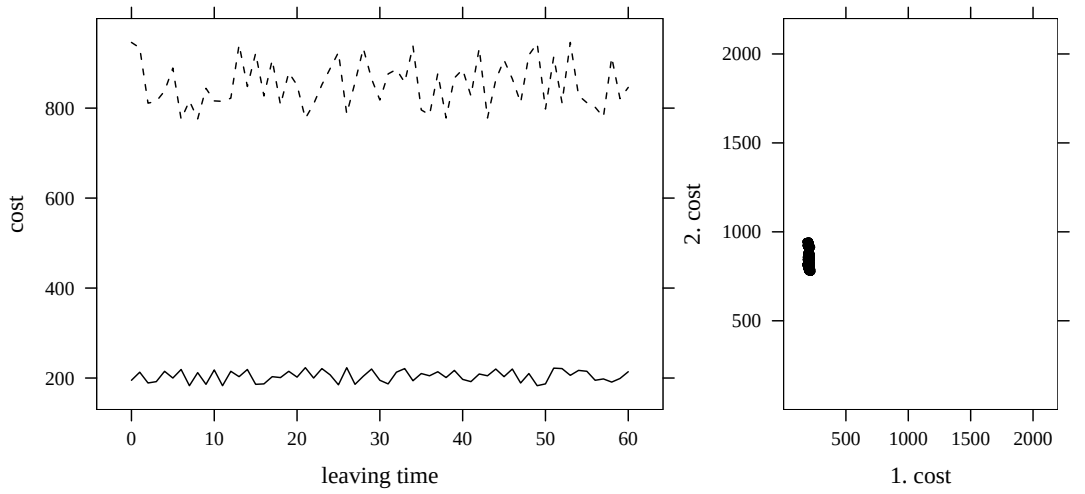


Figure 6: Travel costs for $flag_C = 2$ and $flag_{cor} = 2$.

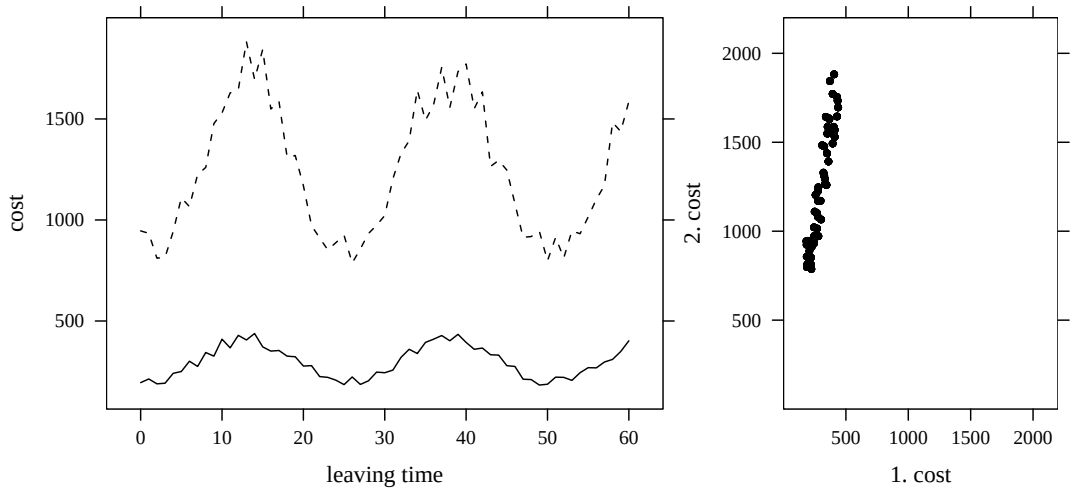


Figure 7: Travel costs for $flag_C = 3$ and $flag_{cor} = 2$.

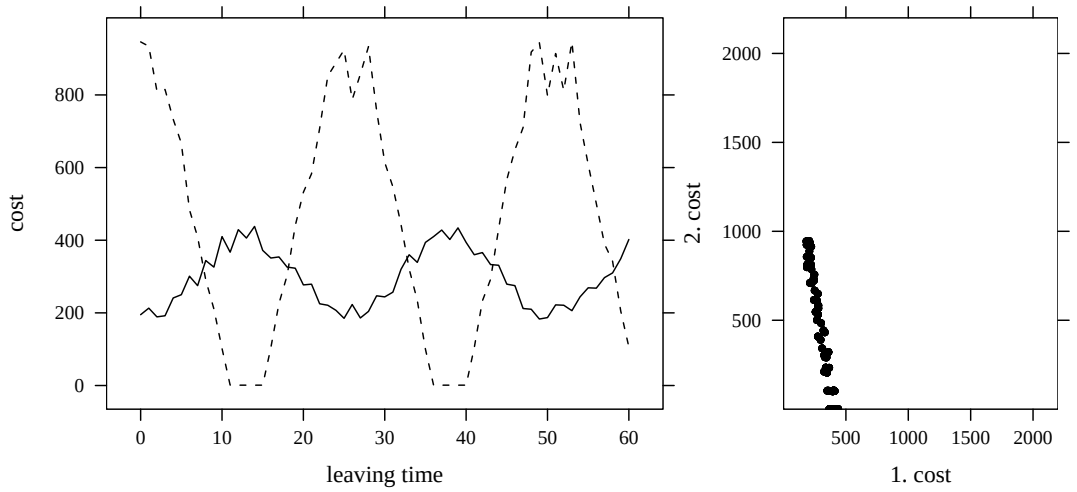


Figure 8: Travel costs for $flag_C = 4$ and $flag_{cor} = 2$.

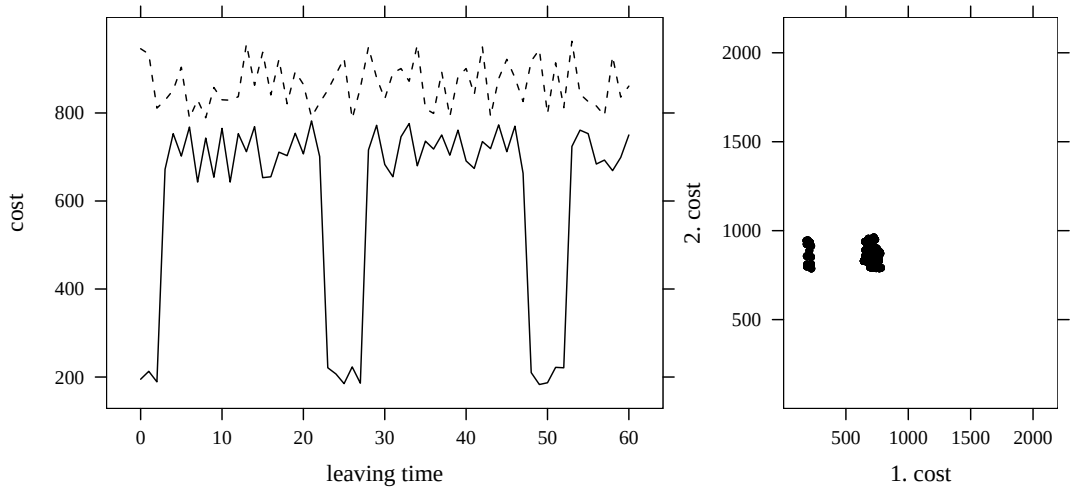


Figure 9: Travel costs for $flag_C = 5$ and $flag_{cor} = 2$.

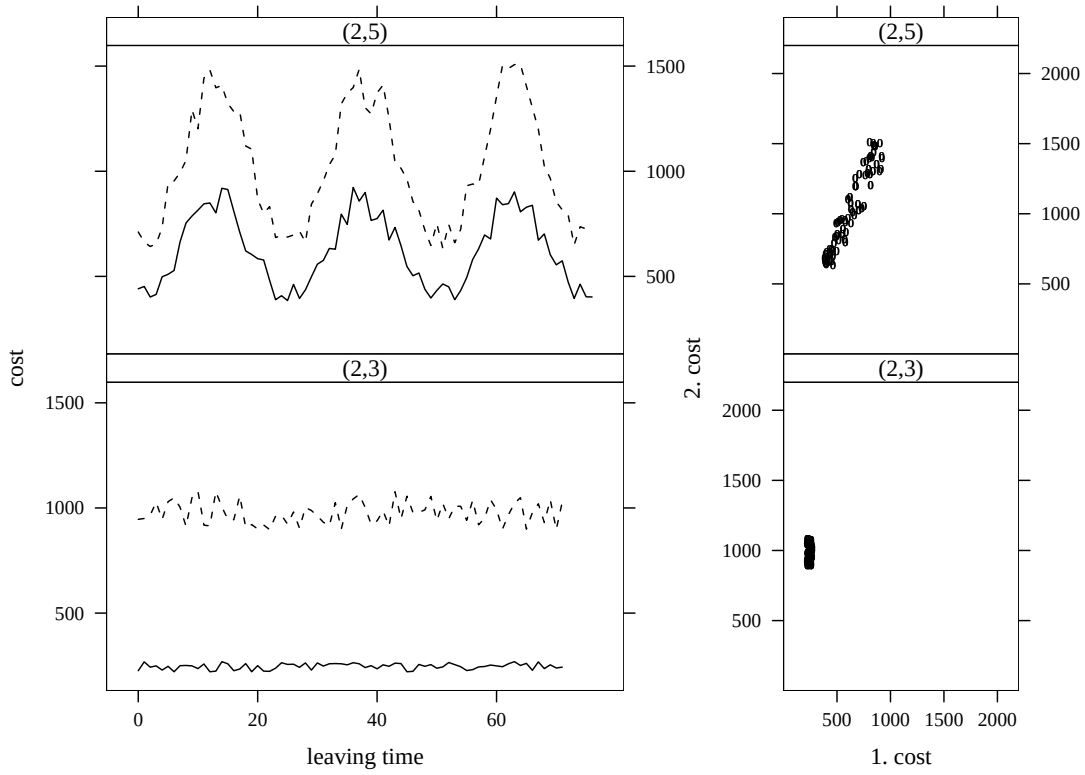


Figure 10: Travel costs for $flag_C = 6$ and $flag_{cor} = 2$.

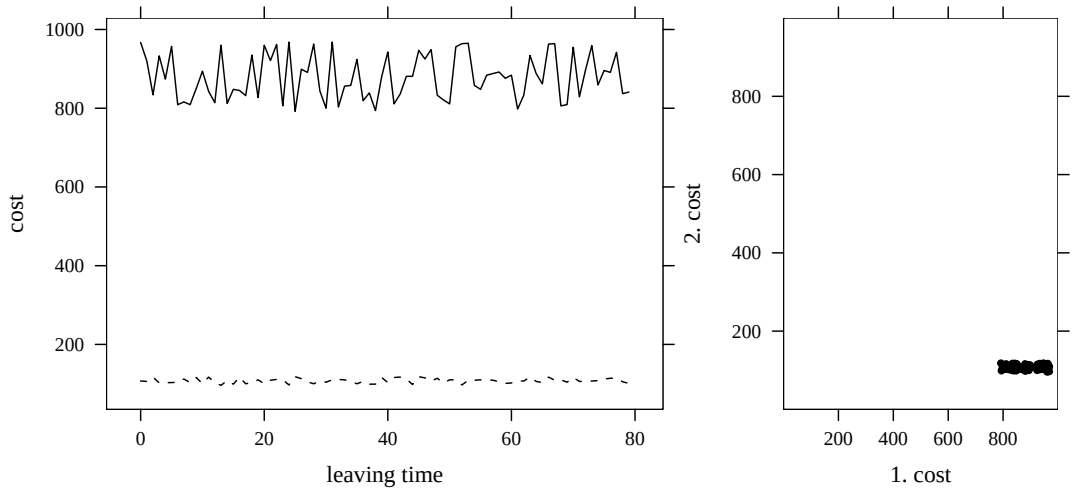


Figure 11: Waiting costs for $flag_W = 0$ and $flag_{cor} = 2$.

0 - Do not consider symmetry.

1 - Consider symmetry.

3.2 Generating waiting costs

Waiting are considered if the input parameter $ub_W > 0$. In this case waiting costs $c_i(u, t, t+1)$, $i = 1, 2$ for each node u in the grid except for the origin o and destination d are generated.

How waiting costs are generated are specified by the $flag_W$ flag. We have 2 possible values:

0 - Node waiting cost $c_i(u)$ is generated in $\{lb_W, ..., ub_W\}$ and correlated as specified by the $flag_{cor}$. Waiting costs $c_i(u, t, t+1)$ are then generated by applying the random perturbation. A plot of the waiting costs is given in Figure 11.

1 - Waiting costs $c_i(u, t, t+1)$ are generated independently for each leaving time in $\{lb_W, ..., ub_W\}$ and correlated as specified by the $flag_{cor}$. The random perturbation is not applied. A plot of the waiting costs is given in Figure 12.

Note the peak effect is not used for waiting costs.

3.3 Generating penalty costs

Penalty costs for the destination node d are generated if the input parameter $ub_P > 0$. Otherwise, the penalty costs are zero. How penalty costs are generated is specified by the $flag_P$ flag. We have 4 possible values:

0 - Penalty costs are generated independently for each arrival time in the interval $\{lb_P, ..., ub_P\}$ and correlated as specified by the $flag_{cor}$. A plot of the penalty costs is given in Figure 13.

1 - Both costs penalizes arrivals in the middle of the time horizon H. A plot of the penalty costs is given in Figure 14 (first and second cost equals the solid line).

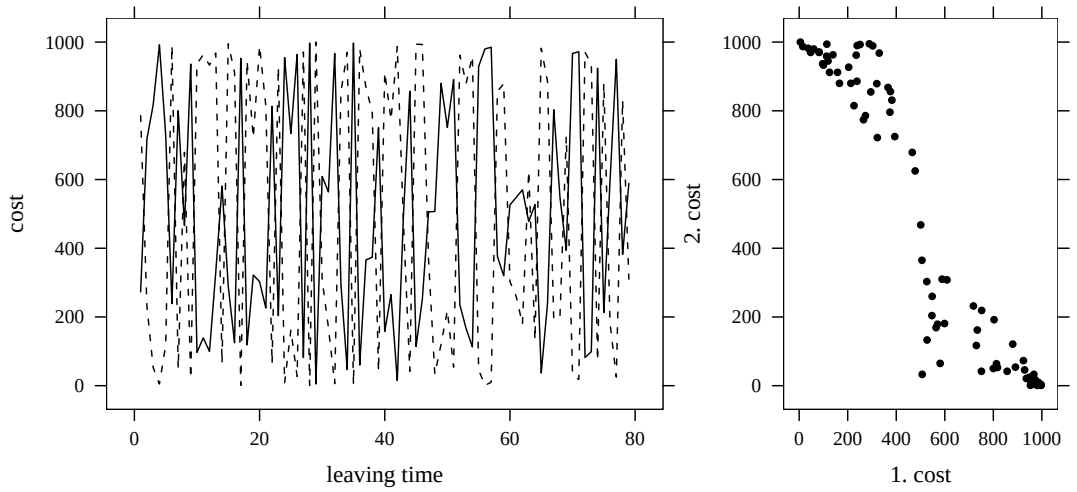


Figure 12: Waiting costs for $flag_W = 1$ and $flag_{cor} = 2$.

2 - Both costs penalizes early/late arrivals. A plot of the penalty costs is given in Figure 15.

3 - The second cost penalizes arrivals in the middle of the time horizon the first cost penalizes early/late arrivals. A plot of the penalty costs is given in Figure 16.

Note the random perturbation is NOT used when generating penalty costs and for $flag_P > 0$ the $flag_{cor}$ have no effect on the penalty costs.

4 Calculating the time horizon

The time horizon H for the STD network is not given as an input parameter to the TEGP generator. The time horizon depends on the size of G and the possible travel times generated for the arcs in G and is found using a preprocessing step:

First, note that due to (1) an upper bound on a possible travel time for an arc is $ub = (1 + \psi)(1 + \rho)ub_T$. As result $H_{ub} = (b + h)ub$ is an upper bound on the travel time of a path of length $b + h$. Upper bound ub is in general not very tight since there may be large fluctuations in travel time for different arcs and leaving time. Hence we use an estimate for the average maximal travel time for a path. This is done by generating all travel time distributions for each arc and leaving time for time horizon H_{ub}. Now by storing the maximum possible travel time for each distribution we can calculate the average maximum possible travel time ub_{ave}. We now set the time horizon H to

$$H = (b + h)ub_{ave}$$

As a result "roughly" all $o - -d$ paths of length $b + h$ can be traveled in the time horizon. Moreover, note that often there exists many paths containing arcs with maximum possible travel time below ub_{ave}. Therefore paths of length greater than $b + h$ may often be possible to travel in the time horizon.

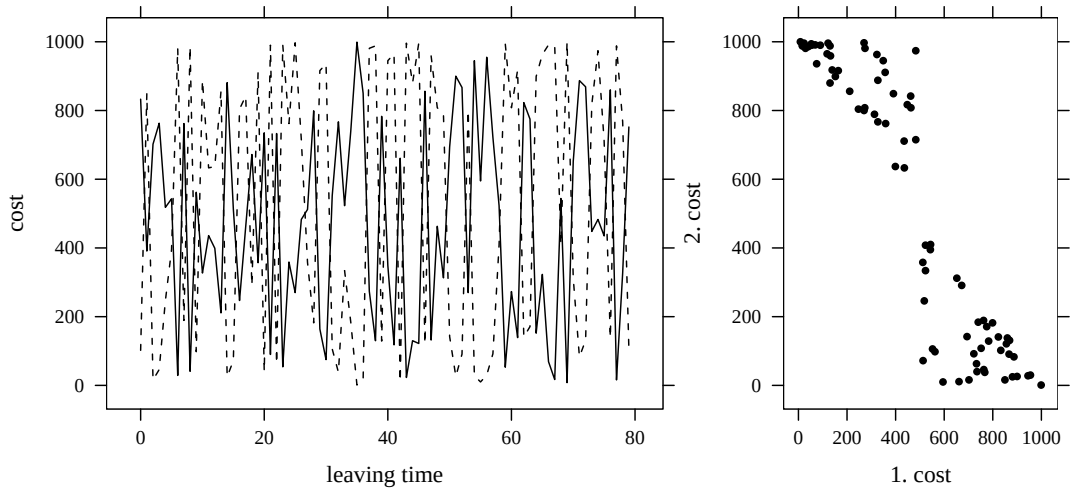


Figure 13: Penalty costs for $flag_P = 0$ and $flag_{cor} = 2$.

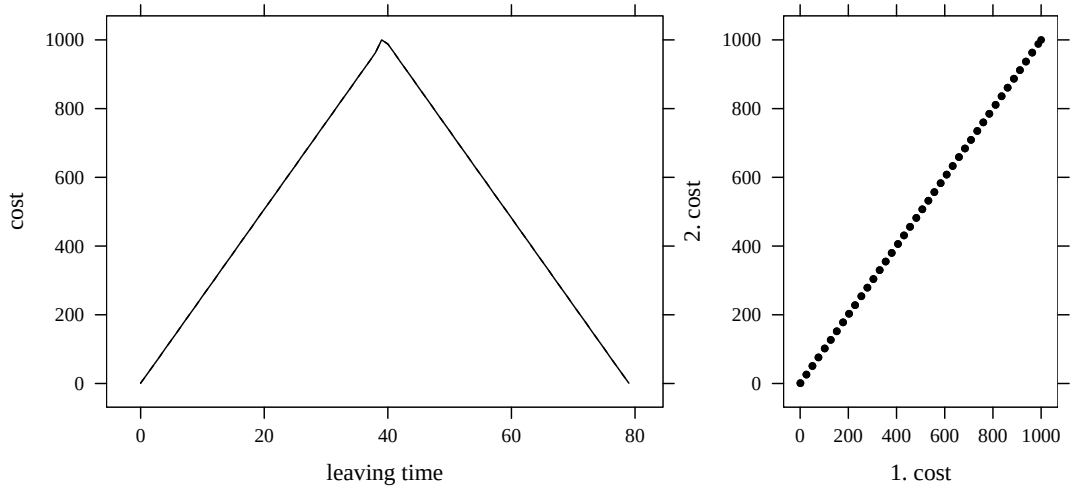


Figure 14: Penalty costs for $flag_P = 1$.

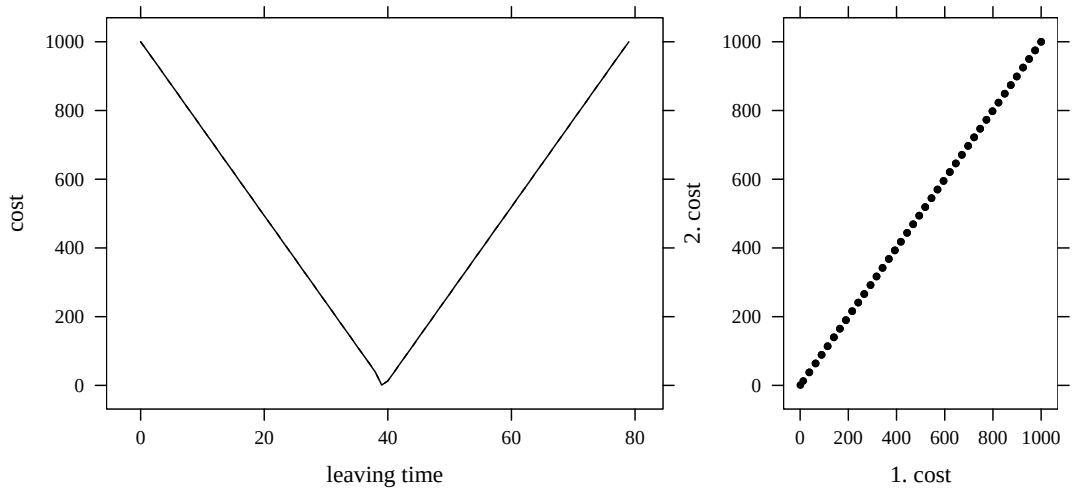


Figure 15: Penalty costs for $flag_P = 2$.

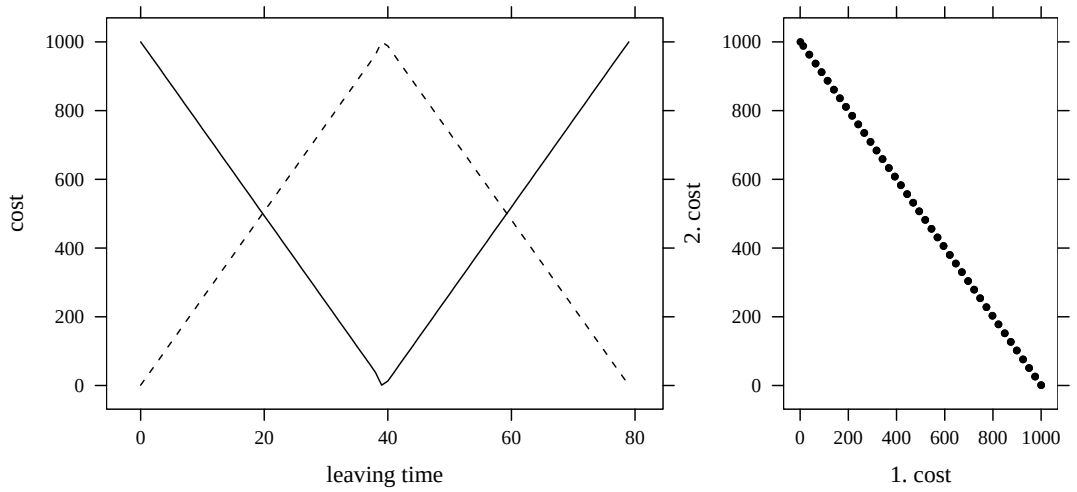


Figure 16: Penalty costs for $flag_P = 3$.

5 Running the program

The program uses command line passing for catching a number of run directives. The following flags/options can be used:

-verbose print out a lot of information to standard output (optional).

-out file name of the xml output file follows (without extension .xml).

Input is read from standard in which can be piped.

Example: `tegp > input -out output`

6 Input parameters

Input Parameters must be integers and are read from standard input or piped using a file with parameters separated by spaces as illustrated below[2].

$b\ h$	Base and height in grid.
H_{cycle}	Time instances in a cycle.
p	Number of peaks in a cycle.
t_{trans}	Time instances in transient peak.
t_{pure}	Time instances in pure peak.
t_p	First peak starting time.
ψ	Mean travel time/cost increase in peaks (pct).
ρ	Variance/mean ratio (pct).
$lb_P\ ub_P$	Min and max penalty cost. If $ub_P < 0$ then no waiting arcs are generated.
$flag_P$	Dependencies of the penalty costs (see Section 3.3).
$lb_T\ ub_T$	The mean traveltime interval.
$lb_W\ ub_W$	Min and max waiting cost. If $ub_W < 0$ then no waiting arcs are generated.
$flag_W$	Waiting costs flag (see Section 3.2).
$lb_C\ ub_C$	Min and max travel cost.
$flag_{cost}$	Dependencies of the travel costs (see Section 3.1).
$flag_{sym}$	Travel time symmetry flag (see Section 3.1).
$flag_{cor}$	Correlation type between costs (see Section 3.1).
$rand$	Random perturbation (promille, see Section 3.1).
$seed$	Seed.

7 Output

Output is written to <filename>.xml where <filename> is the filename specified by **-out**. The xml format is simple to understand and illustrated in Figure 17. Note, the probabilities of the travel time distribution is NOT normalized. This have to be done by the program which uses the test instance.

In general, the xml format is very verbose resulting in large file sizes. However, the xml file may be converted to a desired format, e.g. a time-expanded hypergraph [7], using an xslt stylesheet. For an introduction to xml and xslt see Møller and Schwartzbach [4].

[2]The parameters may also be given on a single line.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stdn nodes="9" arcs="20" timeHorizon="44" name="stdn.xml">
    <node number="1">
        <penalty t="0" c1="1000" c2="1"/>
        <penalty t="1" c1="954" c2="47"/>
        ...
    </node>
    ...
    <node number="4">
        <wait t="0" time="1" c1="485" c2="629"/>
        <wait t="1" time="1" c1="491" c2="693"/>
        ...
    </node>
    ...
    <arc head="1" tail="2">
        <leavingTime t="0" c1="872" c2="22">
            <travelTime t="2" prob="250000"/>
            <travelTime t="3" prob="500000"/>
            <travelTime t="4" prob="250000"/>
        </leavingTime>
        ...
    </arc>
    ...
</stdn>
```

Figure 17: The xml output file.

8 TEGP Class Index

8.1 TEGP Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

9 TEGP Class Documentation

9.1 GridNode Class Reference

Public Member Functions

- GridNode ()
- ∼GridNode ()
- void AllocMem (int H)
- void FreeMem ()

Public Attributes

- int cWait1

First off-peak cost for waiting in the node.

- int cWait2

  Second off-peak cost for waiting in the node.

- int traveltime [4]

  Mean off-peak travel times for grid arcs in forward star. Entrys are 0: north, 1: east, 2: south, 3: west.

- int(∗ c1)[4]

  First cost array for leaving at time t. Example c1[t][2] is the 1. cost for leaving the south arc at time t.

- int(∗ c2)[4]

  Second costs array leaving at time t. Example c2[t][2] is the 2. cost for leaving the south arc at time t.

9.1.1 Detailed Description

Class for representing a grid node and its forward star arcs.

Author: Lars Relund Nielsen.

Version: 1.5

9.1.2 Constructor & Destructor Documentation

9.1.2.1 GridNode::GridNode () `[inline]`

Constructor. Do not allocate memory which have to be done with AllocMem.

9.1.2.2 GridNode::∼GridNode () `[inline]`

Deconstructor. Free memory automatically.

9.1.3 Member Function Documentation

9.1.3.1 void GridNode::AllocMem (int *H*) `[inline]`

Allocate memory for the gridNode.

Parameters: ***H*** Time-horizon.

9.1.3.2 void GridNode::FreeMem () `[inline]`

Free memory.

9.2 Random Class Reference

Public Member Functions

- int Clock_seed ()

- void Init_len (int seed)
- int Int_length (int mid, int mad)
- void Init_w (int seed)
- int Int_weight (int l, int r)
- void Init_sign (int seed)
- int Sign ()
- void Init_num (int seed)
- int Int_number (int mid, int mad)
- double BinomPdf (double n, double p, double x)

9.2.1 Detailed Description

Class for generating random integer numbers etc. Modified code from previous class of Daniele Pretolani. Three routines can be used and a sign routine.

Author: Lars Relund Nielsen.

Version: 0.5

9.2.2 Member Function Documentation

9.2.2.1 double Random::BinomPdf (double *n*, double *p*, double *x*)

Calculate $Pr(X = x)$ when $X \sim bi(n, p)$. Modification of the code of Joe Nellis (`mrknowitall@mtcrossroads.org`) at `http://www.codeproject.com`.

Precondition: Do no checking, i.e. assume that $p \in [0, 1]$ and that $x \in [0, n]$.

9.2.2.2 int Random::Clock_seed ()

Return a seed using the clock value.

9.2.2.3 void Random::Init_len (int *seed*)

Initialization of random routine 1 (use X17).

9.2.2.4 void Random::Init_num (int *seed*)

Initialization of random routine 3 (use X15).

9.2.2.5 void Random::Init_sign (int *seed*)

Initialization of random sign routine (use X7).

9.2.2.6 void Random::Init_w (int *seed*)

Initialization of random routine 2 (use X11).

9.2.2.7 int Random::Int_length (int *mid*, int *mad*)

Returns an integer, in the interval {mid,...,mad}.

9.2.2.8 int Random::Int_number (int *mid*, int *mad*)

Returns an integer, in the interval {mid,...,mad}.

9.2.2.9 int Random::Int_weight (int *l*, int *r*)

Returns an integer, in the interval {l,...,r}.

9.2.2.10 int Random::Sign ()

Returns an zero/one value.

9.3 TegPeak Class Reference

Public Member Functions

- void Run ()
- TegPeak (string filename, bool verbose)
- ∼TegPeak ()

9.3.1 Detailed Description

The class for generating STD networks.

Assumes a underlying topological grid network of size $b \times h$. Every arc is "bi-directional", but no arcs enter the origin and no arcs exit the destination. Thus there are:

- h(b-1)-2 arcs east.
- b(h-1)-2 arcs south.
- h(b-1) arcs west.
- b(h-1) arcs north.

Grid nodes are numbered from the destination node 1 to the origin node $b \cdot h$. This number increases by one south and by h east. Thus, if u is the node with coordinates (x, y) (x right, y down), then node u have number

$$(x-1)h + y$$

This number is used to identify a node in the array of GridNode objects Moreover, each arc in a node u is numbered 0 (north), 1 (east), 2 (south) and 3 (west) if exists.

Author: Lars Relund Nielsen.

Version: 1.5

9.3.2 Constructor & Destructor Documentation

9.3.2.1 TegPeak::TegPeak (string *filename*, bool *verbose*)

Read input parameters. Moreover, set the output file name and the flag for verbose output.

9.3.2.2 TegPeak::∼TegPeak ()

Free memory.

9.3.3 Member Function Documentation

9.3.3.1 void TegPeak::Run ()

Run the generator and write the output to an xml file.

References

[1] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42:177–201, 1993.

[2] R.W. Hall. The fastest path through a network with random time-dependent travel times. *Transportation Science*, 20(3):182–188, 1986.

[3] E.D. Miller-Hooks and H.S. Mahmassani. Least possible time paths in stochastic, time-varying networks. *Computers & Operations Research*, 25:1107–1125, 1998.

[4] A. Møller and M.I. Schwartzbach. *An Introduction to XML and Web Technologies*. Addison-Wesley (in preparation), 2006.

[5] L.R. Nielsen. *Route Choice in Stochastic Time-Dependent Networks*. PhD thesis, Department of Operations Research, University of Aarhus, 2004.

[6] L.R. Nielsen, D. Pretolani, and K.A. Andersen. K shortest paths in stochastic time-dependent networks. Technical Report WP-L-2004-05, Department of Accounting, Finance and Logistics, Aarhus School of Business, 2004. URL `http://www.asb.dk/departments/afl/research/lrg/workingpapers/`. Submitted.

[7] D. Pretolani. A directed hypergraph model for random time-dependent shortest paths. *European Journal of Operational Research*, 123:315–324, 2000.

[8] A.J.V. Skriver and K.A. Andersen. A label correcting approach for solving bicriterion shortest path problems. *Computers & Operations Research*, 27:507–524, sep 2000.

A Stochastic time-dependent networks

In this section we describe discrete stochastic time-dependent networks. Moreover we also present a class of directed hypergraphs used to model STD networks. The hypergraph model is explained by means of an example. For further details see Nielsen [5], Pretolani [7].

A.1 Basic definitions

We consider discrete STD networks where departure times are integer, and travel times are independent integer-valued discrete random variables with time-dependent density functions. We assume that departure and arrival times belong to a finite *time horizon*, i.e. a set $H = \{0, 1, ..., t_{\max}\}$ of integer values. In practice, we assume that the relevant time period is discretized into time intervals of length δ, i.e., the time horizon H corresponds to the set of time instances $0, \delta, 2\delta, ..., t_{\max}\delta$.

Let $G = (N, A)$ be a directed graph with node set N and arc set A. We will refer to G as the *topological network*. As usual, $FS(u) = \{(u, v) \in A\}$ denotes the forward star of node u. Let $o \in N$ and $d \in N$ denote the *origin* and *destination* node in G, respectively. For each arc $(u, v) \in A$ let $L(u, v) \subset H$ be the set of possible leaving times from node u along arc (u, v). Moreover, let $L(u)$, $u \neq d$ denote the set of possible leaving times from node u, i.e.,

$$L(u) = \bigcup_{(u,v) \in FS(u)} L(u, v)$$

and let $L(d)$ denote the set of possible arrival times at node d. For each arc $(u, v) \in A$ and $t \in L(u, v)$, let $X(u, v, t)$ denote the travel time to node v when leaving node u at time t along arc (u, v). The travel time $X(u, v, t)$ is a discrete random variable with density

$$\Pr(X(u, v, t) = t_i) = \theta_{uvt}(t_i), \quad t_i \in I(u, v, t)$$

where

$$I(u, v, t) = \{t_1, ..., t_{\kappa(u,v,t)}\}$$

denotes the set of $\kappa(u, v, t)$ possible arrival times at node v when leaving node u at time t along arc (u, v). That is, for each $t_i \in I(u, v, t)$ the probability of arriving at node v at time t_i when leaving node u at time t is $\theta_{uvt}(t_i)$. Denote by

$$\kappa = \sum_{(u,v) \in A,\ t \in L(u,v)} \kappa(u, v, t)$$

the total number of possible travel times. The value κ can be considered as the size of the STD network.

We assume that travel times are positive and that the traveller cannot get stuck at an intermediate node v. Hence, if it is possible to arrive at node v at time t_i, then it is also possible to leave node v at time t_i. Note that a traveller cannot wait at intermediate nodes.

Definition 1 A *strategy* is a function S with domain

$$Dm(S) \subseteq \{(u, t) : u \in N \setminus \{d\},\ t \in L(u)\}$$

assigning to each pair $(u, t) \in Dm(S)$ a successor arc $(u, v) \in FS(u)$. Furthermore, strategy S must satisfy the following conditions:

1. If $(u, t) \in Dm(S)$ and $S(u, t) = (u, v)$ then $t \in L(u, v)$.
2. If $(u, t) \in Dm(S)$ and $S(u, t) = (u, v)$, $v \neq d$, then $(v, t') \in Dm(S), \forall t' \in I(u, v, t)$.

Strategy S provides routing choices for travelling from all nodes and leaving times in the domain $Dm(S)$ towards the destination d. Therefore, a traveller leaving node u at time t travels along arc $S(u, t)$. Note that a strategy S must provide a routing choice for each possible arrival time at an intermediate node, as required by Condition 2 above. Moreover, given S, we denote by:

$$Dd(S) = \{(d, t) :\ \exists (u, t') \in Dm(S),\ S(u, t') = (u, d),\ t \in I(u, d, t')\}$$

the set of pairs (d, t) where t is a possible arrival time at d when following strategy S. Note that Definition 1 extends the definition given in [7] where a strategy had domain

$$\mathcal{D} = \{(u, t) : u \in N \setminus \{d\},\ t \in L(u)\}. \tag{2}$$

Our assumption that a traveller cannot get stuck at intermediate nodes implies that any traveller who leaves node i at time t arrives at the destination d within time $t_{\max}$. This can be formally stated requiring that a pair (i, t) belongs to $\mathcal{D}$ if and only if it belongs to the domain of some strategy.

In this paper we consider strategies providing route choices when leaving a specific node i at a specific time t towards the destination d. This leads to the definition of an (i, t) strategy.

Definition 2 An (i,t) strategy is a *minimal* strategy S such that $(i,t) \in Dm(S)$. Here, minimality means that there does not exist another (i,t) strategy with domain strictly contained in $Dm(S)$.

In particular, we are interested in $(o,0)$ strategies, defining the route travelled when leaving the origin node o at time zero. In the following, unless otherwise specified, a strategy S refers to a $(o,0)$ strategy.

A strategy is a *path-strategy* if the successor arcs do not depend on time; in other words, a path-strategy must satisfy

$$S\left(u,t\right) = S\left(u,t'\right), \quad \forall \left(u,t\right), \left(u,t'\right) \in Dm\left(S\right). \tag{3}$$

Clearly, a path-strategy S defines a unique, loopless o-d path in G. Indeed, S defines a unique successor arc (u,v) for each u in G such that $(u,t) \in Dm(S)$ for some t. Assume that for each arc (o,v) in G it is possible to leave o at time zero travelling along (o,v), i.e., $0 \in L(o,v)$. With this assumption, each loopless o-d path

$$P = (o = u_1, u_2, \ldots, u_l, u_{l+1} = d)$$

in G defines exactly one path-strategy S. In particular, we have

$$Dm(S) = \bigcup_{i=1}^{l} D^{(i)}, \qquad Dd(S) = D^{(l+1)} \tag{4}$$

where $D^{(1)} = \{(o,0)\}$ and, for $1 < i \leq l+1$:

$$D^{(i)} = \left\{(u_i,t) :\ t \in I(u_{i-1},u_i,t'),\ (u_{i-1},t') \in D^{(i-1)}\right\}. \tag{5}$$

Clearly, we have $S(u_i,t) = (u_i,u_{i+1})$ for each $(u_i,t) \in D^{(i)}$, $1 \leq i \leq l$. Based on the above observations, we can state the following theorem.

Theorem 1 *There is a one-to-one correspondence between o-d paths in G and path-strategies in the STD network.*

A.2 Optimality criteria

Several definitions of the *weight* of a strategy can be given, in fact, several optimality criteria have been considered in the literature. The most frequently used criterion for finding the best strategy is the minimization of the expected travel time, introduced by Hall [2]. In this case, the weight of a strategy corresponds to the expected arrival time at the destination when leaving the origin at time zero. If travel costs are considered, strategies can be ranked according to their expected cost. Moreover, instead of considering expectations, worst cases may be of concern; i.e., our criterion becomes the minimization of *maximum possible* travel time or cost. Other criteria, such as the *minimum possible* travel time, have been considered in the literature [3].

The results reported in this paper apply to each of the criteria mentioned above. In our computational experience we shall concentrate on expected costs. Costs can be introduced in our STD model by letting $c\left(u,v,t\right)$, $t \in L\left(u,v\right)$ denote the travel cost of leaving node u at time t along arc (u,v). Note that we assume that c is deterministic, although time-dependent. Moreover, let $g_d(t)$ be a penalty cost of arriving at node d at time t. The expected cost of a strategy S can be defined by means of recursive equations, associating a value to each pair (u,t) in $Dm(S)$. In particular, if $S(u,t) = (u,v)$, we have:

$$E^S(u,t) = c(u,v,t) + \sum_{t' \in I(u,v,t)} \theta_{uvt}(t') E^S(v,t')$$

where $E^S(d,t) = g_d(t)$ for each $t \in H$. Here, $E^S(u,t)$ represents the expected cost (including penalty costs) incurred when leaving node u at time t following strategy S towards d. The expected cost of S is therefore $E^S(o,0)$. The other criteria cited above can be given a formal definition using similar recursive

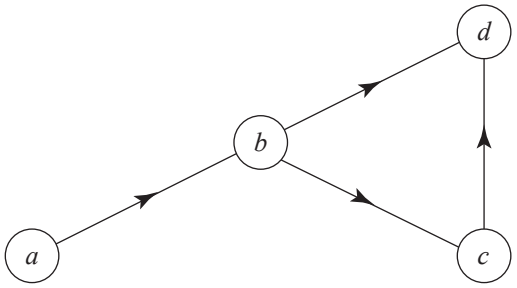


Figure 18: The topological network G.

$(u,v),t$	$(a,b),0$	$(b,c),1$	$(b,c),2$	$(b,d),1$	$(b,d),2$	$(c,d),2$	$(c,d),3$	$(c,d),4$
$I(u,v,t)$	$\{1,2\}$	$\{2,3\}$	$\{3\}$	$\{3\}$	$\{6\}$	$\{3,4\}$	$\{4,5\}$	$\{5,6\}$

Table 1: Input parameters.

equations, see Pretolani [7]. Recall that for all the above criteria, finding the best path-strategy is an $\mathcal{NP}$-hard problem, whereas the best strategy can be found in $O(\kappa)$ time, see Pretolani [7]. Moreover, recall that path-strategies are a subset of strategies and, therefore, the weight of the best strategy provides a (quite efficiently computable) lower bound on the weight of the best path-strategy.

A.3 A directed hypergraph model for STD networks

A *directed hypergraph* is a pair $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = (v_1, ..., v_n)$ is the set of *nodes*, and $\mathcal{E} = (e_1, ..., e_m)$ is the set of *hyperarcs*. A hyperarc $e \in \mathcal{E}$ is a pair $e = (T(e), h(e))$, where $T(e) \subset \mathcal{V}$ denotes the set of *tail* nodes and $h(e) \in \mathcal{V} \setminus T(e)$ denotes the *head* node. Note that a hyperarc has exactly one node in the head, and one or more nodes in the tail. The *cardinality* of a hyperarc e is the number of nodes it contains, i.e., $|e| = |T(e)| + 1$. We call e an *arc* if $|e| = 2$. The *size* of $\mathcal{H}$ is the sum of the cardinalities of its hyperarcs.

In particular, we here consider *acyclic* hypergraphs, where there exists a *valid ordering* $V = (v_1, v_2, \ldots, v_n)$ of the nodes such that, for any $e \in \mathcal{E}$, each node $v_j \in T(e)$ precedes node $h(e)$ in V. The class of directed hypergraphs used here was denoted acyclic B-graphs in Gallo, Longo, Pallottino, and Nguyen [1] which considered the general class of directed hypergraphs. However, we here use the term "hypergraph" to denote the subclass appropriate in this context.

As shown in Pretolani [7] a *time-expanded hypergraph* $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ can be used to model an STD network. We illustrate the model by means of the following example.

Example 1 Consider the topological network $G = (N, A)$ in Figure 18, where a is the origin node and d is the destination node. For each arc in G, the possible departure and arrival times are listed in Table 1. Here a pair $((u, v), t)$ corresponds to a possible leaving time t from node u along arc (u, v). For the sake of simplicity, we assume that $X\,(u, v, t)$ has a uniform density, i.e., for each $t' \in I(u, v, t)$, we have $\theta_{uvt}(t') = 1/|I(i, j, t)|$. For example, if we leave node c at time 2 along arc (c, d), we arrive at node d at time 3 or 4 with the same probability $1/2$.

The time-expanded hypergraph $\mathcal{H} =(\mathcal{V}, \mathcal{E})$ is shown in Figure 19; numbers and dotted lines will be ex-

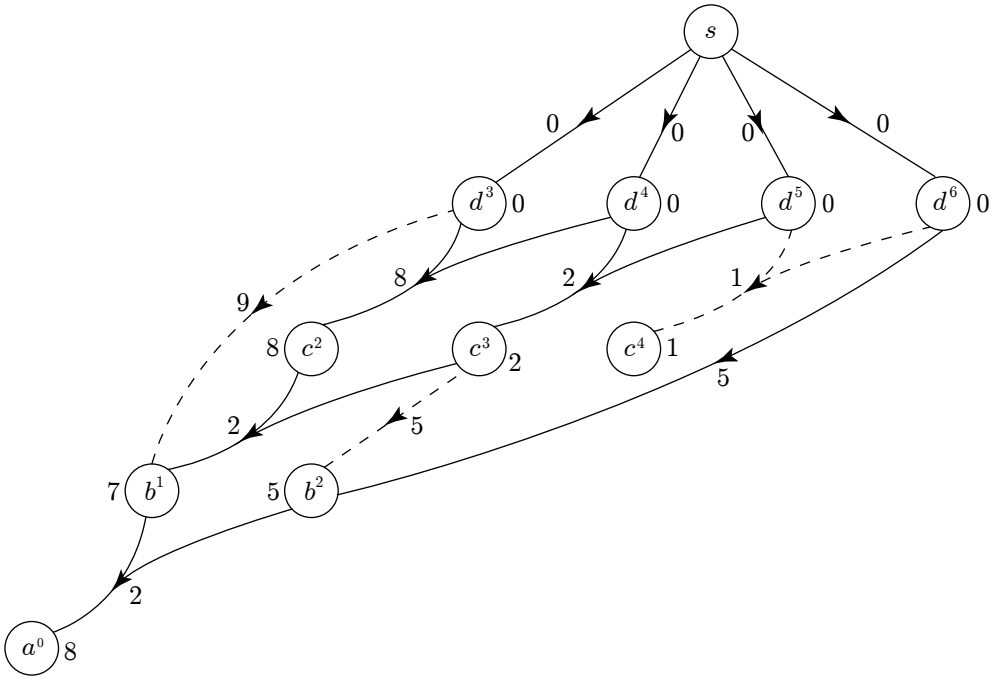


Figure 19: The time-expanded hypergraph $\mathcal{H}$.

plained below. The set $\mathcal{V}$ contains one node u^t for each pair (u,t), $t \in L(u)$ and an origin node s. For each $(u,v) \in A$ and $t \in L(u,v)$, we introduce a hyperarc

$$e_{uv}(t) = \left(\{v^{t_i} : t_i \in I(u,v,t)\}, u^t\right).$$

Moreover, a dummy arc $e_d(t) = (\{s\}, d^t)$ is defined for each $t \in L(d)$. ■

It is obvious that $\mathcal{H}$ is an acyclic hypergraph: a valid ordering can be found by ranking the nodes in *decreasing* order of time. Furthermore, the size of $\mathcal{H}$ is $O(\kappa)$ and $\mathcal{H}$ can be built in $O(\kappa)$ time. Given any strategy S, let us define the sets

$$\mathcal{V}_S = \left\{u^t : (u,t) \in Dm(S) \cup Dd(S)\right\} \cup \left\{s\right\}$$

and

$$\mathcal{E}_S = \left\{e_{uv}(t) : (u,t) \in Dm(S),\, S(u,t) = (u,v)\right\} \cup \left\{e_d(t) : (d,t) \in Dd(S)\right\}.$$

Note that $\mathcal{V}_S$ contains a node u^t for each pair (u,t) corresponding to either leaving from an intermediate node or arriving at the destination. Moreover, for each node $u^t \in \mathcal{V}_S$, a single hyperarc in $\mathcal{V}_S$ with head u^t exists, more precisely, a dummy arc $e_d(t)$ if $u = d$, and a hyperarc $e_{uv}(t)$ if $u \neq d$. Let us denote by $r = o^0$ the node in $\mathcal{H}$ corresponding to the pair $(o,0)$. It has been shown in Pretolani [7] that $\pi_S = (\mathcal{V}_S, \mathcal{E}_S)$ is a *hyperpath* from node s to node r (s-r hyperpath) in $\mathcal{H}$. More precisely, the following property holds:

Property 1 *There is a one-to-one correspondence between $(o,0)$ strategies and s-r hyperpaths in $\mathcal{H}$.*

Pretolani [7] showed that the value of a $(o,0)$ strategy under each one of the optimality criteria in Section A.2 corresponds to the weight of the corresponding s-r hyperpath in $\mathcal{H}$ for a suitable definition of hyperpath weight, given in terms of *additive weighting functions*, see Gallo et al. [1]. Therefore, the best strategy can be found by finding the minimum weight s-r hyperpath, i.e., by solving a *shortest hyperpath* problem in $\mathcal{H}$. Quite efficient procedures for finding shortest hyperpaths are defined in Gallo et al. [1];

for acyclic hypergraphs, the computational complexity is linear in the size of the hypergraph. Thus, under time-adaptive route choice, the best strategy can be found in $O(\kappa)$ time. Clearly, this result does not extend to the a priori case in which a path-strategy is required.

Example 1 (continued) Hyperarcs in solid lines in Figure 19 represent the s-r hyperpath π_S corresponding to the best $(o, 0)$ strategy S for the expected cost criterion. Close to each hyperarc $e_{uv}(t)$, we report cost $c(u, v, t)$; penalty costs are zero and reported close to dummy arcs $e_d(t)$. The number close to each node u^t is the expected travel cost $E_S(u, t)$; thus, the minimum expected cost is 8. We have $S(c, t) = (c, d)$ for each time $t = 2, 3$, thus, S defines a unique successor for node c. However, the successor of b is (b, c) at time 2 and (b, d) at time 3, thus S is not a path-strategy.

Note that S defines the time-adaptive route to travel when leaving node a at time zero, but it does not define a successor for all possible nodes and leaving times, e.g. for node c at time 4. A non-$(o, 0)$ strategy S' defining the route to travel for all possible nodes and leaving times (i.e., with domain $Dm(S') = \mathcal{D}$) is obtained by adding the pair $(c, 4)$ to $Dm(S)$ and defining $S'(c, 4) = (c, d)$.

Note also that it is not possible to arrive at node c at time 4. Thus, the pair $(c, 4)$ might be eliminated from $\mathcal{V}$, i.e., time 4 might be eliminated from $L(c)$ an $L(c, d)$. However, pair $(c, 4)$ shall be used later, in relation to waiting.

■