JaamSim - A Guided Example

1. JaamSim

JaamSim is a discrete event simulation package. It is Java-based, freeware, open soource and, apparently, rather complete. Its webpage is at https://jaamsim.com/.

There is a manual and some other documentation available at the site and elsewhere, but as far as I have investigated it is not particularly friendly (sort of "you need to know in order to understand"). There is an active and helpful forum which can be reached from the webpage (https://groups.google.com/forum/#!forum/jaamsim-users).

2. About this Guide and the Author

This document is intended to be the guide, or the type of guide, that I'd had liked to have with me when creating my first "serious" model in JaamSim: a sequential step-by-step support, with comments and remarks on why or why-not about the features of JaamSim, so that it created familiarity with the program in a mentored way, rather than jumping straightforwardly into the User Manual.

In case it is needed to make it explicit: this document by no means intends to substitute the User Manual, and in fact it does not even cover many of the capabilities described in it – not to mention the Programming Manual!

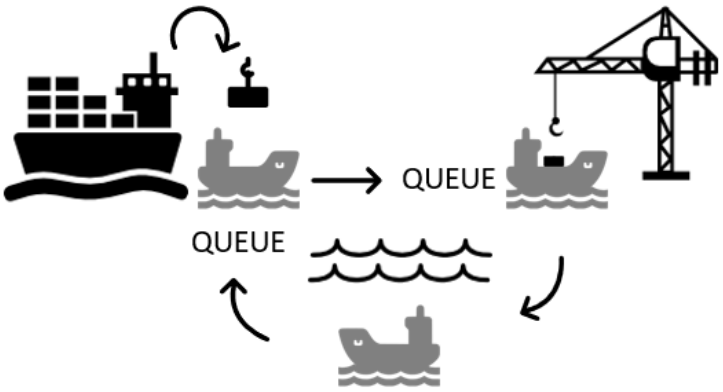
I am neither an expert in simulation nor in JaamSim. I have been interested in this field for many years and occasionally I have to create some logistics models for my job, but this activity is so sparse that I almost have to restart from scratch every time!

You can reach me at rigonz2@netscape.net.

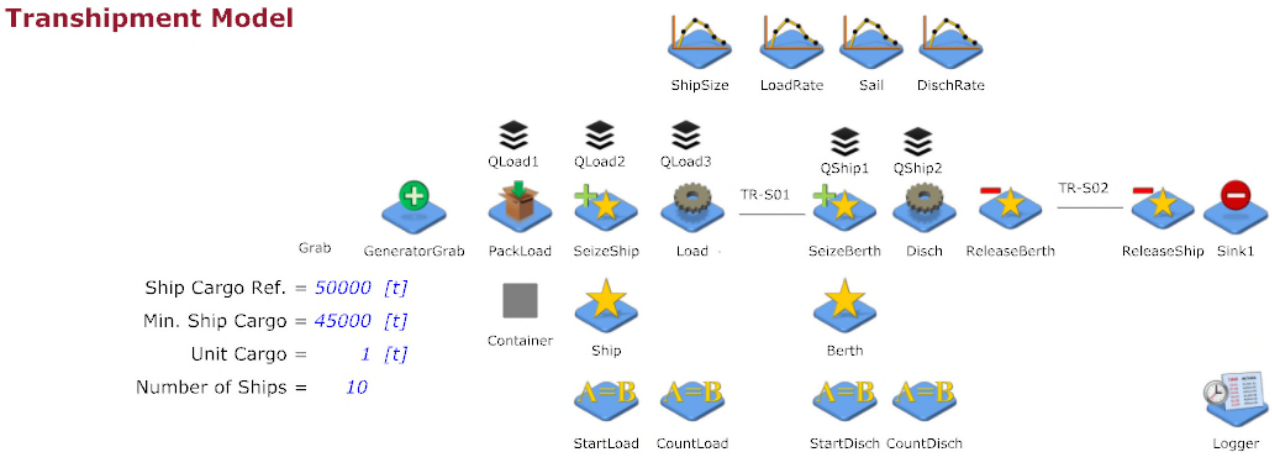
3. The Model

The model represents a realistic and relatively common and important case, a logistics, maritime transportation operation: the discharge of a mother vessel into lightering vessels or barges, which sail to a receiving port, discharge, and sail back to the mother vessel, in a round-trip. There are a number of those lightering vessels (the fleet), and the final purpose of the model is to investigate how much time does it take to unload the cargo under different conditions related to the mother vessel, the fleet, the receiving port, and the sailing conditions.

In a nutshell, this is a representation of the reality to be modelled:



The *equivalent* JaamSim model is:



The model can be downloaded from this address:

https://app.box.com/s/gvb55up4yd6tr3aalc84tk3dhbej5gii

4. First Steps

Here please go through chapters 1-4 of JaamSim User Manual in order to get a basic familiarity with the program: this guide does not explain most of the valuable information contained in those chapters.

5. The Entities

A discrete-event simulation essentially tracks entities that go through changes applied to them at given moments in time. Thus, entities are one of the founding blocks of JaamSim.

The entity used in this model is a 1 MT batch of cargo. It could be anything, from 1 kg o 1000 MT, but 1 MT provided more than enough granularity for my purposes, and the model can be easily modified for a different batch size. The name of the entity is *Grab* (because the model was intended for bulk solid cargo).

The weight of the *Grab* is defined through an InputValue (ModelBuilder/BasicObjects/InputValue).

Unit Cargo = 1 [t]

The entity has to be identifiable, but this identification can be just conceptual. In this case the cargo is bulk, so there is no physical separation among the entities that I have described, all the cargo is mixed: I just define each 1 MT that crosses the side of the mother vessel as an entity called *Grab*.

The default image for an entity is a sphere, but this can be modified at the Input Editor of the entity (Graphics tab/DisplayModel). It is possible to add new images, but this is not clearly explained in the Manual. The new image has to be in the set of ImageModel (ObjectSelector/DisplayModels/ImageModel). There is no folder that I have seen where the default images are stored, so it is necessary to duplicate one of the existing images at ImageModel (select & right click), rename it (F2) and then link it to the desired image (InputEditor/KeyInputs/ImageFile). Only after this is done, the new image will be available for the entities.

In this case, the image used for the *Grab* entity is:

Input Editor - Grab

Key Inputs | Graphics

Keyword	Default	Value
Position	0.0 0.0 0.0 m	-2.900000 0.200000 0.000000 m
Alignment	0.0 0.0 0.0	0.0 0.0 -0.5
Size	0.5 0.5 0.5 m	
Orientation	0.0 0.0 0.0 ...	
Region	*None*	
RelativeEntity	*None*	
DisplayModel	Sphere	GrabModel
Show	TRUE	TRUE
Movable	TRUE	TRUE
VisibleViews	*All Views*	
DrawRange	0.0 Infinity m	

6. The Generation of Entities

The entities to be used in the model require an explicit generation. In this model all entities are created at once (InterArrivalTime = 0), as the mother vessel reaches the transhipment zone loaded with them.

How many entities are needed? Well, the cargo of the vessel divided by the weight of the entity. The cargo of the mother vessel is defined as an InputValue (ModelBuilder/BasicObjects/InputValue). The assigned figure is 50,000 MT.

Ship Cargo Ref. = *50000 [t]*

The part in black is a simple text (ModelBuilder/GraphicObjects/Text), and the part in blue is the InputValue (whose name is *ShipCargoRef*).

From here the number of entities to be generated is straightforward: [*ShipCargoRef*].Value/[UnitCargo].value, with a ceiling just to be of the upper side:

Input Editor - GeneratorGrab

Key Inputs | Thresholds | Maintenance | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
StateGraphics	*None*	
NextComponent	*None*	QLoad1
FirstArrivalTime	0.0 s	
InterArrivalTime	1.0 s	0 s
EntitiesPerArrival	1.0	
PrototypeEntity	*None*	Grab
BaseName	*Generator*	
MaxNumber	*Infinity*	ceil([ShipCargoRef].Value/[UnitCargo].Value)

The “instantaneous” generation of these entities before the simulation starts places them in a queue (*QLoad1*).

7. The Fleet of Lighter Vessels

These ships are treated as a resource (ModelBuilder/ProcessFlow/Resource). The name of the resource is *Ship*. The number of such resources is defined as a parameter through an InputValue whose name is *NumberShip*.

Number of Ships = *10*

Input Editor - Ship

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
Capacity	1.0	NumberShip
StrictOrder	FALSE	

There is a fundamental issue with this approach: while a resource is treated as an undifferentiated entity, not all lighter ships are equal, as they may differ in cargo capacity, sailing conditions, mechanical features, etc.

At least the following alternatives on this regard can be used, with an increasing degree of detail:

- Use resources, with average values.
- Use averages and introduce the differences at the servers.
- Use entities and introduce the differences via the AttributeDeffinitionList.

The second is the selected option in this model.

The capacity of each lighter vessel is defined as follows:

- A continuous probability distribution (ModelBuilder/ProbabilityDistributions/Continuous) is used (its name is *ShipSize*). The values of this distribution are based on real observations and correspond to actual cargo capacities, in tons

  Input Editor - ShipSize

  Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
UnitType	*None*	MassUnit
RandomSeed	*None*	1
MinValue	-Infinity kg	
MaxValue	Infinity kg	
ValueList	*None*	400 600 700 900 1000 1100 1200 1300 1600 1900 2600 [t]
CumulativeProbability...	*None*	0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

  The cargo of each lighter vessel (as defined by *ShipSize*) is assigned to a container entity (ModelBuilder/ProcessFlow/EntityContainer) whose name is … *Container*.
  The packing activity (ModelBuilder/ProcessFlow/Pack) is conducted at the activity named *PackLoad*. This activity bundles together the entities queuing at the WaitQueue (which are *Grabs*). The number of *Grabs* to be put into the *Container* is specified by a call to *ShipSize*, the probability distribution.
  A small ServiceTime is provided (0.01 s) as an instantaneous operation seems to yield erroneous results.
  Containers are not generated explicitly as the standard entities are: they are created at the packing activity. So, in this case, the containers are created at the beginning of the simulation, while t=0. *Containers* are generated in as much as there are elements (*Grabs*) in the incoming queue, which is defined by the size of the mother vessel (*ShipCargoRef*). It is possible, but unlikely, that the initial number of *Grabs* will exactly match the cumulative requirements of the *Containers*; however, the most common case will be that at the end of the packing activity the queue of *Grabs* has still few of them whose number is less than the capacity of the Container to be packed. In this case the packing cannot be completed for all the *Grabs*, and these few units will remain “undischarged” from the mother vessel. While this is not a perfect solution, it is of no relevance in this particular case because mother vessels are often chartered with a ±10% range for their cargo capacity.

Input Editor - PackLoad

Key Inputs | Thresholds | Maintenance | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
StateGraphics	*None*	
DefaultEntity	*None*	
NextComponent	*None*	QLoad2
StateAssignment	*None*	
ProcessPosition	0.0 0.0 0.01...	
WaitQueue	*None*	QLoad1
Match	*None*	
PrototypeEntityContai...	*None*	Container
NumberOfEntities	1.0	ceil([ShipSize].Value/[UnitCargo].Value)
NumberToStart	*NumberOfEntiti*	
ServiceTime	0.0 s	0.01 s

8. The Discharge of the Mother Vessel

The discharge of the mother vessel can proceed under the following conditions:

- - there is product onboard, and
- - there is a lighter vessel alongside.

With regard to the lighter vessels: they stay in a queue until there is a free slot alongside the mother vessel.

The model assumes that there is only one discharge position at the mother vessel, and that there are no other interferences in the discharge. Reality is that bad weather affects the operation, and that sometimes two lighters can be loaded simultaneously. Both circumstances can be included in the model with relative easiness with, for instance, a resource for the weather, and a second loading server for the two-slots discharge operation.

The sequence in the model is as follows:

- *Containers* are queued after being packed (*QLoad2*),
- A seize task (ModelBuilder/ProcessFlow/Seize) joins the *Container* and the *Ship* (a resource defined above) and places them in another queue (*QLoad3*). The seize operation consumes not time, but if there is no *Ship* or *Container*, the seize is not performed. This seize activity is named *SeizeShip*.

Input Editor - SeizeShip

Key Inputs | Thresholds | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
StateGraphics	*None*	
DefaultEntity	*None*	
NextComponent	*None*	StartLoad
StateAssignment	*None*	
WaitQueue	*None*	QLoad2
Match	*None*	
ResourceList	*None*	Ship
NumberOfUnits	{ 1.0 }	

- The "loading" of the lightering vessels or *Ships* takes place in the server (ModelBuilder/ProcessFlow/Server) named *Load*. The *Container* goes through this server and is released after a specified amount of time. The resource *Ship* does not "go" through this server, it has been seized before, but we may imagine as if the *Container* and the resource *Ship* go together.

Input Editor - Load

Key Inputs | Thresholds | Maintenance | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
StateGraphics	*None*	
DefaultEntity	*None*	
NextComponent	*None*	CountLoad
StateAssignment	*None*	
ProcessPosition	0.0 0.0 0.01...	
WaitQueue	*None*	QLoad3
Match	*None*	
ServiceTime	0.0 s	[Load].obj.Count*[UnitCargo].Value/[LoadRate].Value

The loading time (ServiceTime) is defined as the cargo in the container divided by a loading rate:

- The cargo can be known with [*Load*].obj.Count, which provides the number of *Grabs* in the *Container* which is in the server *Load*. (The weight of each *Grab* is specified at *UnitCargo*).
- The material transfer rate from the mother vessel to the lighter vessel is specified from a continuous probability distribution (ModelBuilder/ProbabilityDistributions/ ContinuousDistribution) whose name is *LoadRate* and is obtained from real data (its units are t/d).

Input Editor - LoadRate

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
UnitType	*None*	MassFlowUnit
RandomSeed	*None*	3
MinValue	-Infinity kg/s	
MaxValue	Infinity kg/s	
ValueList	*None*	1260 1630 2210 2620 3010 3370 3690 4100 4680 5280 5900 [t/d]
CumulativeProbability...	*None*	0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

Discharge rates vary along the discharge process: they tend to be high at the beginning of the operation, when holds are full, and tend to decrease when the holds are emptier. This important feature is not included in the model, which relies on averages on this regard.

Once the load is completed, the *Ship* departs the mother vessel.

9. Sailing from Mother Vessel to Port

Sailing is simulated with an EntityDelay (ModelBuilder/ProcessFlow/EntityDelay) named *TR-S01*. The sailing time is defined from a continuous probability distribution named *Sail*, obtained from real data.

Input Editor - TR-S01

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
StateGraphics	*None*	
DefaultEntity	*None*	
NextComponent	*None*	QShip1
StateAssignment	*None*	
Duration	*None*	Sail
Animation	TRUE	
Width	1.0	
Color	black	

Input Editor - Sail

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
UnitType	*None*	TimeUnit
RandomSeed	*None*	3
MinValue	-Infinity s	
MaxValue	Infinity s	
ValueList	*None*	0.90 1.61 1.94 2.24 2.31 2.37 2.42 2.47 2.56 2.73 5.71 [d]
CumulativeProbabilityList	*None*	0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

10. Arrival to Port

Upon “arrival to port” (actually, upon completion of the sailing activity through *TR-S01*), the *Ship* enters into a queue (*QShip1*).

A resource is defined for the port, called *Berth*. In the model it is assumed that only one *Berth* is available, and the *Ships* need to wait until it is released. If there are *Ships* in the queue and the *Berth* is available, then it is seized.

Input Editor - SeizeBerth

Key Inputs | Thresholds | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
StateGraphics	*None*	
DefaultEntity	*None*	
NextComponent	*None*	StartDisch
StateAssignment	*None*	
WaitQueue	*None*	QShip1
Match	*None*	
ResourceList	*None*	Berth
NumberOfUnits	{ 1.0 }	

Input Editor - Berth

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
Capacity	1.0	1
StrictOrder	FALSE	

11. Ship Discharge at the Port

The discharge operation happens at a server called *Disch*. The discharge time is calculated as the quotient of a cargo quantity and a discharge rate.

The cargo quantity is determined from [*Disch*].obj.Count, the actual number of *Grabs* brought by the *Ship* in its *Container*.

The discharge rate is calculated from a continuous probability distribution, also based on actual data, named *DischRate*. Its units are also t/d.

Input Editor - Disch

Key Inputs | Thresholds | Maintenance | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
StateGraphics	*None*	
DefaultEntity	*None*	
NextComponent	*None*	CountDisch
StateAssignment	*None*	
ProcessPosition	0.0 0.0 0.01...	
WaitQueue	*None*	QShip2
Match	*None*	
ServiceTime	0.0 s	[Disch].obj.Count*[UnitCargo].Value/[DischRate].Value

Input Editor - DischRate

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
UnitType	*None*	MassFlowUnit
RandomSeed	*None*	3
MinValue	-Infinity kg/s	
MaxValue	Infinity kg/s	
ValueList	*None*	620 1690 2290 2610 2980 3320 3740 4350 4930 5730 9080 [t/d]
CumulativeProbability...	*None*	0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

The completion of the discharge automatically releases the *Berth* resource. The *Container* is “left onboard” the Ship until it is released.

Input Editor - ReleaseBerth

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
DefaultEntity	*None*	
NextComponent	*None*	TR-S02
StateAssignment	*None*	
ResourceList	*None*	Berth
NumberOfUnits	{ 1.0 }	

12. Sailing back to Mother Vessel from Port

The discharged *Ship* sails back to the mother vessel through another EntityDelay (*TR-S02*) whose duration is controlled by the same *Sail* distribution. It would be possible to differentiate between sailing loaded and unloaded, but the difference was not considered relevant.

Input Editor - TR-S02

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
StateGraphics	*None*	
DefaultEntity	*None*	
NextComponent	*None*	ReleaseShip
StateAssignment	*None*	
Duration	*None*	Sail
Animation	TRUE	
Width	1.0	
Color	black	

Once the *Ship* completes its sailing, it is released. There is no "queue" for the Ships at the mother vessel, they are just in the resource pool. After this, the surviving entity (the *Container*) is sunk.

Input Editor - ReleaseShip

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
DefaultEntity	*None*	
NextComponent	*None*	Sink1
StateAssignment	*None*	
ResourceList	*None*	Ship
NumberOfUnits	{ 1.0 }	

13. Material Handling – Counting

The control of the material handled is done through two Assign activities (ModelBuilder/ProcessFlow/Assign): *CountLoad* and *CountDisch*. The Assign activity is necessary because "The Assign object is the only place where an Attribute's value can be modified".

CountLoad occurs after the *Load* server, while *CountDisch* is placed after the *Disch* server.

Both assigns use an attribute of the corresponding activity (AttributeDeffinitionList), named *N*, to add the number of *Grabs* that are served with the expression: this.*N* = this.*N* + this.obj.Count. The attribute *N* is initialized to 0, and later the operator .Count accesses the number of *Grabs* in the *Container* every time that a *Ship* is loaded or unloaded.

There is a time delay which may be occasionally important: these two attributes/variables only count the material that is fully processed: the amount of product that is in the Ship before it is fully loaded or fully empty is not counted by the attribute *N*. If this degree of detail is required, then it is necessary to enter at the *Pack* and *UnPack* activities. On this regard it is worth noting that NumberAdded refers to the individual items handled (*Grabs*), while NumberProcessed counts the number of *Containers*.

Input Editor - CountLoad

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	{ N 0 [t] }
CustomOutputList	*None*	
DefaultEntity	*None*	
NextComponent	*None*	TR-S01
StateAssignment	*None*	
AttributeAssignmentList	*None*	{ 'this.N = this.N + this.obj.Count*[UnitCargo].Value' }

Input Editor - CountDisch

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	{ N 0 [t] }
CustomOutputList	*None*	{ T '[Disch].NumberAdded == 0? [Disch].SimTime/1[d]:0' }
DefaultEntity	*None*	
NextComponent	*None*	ReleaseBerth
StateAssignment	*None*	
AttributeAssignmentList	*None*	{ 'this.N = this.N + this.obj.Count*[UnitCargo].Value' }

Data

JaamSim allows to provide output through the GUI in different formats, and three are used in this model:

- Text (ModelBuilder/GraphicsObjects/Text),
- Bar Gauge (ModelBuilder/GraphicsObjects/BarGauge), and
- Graph (ModelBuilder/GraphicsObjects/Graph).

13.1. Text

Textual information is provided for the SimulationTime, the quantities loaded, discharged and floating, the number of *Ships* at each position (loading, discharging, sailing back/forth, or queuing at mother vessel or port), plus the times at which certain events happened. Different operators are used to extract the required information:

Qtty Loaded, MT = 49,704
Qtty Discharged, MT = 49,704
Qtty Floating, MT = 0
Ships Loaded = 39
Ships Discharged = 39

Ships Load = 0
Ships Disch = 0
Ships Sail 1 = 0
Ships Sail 2 = 0
Ships Queue Load = 10
Ships Queue Disch = 0

First Load Start, d = 0.0
Last Load End, d = 24.1
First Disch Start, d = 2.5
Last Disch End, d = 27.6

Input Editor - Text52

Key Inputs | Font | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
TextHeight	0.3 m	0.2 m
Format	*None*	'Qtty Floating, *100 MT = %,.0f'
UnitType	DimensionlessUnit	DimensionlessUnit
Unit	*None*	
DataSource	*None*	[CountLoad].N-[CountDisch].N
FailText	Input Error	

Input Editor - Text53

Key Inputs | Font | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
TextHeight	0.3 m	0.2 m
Format	*None*	'Ships Loaded = %,.0f'
UnitType	DimensionlessUnit	DimensionlessUnit
Unit	*None*	
DataSource	*None*	[Load].NumberProcessed
FailText	Input Error	

Input Editor - Text57

Key Inputs | Font | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
TextHeight	0.3 m	0.2 m
Format	*None*	'Ships Sail 1 = %,.0f'
UnitType	DimensionlessUnit	DimensionlessUnit
Unit	*None*	
DataSource	*None*	[TR-S01].NumberInProgress
FailText	Input Error	

Input Editor - Text59

Key Inputs | Font | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
TextHeight	0.3 m	0.2 m
Format	*None*	'Ships Queue Load = %,.0f'
UnitType	DimensionlessUnit	DimensionlessUnit
Unit	*None*	
DataSource	*None*	[Ship].AvailableUnits
FailText	Input Error	

Input Editor - Text60

Key Inputs | Font | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
TextHeight	0.3 m	0.2 m
Format	*None*	'Ships Queue Disch = %,.0f'
UnitType	DimensionlessUnit	DimensionlessUnit
Unit	*None*	
DataSource	*None*	[QShip1].QueueLength
FailText	Input Error	

Input Editor - Text34

Key Inputs | Font | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
TextHeight	0.3 m	0.2 m
Format	*None*	'First Load Start, d = %,.1f'
UnitType	DimensionlessUnit	TimeUnit
Unit	*None*	d
DataSource	*None*	[StartLoad].T1
FailText	Input Error	

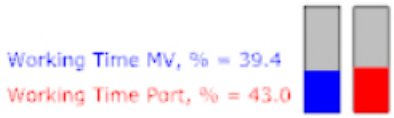
Input Editor - Text33

Key Inputs | Font | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
TextHeight	0.3 m	0.2 m
Format	*None*	'Last Disch End, d = %,.1f'
UnitType	DimensionlessUnit	TimeUnit
Unit	*None*	d
DataSource	*None*	[Disch].ReleaseTime
FailText	Input Error	

13.2. Bar Gauges

The bars gauges provide a visual representation of a level-type variable, in this case of the actual working time at the mother vessel and the receiving port:



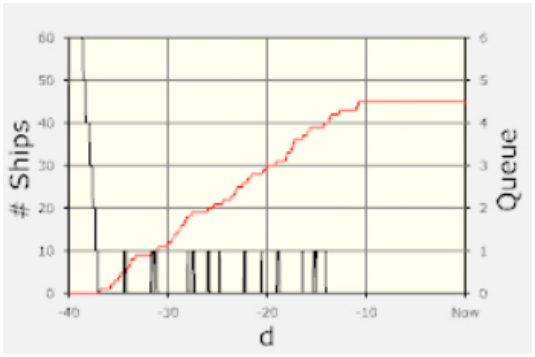
Input Editor - BarGauge2

Key Inputs | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
DataSource	0.5	[Disch].WorkingTime/[Disch].SimTime
Colour	blue	255 0 0
BackgroundColour	gray75	

13.3. Charts

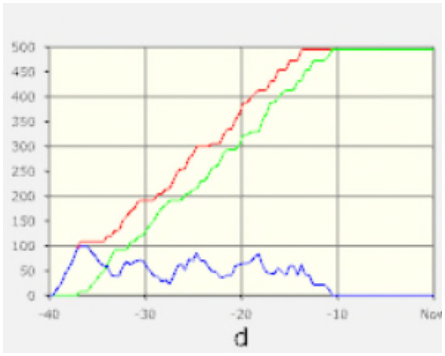
The charts in JaamSim allow to represent the time evolution of selected variables. In this model three charts are used:



Input Editor - Graph1

Key Inputs | X-Axis | Y-Axis | Secondary Y-Axis | Graphics

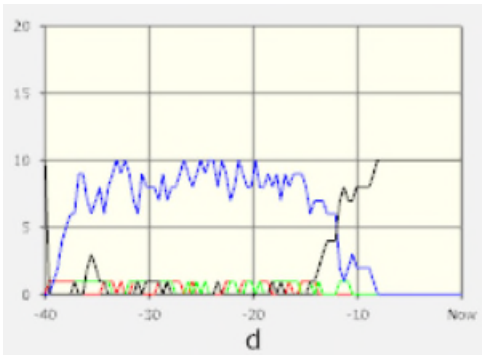
Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
Title	Graph Title	''
NumberOfPoints	100	1000
UnitType	DimensionlessUnit	
DataSource	*None*	{ [Disch].NumberProcessed }
LineColours	{ red }	
LineWidths	1.0	
SecondaryUnitType	DimensionlessUnit	
SecondaryDataSource	*None*	{ [QLoad3].QueueLength }
SecondaryLineColours	{ black }	
SecondaryLineWidths	1.0	



Input Editor - Graph2

Key Inputs | X-Axis | Y-Axis | Secondary Y-Axis | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
Title	Graph Title	''
NumberOfPoints	100	
UnitType	DimensionlessUnit	
DataSource	*None*	{ [CountLoad].N } { [CountDisch].N } { [CountLoad].N-[CountDisch].N }
LineColours	{ red }	{ 255 0 0 } { 0 255 0 } { 0 0 255 }
LineWidths	1.0	
SecondaryUnitType	DimensionlessUnit	
SecondaryDataSource	*None*	
SecondaryLineColours	{ black }	
SecondaryLineWidths	1.0	



Input Editor - Graph3

Key Inputs | X-Axis | Y-Axis | Secondary Y-Axis | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
Title	Graph Title	''
NumberOfPoints	100	
UnitType	DimensionlessUnit	
DataSource	*None*	{ [Load].NumberInProgress } { [Disch].NumberInProgress } { '[TR-S01].NumberInProgress + [TR-S02].NumberInProgress' } { ' [Ship].AvailableUnits +[QShip1].QueueLength' }
LineColours	{ red }	{ 255 0 0 } { 0 255 0 } { 0 0 255 } { 0 0 0 }
LineWidths	1.0	
SecondaryUnitType	DimensionlessUnit	
SecondaryDataSource	*None*	
SecondaryLineColours	{ black }	
SecondaryLineWidths	1.0	

13.4. Logger

A logging activity (ModelBuilder/BasicObjects/ExpressionLogger) named *Logger* records the number of *Ships* handled at 1 h intervals. If the interval is not specified, the data is recorded every time there

is a change at DataSource. (I recommend that you check the output in this case, because sometimes the time lapse is too large, and several activities may be “lost”).

Input Editor - Logger

Key Inputs | Tracing | Graphics

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
UnitTypeList	DimensionlessUnit	
DataSource	*None*	{ [Disch].NumberProcessed } { [Load].NumberProcessed }
IncludeInitialization	TRUE	
StartTime	0.0 s	
EndTime	Infinity s	
Interval	*None*	1 [h]

```
494020.975055   Idle    Idle    5.0     12.0
513204.49289    Working Idle    5.0     12.0
560194.113276   Working Working 6.0     12.0
649112.508882   Idle    Working 10.0    14.0
687640.230533   Idle    Idle    10.0    15.0
723198.659861   Working Idle    10.0    15.0
745335.606559   Working Working 11.0    15.0
1023341.279611  Working Idle    14.0    21.0
```

The output is in a file named “<configuration file name>-<EntityLogger name>.log”, in ASCII format wit tab separations. Log files are overwritten when the simulation starts, so you may need to rename.

13.5. Multiple Runs and Control of the Simulation

This is controlled from the simulation entity (ObjectSelector/Simulation). It is possible to set a maximum simulation time, a condition for pause, or multiple runs.

Input Editor - Simulation

Key Inputs | Multiple Runs | GUI

Keyword	Default	Value
AttributeDefinitionList	*None*	
CustomOutputList	*None*	
RunDuration	3.1536E7 s	40 d
InitializationDuration	0.0 s	
PauseCondition	*None*	([CountDisch].N>[ShipCargo].Value)&&([Ship].AvailableUnits==[NumberShip].Value)
ExitAtPauseCondition	FALSE	TRUE
ExitAtStop	FALSE	
GlobalSubstreamSeed	0.0	[Simulation].RunIndex(1)
PrintReport	FALSE	TRUE
ReportDirectory	*Configuration File*	
UnitTypeList	DimensionlessUnit	
RunOutputList	*None*	
TickLength	1.0E-6 s	

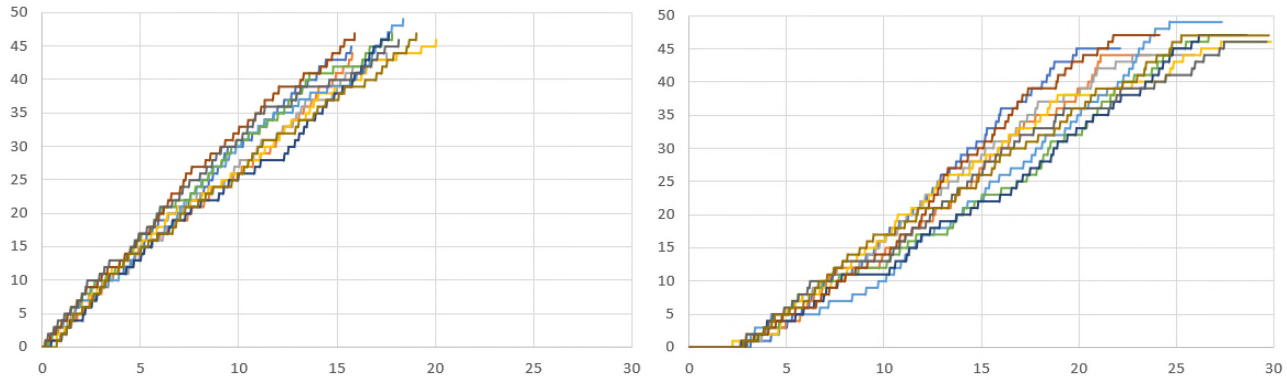
Input Editor - Simulation

Key Inputs | Multiple Runs | GUI

Keyword	Default	Value
RunIndexDefinitionList	1	10
StartingRunNumber	1	
EndingRunNumber	1	10

14. Typical Results

The following charts show some of the results that can be obtained from this model.

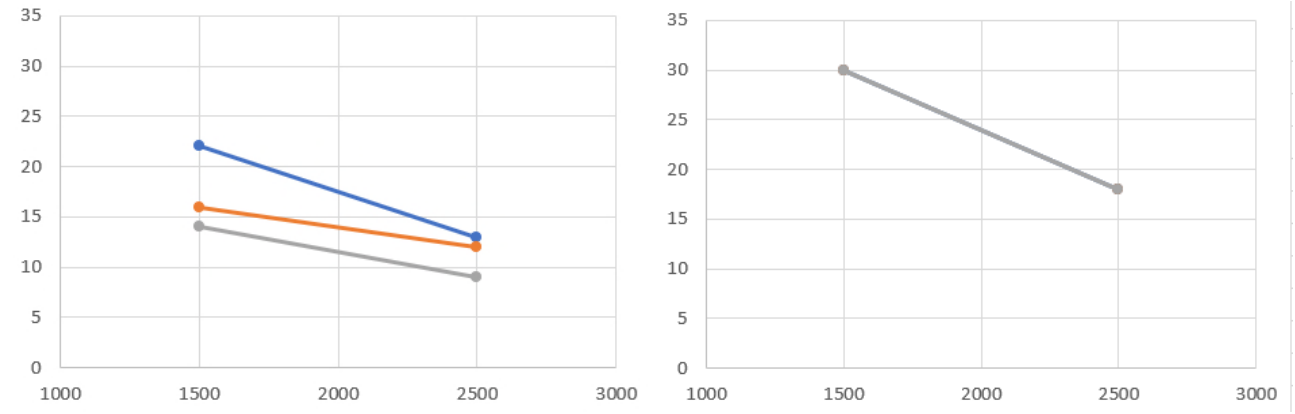


Progress of Operations, 10 runs

- Left: load at mother vessel; right: discharge at port
- Horizontal axis: time, d; Vertical axis: cargo, kt

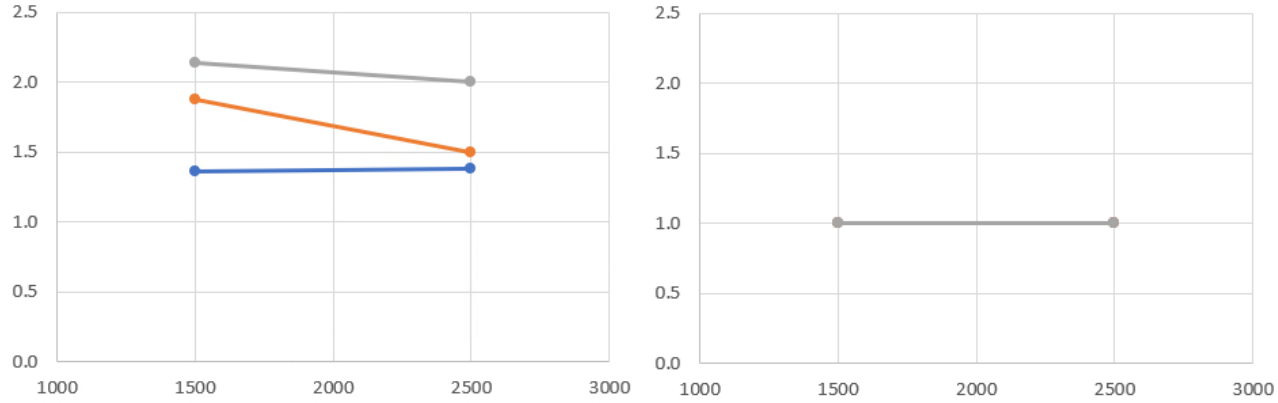
Progress of Operations, 10 runs

- Left: completion time, d; right: utilization rate, %
- Blue: port; orange: mother vessel
- Horizontal axis: id of runs



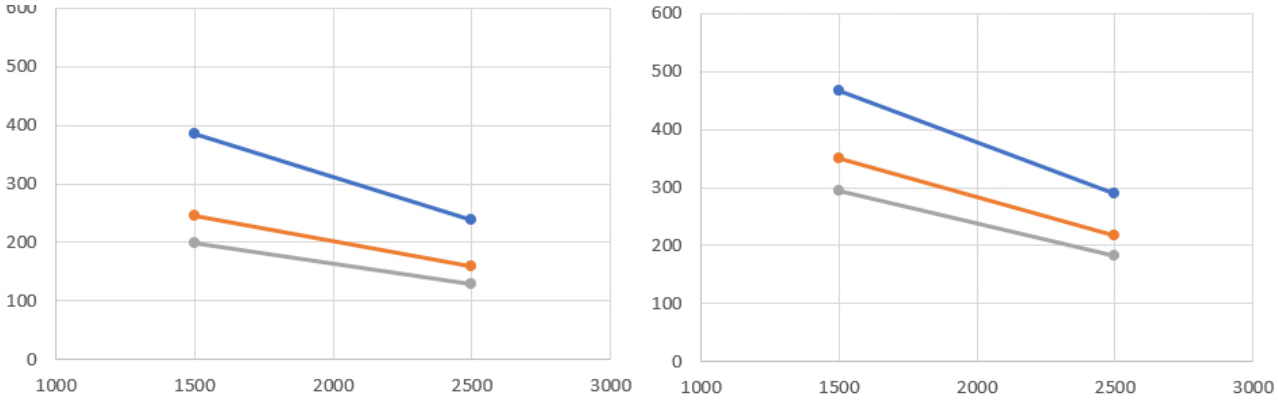
Estimated Fleet Size, #

- Left: effective discharge rate at mother vessel: 3,000 t/d; right: 5,000 t/d.
- Receiving capacity at port: blue: 2,000 t/d, brown: 3,000 t/d, grey: 4,000 t/d.
- Horizontal axis: average capacity of lighter, t.



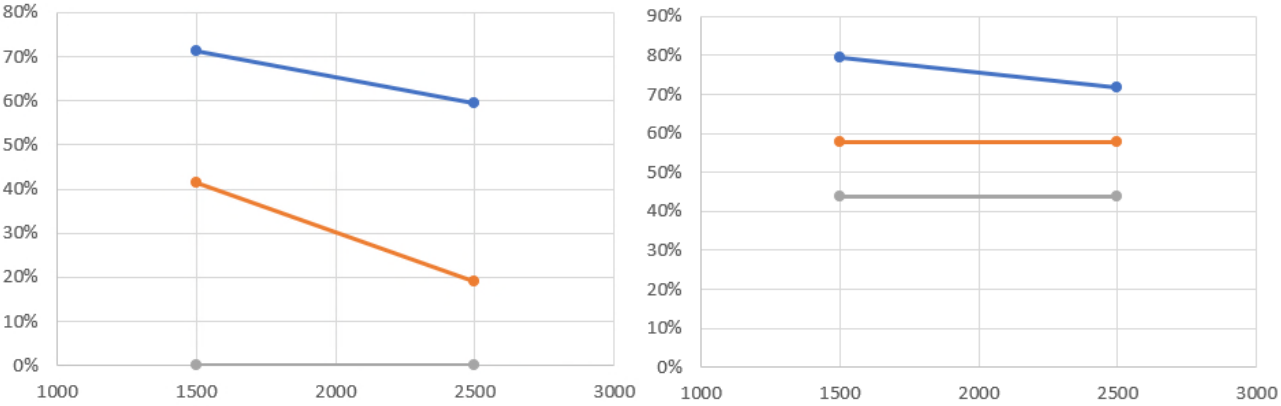
Average Number of Cycles for the Lighters, #

- Left: effective discharge rate at mother vessel 3,000 t/d; right: 5,000 t/d.
- Receiving capacity at port: blue: 2,000 t/d, brown: 3,000 t/d, grey: 4,000 t/d.
- Horizontal axis: average capacity of lighter, t.



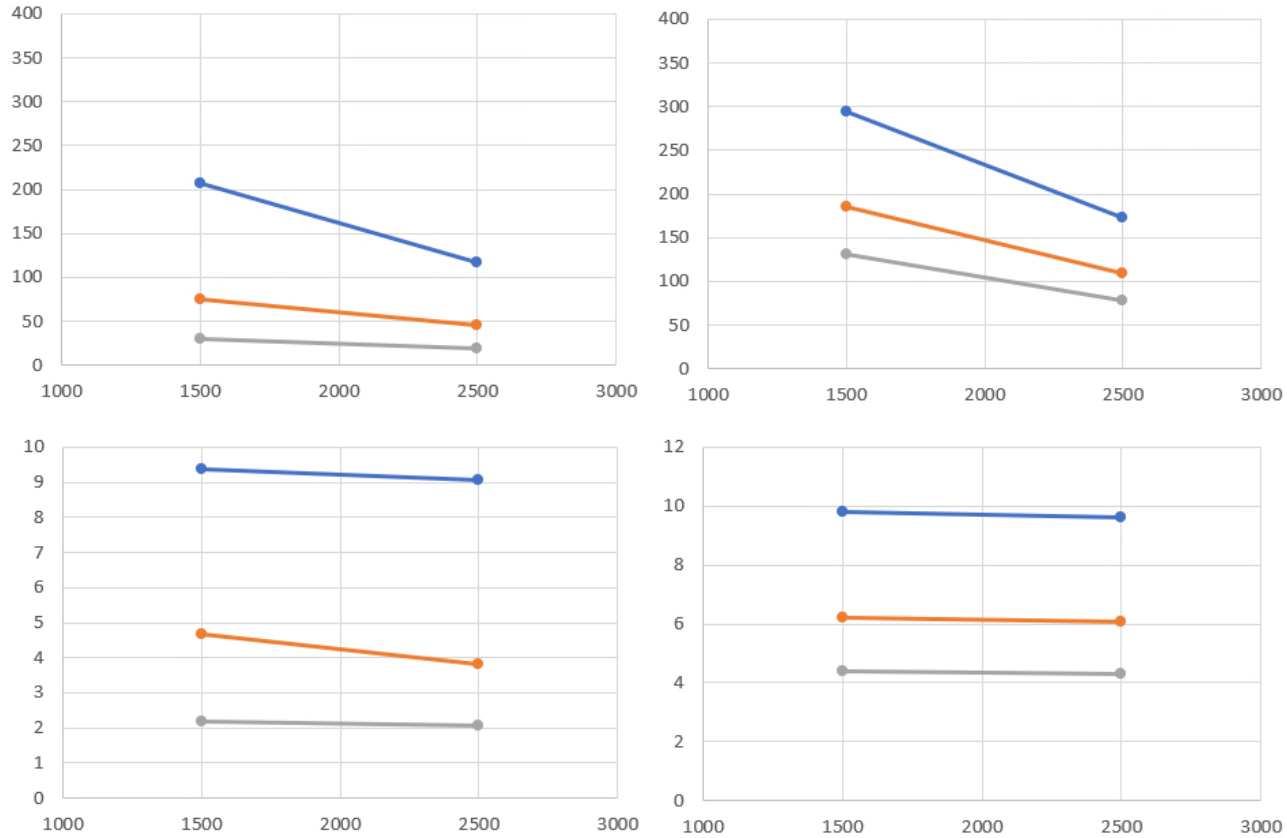
Total Utilization of Lighters, d

- Left: effective discharge rate at mother vessel 3,000 t/d; right: 5,000 t/d.
- Receiving capacity at port: blue: 2,000 t/d, brown: 3,000 t/d, grey: 4,000 t/d.
- Horizontal axis: average capacity of lighter, t.



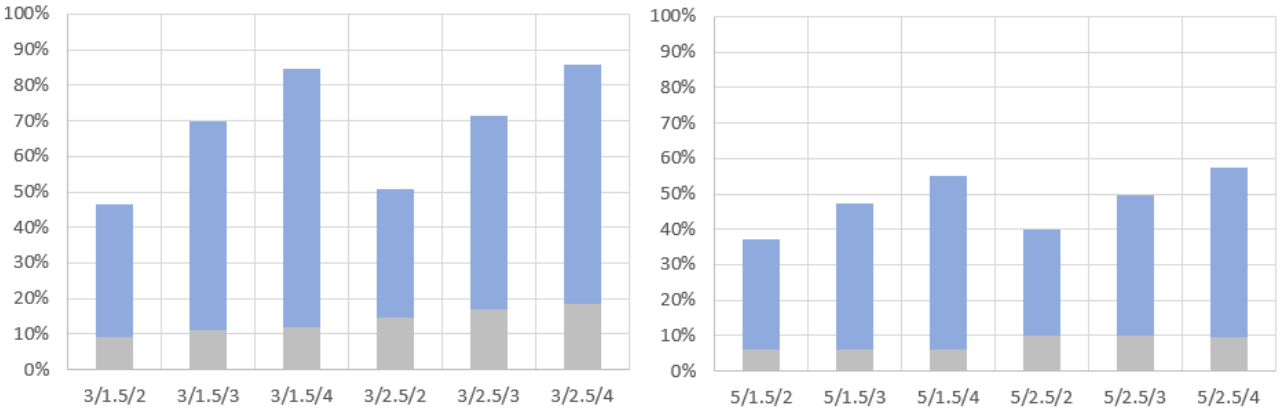
Waiting Time Spent at port, % of total wait time

- Left: effective discharge rate at mother vessel 3,000 t/d; right: 5,000 t/d.
- Receiving capacity at port: Blue: 2,000 t/d, brown: 3,000 t/d, grey: 4,000 t/d.
- Horizontal axis: average capacity of lighter, t.



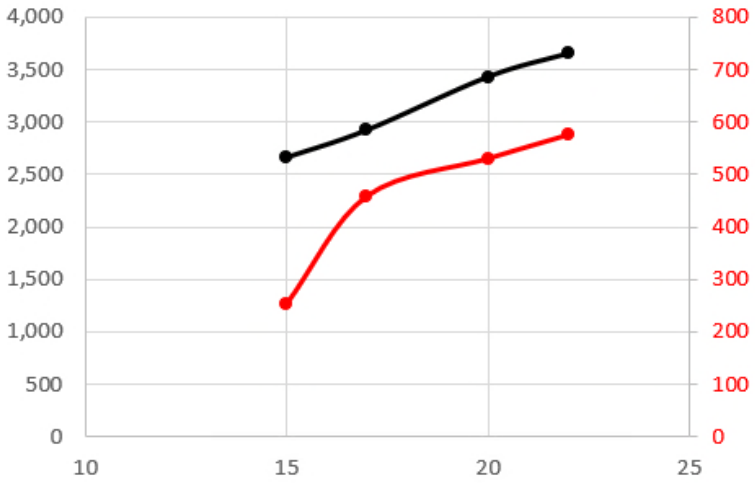
Waiting Time of Lighters, d

- Above: total; below: average
- Left: effective discharge rate at mother vessel 3,000 t/d; right: 5,000 t/d.
- Receiving capacity at port: blue: 2,000 t/d, brown: 3,000 t/d, grey: 4,000 t/d.
- Horizontal axis: average capacity of lighter, t.



Total Distribution of Time, %

- Left: effective discharge rate at mother vessel 3,000 t/d; right: 5,000 t/d.
- Columns: grey: loading and unloading; blue: sailing.
- Horizontal axis code: N1/N2/N3: N1: discharge rate at mother vessel, kt/d; N2: average capacity of lighter, kt; N3: receiving capacity at port, kt/d.



Effect of Fleet Size

- Effective discharge rate at mother vessel 3,000 t/d; lighter size: 1,500 t; discharge rate at port: 2,000 t/d.
- Black, left axis: gross unloading rate at mother vessel, t/d; red, right axis: utilization of lighter vessels, d.
- Horizontal axis: fleet size.