Mintc: transit assignment with the Mint algorithm

Reference documentation. Parallel version, with weights by link type.

1. Purpose and scope

Mintc assigns an origin-destination matrix to a transit network described by frequencies (headways between vehicles). It implements the **Mint** algorithm (P. Palmier, CETE Nord Picardie). Mint keeps the principle of the optimal strategies of Spiess and Florian (1989) and addresses three limits:

- **L1**: the split between attractive lines also depends on travel times, not only on frequencies.
- **L2**: walking links take part in the split as strategies with infinite frequency.
- **L3**: a user can stay in the vehicle or alight to transfer, whichever costs less.

The program produces:

- the volume, boardings and alightings of every link;
- the expected time, the expected generalized cost and the average number of boardings of every OD pair;
- the attractive strategies of every link, for each destination (optional).

The computation is done by destination. Destinations are processed in parallel on several threads.

This document describes the program as delivered. It does not replace the reference paper: *Mint: proposition of a new transit assignment algorithm for frequency based networks* (Palmier).

2. Quick start

Command line

```
Mintc network.csv matrix.csv output parameters.txt [nb_threads]
```

Argument	Role
`network.csv`	Network file (section 3.1)
`matrix.csv`	Matrix file (section 3.2)

Argument	Role
`output`	Prefix of the result files (section 7)
`parameters.txt`	Parameter file (section 3.3)
`nb_threads`	Optional. Number of threads. Default: number of logical cores. Capped at the number of destinations.

Without the first four arguments, the program prints the usage reminder and stops.

Build

Single-file C# console application, `Program.cs`, namespace `Mintc`. No external dependency. It was compiled and tested with the Mono compiler `mcs`. It was not tested with other compilers.

Files produced

`output_od.txt`, `output_aff.txt`, `output_times.txt`, `output_stg.txt`, `output_log.txt`.

3. Input files

All files are text files, `;` separator, no header line. Numbers are read with the machine's culture (see section 9).

3.1 Network

One line per line segment or walking link.

Col.	Name	Type	Meaning
1	`i`	text	Initial node
2	`j`	text	End node
3	`line`	text	Line identifier
4	`time`	real	Travel time, in minutes
5	`hdwy`	real	Headway between vehicles, in minutes. **0 = walking link or connector, no waiting**
6	`vcap`	real	Vehicle capacity. Read, **not used**

Col.	Name	Type	Meaning
7	`allowboai`	integer	1: boarding allowed at the initial node. 0: forbidden
8	`allowalij`	integer	1: alighting allowed at the end node. 0: forbidden
9	`type`	text	Optional. Link type, for weights by type (section 4)

Rules:

- Nodes are created automatically from columns 1 and 2.
- Spaces around `i`, `j`, `line` and `type` are ignored.
- Empty lines are ignored.
- A non-empty line with fewer than 8 fields stops the program, with its line number.
- The restrictions in columns 7 and 8 are applied.

Modelling conventions:

- `hdwy > 0`: transit line segment. Consecutive segments with the **same line identifier** mean staying in the same vehicle. Changing to another line identifier is a boarding.
- `hdwy = 0`: walking, or a connector between a zone and a stop.
- An origin zone should have **a single outgoing link** if its demand is to be split between several lines at the stop. See section 9.

3.2 Matrix

Col.	Name	Meaning
1	`o`	Origin node identifier
2	`d`	Destination node identifier
3	`demand`	Demand volume, real

Line order is free. The computation is grouped by destination, in the order in which each destination first appears. Empty lines are ignored. A non-empty line with fewer than 3 fields stops the program.

3.3 Parameters

The first six lines are mandatory, in this order, one value per line.

Line	Name	Meaning
1	`cwait`	Weight of waiting time. **Must be > 0**
2	`cmap`	Weight of walking time (global value)
3	`tboa`	Boarding time, in minutes
4	`cboa`	Boarding weight (global value)
5	`output_strategies`	`true` or `false`: write `output_stg.txt` or not
6	`alg_parameter`	Computation parameter: size of the buckets used to order the links

Optional lines, after the first six:

Format	Effect
`type;cmap;cboa`	Weights specific to link type `type` (section 4)
`tboa_dans_temps;1`	Adds `tboa` to the `temps` column. Default: 0 (behaviour of the paper)
`# comment`	Ignored line. Empty lines are ignored too

A malformed optional line stops the program. So does a type defined twice.

Example file:

```
1
2
2
1
true
10
# type;cmap;cboa
acces;6;1
marche;3;1
metro;2;0.5
bus;2;2
```

4. Generalized cost and weights by link type

The cost of a trip is the sum of four components:

Component	Cost
In-vehicle time (link with `hdwy > 0`)	`time` (weight 1)
Walking time (link with `hdwy = 0`)	`time × cmap`
Boarding	`tboa × cboa`
Waiting	`cwait × headway` (see section 5)

Weights by type. Each link can carry a type (column 9). A `type;cmap;cboa` line in the parameter file defines weights for that type:

- `cmap` applies to the walking time **of the link travelled**, according to that link's type. It lets you tell access, egress and transfer walking apart.
- `cboa` applies to the boarding penalty, according to the type **of the boarded link**. It lets you tell, for example, bus from metro.
- `tboa` stays global.

A link with no type, or whose type is not listed, uses the global `cmap` and `cboa` (lines 2 and 4). Types are case-sensitive. A `cmap` given to a transit type, or a `cboa` given to a walking type, has no effect.

When do weights change the strategies? Attractiveness and flow shares depend on the *differences* in cost between alternatives. A penalty common to all alternatives of a link shifts all costs by the same amount and changes nothing. So `cboa` only matters if the alternatives involve different numbers of boardings, and `cmap` only if they involve different walking times. If `tboa = 0`, `cboa` has no effect.

5. Algorithm

5.1 Principle

For each destination, the network is explored backwards from the destination. Every link receives:

- a **set of attractive strategies**, at most one per line;
- an **expected average cost** to the destination (`avg_cout`) and an **expected average time** (`avg_time`).

Links are processed in the order of a bucket labelling: the next link is taken from the non-empty bucket with the lowest index. A link that was already processed can be reopened if a better strategy appears.

5.2 A strategy

A strategy of a link `u` describes "take this line next". It carries:

Field	Meaning
`line`	The strategy's line. Unique key within the link: a line can appear only once
`hdwy`	Headway of the line; 0 for a strategy with no waiting (walking)
`n`	Frequency, `60 / hdwy`, in vehicles per hour
`c`	Minimum cost: average cost of the pivot, plus the cost of the link, plus boarding
`t`	Corresponding minimum time, unweighted
`p`	Share of the demand that follows this strategy
`pivot`	Next link. -1: end of the trip at the destination

5.3 Formulas

Notation: `n_k = 60 / hdwy_k`, `c_k` is the minimum cost of strategy `k`. Sums run over the transit strategies (`hdwy > 0`) of the link.

- **M**: `M = (60·cwait + Σ n_k·c_k) / Σ n_k`. This is the expected maximum cost when the lines are combined.
- **M0**: attractiveness threshold of the link. It is the minimum of `M`, of the best `c + hdwy·cwait` and of the cost of walking strategies.
- **Attractiveness**: a strategy is attractive if `c ⩽ M0`. The others are removed. `M` and `M0` are then recomputed, until stable.
- **Share of a transit strategy**: `p_k = (M0 − c_k) / (hdwy_k · cwait)`. It depends on times, not only on frequencies (limit L1).
- **Share of a walking strategy** (`hdwy = 0`): `p = (M − M0) · Σn / (60 · cwait)`. If there is no transit strategy, `p = 1`.
- **Average cost**: `avg_cout = Σ p_k · 0.5 · (c_k + M0)`.
- **Average time**: `avg_time = Σ p_k · (t_k + 0.5 · (M0 − c_k) / cwait)`. By default, this time includes neither weights nor `tboa`.

Shares sum to 1.

Mapping to the paper's notation. In the code, `M0` is the paper's `M` (walking included) and the code's `M` is the paper's `M'` (walking excluded). `c` is the minimum time `μ`.

5.4 Creating and improving strategies

- A link reached for the first time receives a strategy. If it changes line, this is a boarding: the strategy is built from the pivot's average cost. Otherwise it inherits the pivot's strategies, with its own time and cost added.
- A link already reached can be improved. If the new strategy of a given line costs less than the existing one (smaller `c`), it replaces it. If the improvement does not reduce the link's average cost, the link goes back to its previous state.
- **Independence of strategies**: a line appears only once per link.
- **Cycles**: a link is added only if it does not already belong to the links that make up the existing strategies.
- `allowboai` and `allowalij` restrict boardings and alightings.

5.5 Loading the demand

For each OD pair:

1. The demand is placed on the outgoing link of the origin that has the smallest `avg_cout` among the reached links.
2. It is propagated from link to link by the shares `p`, in decreasing order of `pos`, towards the pivot of each strategy.
3. The volume is added to every link crossed. A boarding is counted when the pivot is a transit link of another line (or the first transit link). An alighting is counted when leaving a transit line for another one, or at the destination.

`ncorr` is the total number of boardings divided by the demand: it is the **average number of boardings** of the OD pair. A direct trip on one line gives 1. A walk-only trip gives 0.

6. Parallel execution

- **Unit of work**: one destination. Destinations are computed independently.
- **One network copy per thread**: each thread works on its own copy. Nodes, which are never modified, are shared.
- **Order of results**: results are written and accumulated in the original destination order, whatever the number of threads. The five output files are identical between 1 and 8 threads (tested).
- **Memory**: one network copy per thread, plus results waiting to be written, limited to 4 times the number of threads.

- **Error in a thread**: the program stops with a message that carries the original error.
- **Progress display**: `Destinations: k/N`, where `k` is the number of completed destinations.
- **Precision**: the totals in `output_aff.txt` are added destination by destination. They may differ from the sequential version by about 1e-6 in relative terms, because of `float` rounding.

The speed gain depends on the machine and on the network. It was not measured (the test environment had a single core).

7. Output files

Files start with the `output` prefix given as an argument. Numbers are written with the machine's culture.

`output_od.txt` : results by OD pair

Header: `o;d;temps;cout;ncorr`. One line per OD pair of the matrix, in destination order.

Column	Content
`o`, `d`	Origin and destination
`temps`	Expected average time (waiting included). Unweighted. Does not include `tboa`, unless the `tboa_dans_temps` option is set
`cout`	Expected average generalized cost
`ncorr`	Average number of boardings

An OD pair in error is written `o;d;;` (see section 8).

`output_aff.txt` : assignment by link

Header: `num;i;j;ij;line;volau;boai;alij`. One line per link with a strictly positive volume.

Column	Content
`num`	Link number (order of the network file, from 0)
`i`, `j`, `ij`, `line`	Nodes, `i-j`, line
`volau`	Link volume

Column	Content
`boai`	Boardings
`alij`	Alightings

`output_times.txt` : costs and times by link and destination

Header: `d;id;i;j;line;ij;hdwy;time;cout;temps`. For each destination, one line per link whose average cost is strictly positive: `cout` = `avg_cout` and `temps` = `avg_time` of the link towards that destination. This file is always written and can be very large (destinations × links).

`output_stg.txt` : strategies

Written only if `output_strategies = true`. The file is created in all cases and stays empty if the option is `false`. For each destination: one line with its name, then, for each link:

- a link line: `id;i;j;line;hdwy;time;m;M;M0;n;avg_cout;avg_time;pos;reached`;
- one line per strategy, starting with a space: `line;hdwy;n;p;c;C;pivot`.

For transit strategies, the `C` column repeats the link's `M`, shared by all its strategies (see section 9).

`output_log.txt` : log of OD pairs in error

Header: `ligne;raison;contenu`. See section 8.

8. Errors and log

OD pair in error. The OD pair is written `o;d;;;` in `output_od.txt` and a line is added to the log: `ligne;raison;contenu`, with the matrix line number, the reason and the original line. The log is sorted by destination, in processing order. Possible reasons (the text written to the log is in French):

Reason	Case
demande nulle, négative ou non numérique	`demand ⩽ 0`
origine inconnue du réseau	The node does not exist in the network file
destination inconnue du réseau	Same for the destination

Reason	Case
destination inatteignable depuis l'origine	No outgoing link of the origin is reached from the destination

An OD pair in error loads no demand.

Blocking errors. They stop the program with a message:

- network line with fewer than 8 fields, matrix line with fewer than 3 fields (line number given);
- malformed type line in the parameters, or a type defined twice;
- `cwait ⩽ 0`;
- error in a computation thread.

9. Points of attention and limits

1. **Decimal separator.** Numbers are read and written with the machine's regional culture. On a French Windows, the comma is the decimal separator: parameter and output files follow that setting. Use the same setting to write and to read the files.
2. **Origin.** All the demand of an OD pair leaves through the **single** outgoing link of the origin that has the smallest average cost. There is no split between several outgoing links. To split demand between several lines, the zone must have one connector to a stop node where the strategies are chosen.
3. **Type 2 walking.** Walking treated as an infinite-frequency line in the middle of a trip is not implemented. A link with `hdwy = 0` followed by a line behaves like the paper's "type 1" walking (it inherits its pivot's strategies). Walking straight to the destination is, however, an infinite-frequency strategy.
4. **Capacity.** `vcap` is read but never used. The assignment is uncapacitated.
5. **`C` and `T`.** These strategy fields are not used in any decision. In `output_stg.txt`, `C` equals the link's `M` for transit strategies.
6. **Buckets.** The bucket index is `min((cost / alg_parameter)², 10000)`. The paper describes intervals that depend on the square root of time. The effect of a different exponent was not tested. `alg_parameter` sets the order in which links are processed.
7. **File size.** `output_times.txt` and `output_stg.txt` grow with destinations × links.
8. **Precision.** Computations use 32-bit `float`.
9. **Boarding time.** By default, `temps` does not include `tboa`, like the paper's time `T`. The cost `cout` includes it, through `tboa × cboa`.

10. Validation

The program was compared with the numerical examples of the paper (weights at 1, `tboa = 0` unless stated).

Case	Paper	Program
Example 1: two lines, equal times	Shares 0.71 / 0.29, T = 24.285	0.714 / 0.286, T = 24.286
Example 2: different times	Shares 0.595 / 0.405, T = 22.56	0.595 / 0.405, T = 22.560
Example 3: two lines and a walking link	Shares 0.4 / 0.33 / 0.27	0.4 / 0.333 / 0.267
Emme network, fig. 12 (walk to E)	T = 24.46, flows 0.143 / 0.286 / 0.175 / 0.111 / 0.169 / 0.117 / 0.571	T = 24.467, same flows to 3 decimals
Emme network, fig. 14 (generalized time)	C = 34.84, T = 25.15, same flows to 3 decimals	C = 34.842, T = 25.152, same flows

For example 3, the program gives T = 16.73 and the paper 16.68. Computing by hand with the minimum times paired correctly gives 16.733.

Other checks, on synthetic networks: identical results with 1, 4 and 8 threads, no regression when no link type is defined, empty log on a matrix without errors. No validation was done on a large real network.

11. Change history compared with the version received

Parallelization. One destination per unit of work, one network copy per thread, ordered writing of results.

Corrections (they can change results):

1. Network columns 7 and 8 (`allowboai`, `allowalij`) were overwritten with 1 for every destination. They are now respected.
2. `id_bucket` was passed by value: a link falling into a bucket already passed was never processed. It is now passed by reference.
3. Restoring a link after an optimization with no gain had no effect. It now works.
4. In `OptimizeReduce`, the link time and the boarding cost were counted twice in the replacement test. The test now compares `s.c` with the existing strategy.
5. `pos` is reset to -1 for each destination, which makes the result independent of the processing order.

On a synthetic test set (2,112 links, 14,280 OD pairs), corrections 2 to 4 change the cost of 952 OD pairs, 940 of them downwards. The effect of correction 1 depends on whether the network file contains restrictions.

Corrections with no effect on the computation. Correct progress counter (integer division kept it at 0). Trailing empty lines ignored. Error messages with line numbers. Stop with a message if `cwait ⩽ 0`.

New features. OD pairs in error (empty line in `_od`, `_log` file). `cmap` and `cboa` weights by link type. `tboa_dans_temps` option. `nb_threads` argument. Comment lines and empty lines accepted in the parameter file.

Tried then removed. Adding `tboa` to the time by default: the paper publishes a time without boarding time. The `tboa_dans_temps` option adds it.

12. Complete example

Example 2 of the paper: two lines between a stop and the destination, with different times and headways.

`network.csv`

```
Z;S;C;0;0;100;1;1
S;D;L1;20;12;100;1;1
S;D;L2;15;30;100;1;1
```

`matrix.csv`

```
Z;D;1
```

`parameters.txt` (weights at 1, no boarding)

```
1
1
0
0
true
10
```

Command:

```
Mintc network.csv matrix.csv res parameters.txt
```

`res_od.txt`

```
o;d;temps;cout;ncorr
Z;D;22.55952;22.55952;0.9999999
```

`res_aff.txt` (excerpt)

```
num;i;j;ij;line;volau;boai;alij
1;S;D;S-D;L1;0.595238;...
2;S;D;S-D;L2;0.4047619;...
```

Line 1 receives 59.5% of the demand and line 2 40.5%. The faster line, despite its 30-minute headway, receives a larger share than a split by frequencies alone would give (28.6%).