Discrete Event Simulation (DEVS) of Strategic Airlift

SYSC 5104 – Fall 2019

Prepared By:

Rob Barwell 101092722

**Contents**

[Part 1 – Conceptual Model – Strategic Airlift 3](#_Toc22030564)

[Background 3](#_Toc22030565)

[Model Overview 3](#_Toc22030566)

[Model Assumptions / Constraints 3](#_Toc22030567)

[Model Components 3](#_Toc22030568)

[Model Diagram 4](#_Toc22030569)

[Part 2 – Formal Model Specifications 5](#_Toc22030570)

[Notes 5](#_Toc22030571)

[Atomic Models 5](#_Toc22030572)

[Pallet Generator = <X, Y, S, ta, δext, δint, λ> 5](#_Toc22030573)

[Packing Facility = <X, Y, S, ta, δext, δint, λ> 7](#_Toc22030574)

[Aircraft Loader = <X, Y, S, ta, δext, δint, λ> 9](#_Toc22030575)

[Aircraft = <X, Y, S, ta, δext, δint, λ> 11](#_Toc22030576)

[Destination = <X, Y, S, ta, δext, δint, λ> 13](#_Toc22030577)

[Coupled Models 14](#_Toc22030578)

[Loading Process = <*X*, *Y*, D, {Mi}, IC, EIC, EOC, select> 14](#_Toc22030579)

[Strategic Airlift = <*X*, *Y*, D, {Mi}, IC, EIC, EOC, select> 14](#_Toc22030580)

[Part 3 – Model Experiments 15](#_Toc22030581)

[Notes 15](#_Toc22030582)

[Atomic Experiments 15](#_Toc22030583)

[Pallet Generator (PalletGen\_Test.exe) 15](#_Toc22030584)

[Packing Facility (LoadPacking\_Test.exe) 17](#_Toc22030585)

[Aircraft Loader (AircraftLoader\_Test.exe) 18](#_Toc22030586)

[Aircraft (Aircraft\_Test.exe) 19](#_Toc22030587)

[Destination (Destination\_Test.exe) 20](#_Toc22030588)

[Coupled Experiments 21](#_Toc22030589)

[Loading Process (LoadingProcess\_Test.exe) 21](#_Toc22030590)

[Model Experiments 22](#_Toc22030591)

[Overview 22](#_Toc22030592)

[Experiments 24](#_Toc22030593)

[E1 – Expired Pallet Generation Experiment 24](#_Toc22030594)

[E2 – Pallet Expiration Time Experiment 25](#_Toc22030595)

# Part 1 – Conceptual Model – Strategic Airlift

## Background

Page 81 of Canada’s current Defence Policy, Strong, Secure, Engaged specifies the Government of Canada’s desire to conduct military operations domestically and abroad. Questions have been raised about whether Canada has enough strategic airlift capability to deliver on those promises. This model will attempt to start providing answers to these questions.

**Definition: Strategic Airlift** – The rapid transport of a large number of passengers and/or over-sized heavy cargo over long distances within Canada or between Canada and a theatre of operations.[[1]](#footnote-1)

## Model Overview

This model starts with someone providing a pallet of cargo (people or goods) for shipping at a source location. Each pallet has an expiration time associated with it. This expiration time can also be used to denote priority. The pallet will then be sent to a queue for loading on an aircraft. The aircraft is used to transport pallets between locations. The only constraint on the location is the ability to receive, process, and store the pallet. The aircraft also has further constraints regarding the amount of cargo it can carry and its availability due to maintenance and crew availability.

## Model Assumptions / Constraints

The following assumptions and constraints are placed on the model to keep it contained for assignment 1:

1. Pallets each have a size of 1 and can be shipped with any other pallet;
2. Pallets only have one source location in Canada;
3. The aircraft only flies directly between Canada and another location. It does not have intermediate stops and does not fly between other locations with cargo;
4. Pallets can be delivered to any destination within 30 hours. Therefore, flight time between 3 and 30 hours will be used to approximate the distance between Canada and foreign locations. This reduces the complexity required to calculate actual time between geographical points.
5. Only one type of aircraft is available to represent a C130J and transport 4 pallets of goods each time. Only 2 aircraft will be used in this model.
6. The destination will only do basic processing in this iteration of the model. Additional complexities can be added in future versions.

## Model Components

The initial version of this model can be composed of 4 sub models:

1. Shipping Generator – Pallets will be generated with a destination, and expiration time.
2. Shipping Pallet –Pallet will wait in a queue prioritized by expiration time until it can be loaded on an aircraft or expires and exits the queue.
3. Aircraft – Pallets are taken from the queue and loaded on an aircraft to transport between locations. Each time the aircraft flies it will reduce the time until the next maintenance cycle and crew change.
4. Destination – The destination location receives the pallets from the aircraft and provides them to the end customer after a given processing delay.

## Model Diagram



# Part 2 – Formal Model Specifications

## Notes

1. The smallest unit of time in this model is minutes. The EIRational time library was chosen to track time by the minute to support this model. During output, time is converted to hours or days depending on the situation.

## Atomic Models

### Pallet Generator = <X, Y, S, ta, δext, δint, λ>

|  |  |
| --- | --- |
|  | |
| **State Variables**  Sigma = 0  Phase = Waiting  Pallet = {} (<id, expiration time>) | **Formal Specifications**  X = {}  Y = {Pallet ∈ <N, R>}  S = {{Sigma, Phase, Pallet}} |
| δext {}  δint (e)  {  **case** phase:  waiting:  phase=busy;  pallet={current pallet id, expire time between 1 and X(default 10) days};  increase current pallet id by 1;  sigma=sigma+30 minutes;  busy:  phase=waiting;  pallet={};  sigma=next time between X(default 4) and Y(default 8) hours;  }  λ(s)  {  if !empty(pallet)  send pallet to port Out;  } | |

**Experimentation Strategy:**

Verify the following over a period of 4 days:

1. The generated pallets have an expiration time between 1 and 10 days; and
2. Pallets are generated every 4 to 8 hours.

### Packing Facility = <X, Y, S, ta, δext, δint, λ>

|  |  |
| --- | --- |
|  | |
| **State Variables**  Sigma = ∞  Phase = Waiting  Pallet Queue = {} <Pallet>  Expired Pallet Queue = {} <Pallet>  Load = {} (<id, expiration time, flight time>)  Aircraft Available = X (default 2) | **Formal Specifications**  X = {Pallet ∈ <N, R> | Aircraft Status ∈ <N, {0,1}>}  Y = {Load ∈ <N, R, R> | Expired Pallets ∈ <Pallet>}  S = {{Sigma, Phase, Pallet Queue, Expired Pallet Queue, Load, Aircraft Available}} |
| δext (Pallet Queue, e, x <Pallet | Aircraft Status>)  {  If x is Pallet  Add x to Pallet Queue;  Sort pallets by soonest expiration time;  If x is Aircraft Status  Update aircraft available to X;  Remove expired pallets from pallet queue and add to expired pallet queue;  If Aircraft available > 0 && Pallet Queue length >= Aircraft Load Size (default 4 pallets)  Load = {current load id, expire time is the lowest pallet expire time, flight time between 3 and X(default 30) hours};  Increase current load id by 1;  Remove top 4 pallets from pallet queue;  Phase = Load Ready  Sigma = 0  Else  Sigma = ∞  }  δint (e)  {  **case** phase:  Load Ready:  Load = {};  Expired Pallet Queue = {};  Phase = Waiting;  Sigma = ∞;  }  λ(s)  {  if !empty(Load)  send Load to port Out;  if !empty(Expired Pallet Queue)  send Expired Pallet Queue to port Out;  } | |

**Experimentation Strategy:**

Verify the following over a period of 4 days:

1. Loads are generated once the pallet queue reaches 4 and an aircraft is available;
2. Pallets expire; and
3. Pallets get queued waiting for the next available aircraft.

### Aircraft Loader = <X, Y, S, ta, δext, δint, λ>

|  |  |
| --- | --- |
|  | |
| **State Variables**  Sigma = ∞  Phase = Waiting  Aircraft Status Queue = All Available <Aircraft ID, Status>  Aircraft Load Queue = {} <Load> | **Formal Specifications**  X = {Load ∈ <N, R, flight time> | Aircraft Status ∈ <N, {0,1}>}  Y = {Aircraft Load ∈ <N, Load>}  S = {{Sigma, Phase, Aircraft Status Queue, Aircraft Load Queue}} |
| δext (Load, e, x <Load | Aircraft Status>)  {  If x is Load  Add load to aircraft load queue;  If x is Aircraft Status  Update Aircraft Status Queue  If available aircraft and load waiting in queue  Phase = Loading AC  Sigma = 0  Else  Sigma = ∞  }  δint (e)  {  **case** phase:  Loading AC:  Remove waiting load from aircraft load queue;  Phase = Waiting;  Sigma = ∞  }  λ(s)  {  if phase = Loading AC  send next Aircraft Load to port Out;  } | |

**Experimentation Strategy:**

1. When a load arrives, it is sent to the first available aircraft; and
2. When an aircraft becomes active it takes the next available load.

### Aircraft = <X, Y, S, ta, δext, δint, λ>

|  |  |
| --- | --- |
|  | |
| **State Variables**  Sigma = ∞  Phase = Waiting  Current Load = {} <Aircraft Load>  Total Flying Time = 0  Delivery Time=0 | **Formal Specifications**  X = {Aircraft Load ∈ <N, Load>}  Y = {Aircraft Load ∈ <N, Load> | Aircraft Status ∈ <N, {0,1}> }  S = {{Sigma, Phase, Current Load, Total Flying Time, Delivery Time}} |
| δext (Load, e, x <Aircraft Load>)  {  If Aircraft Load for this aircraft  Add x as current load;  Phase = transporting;  Add load flying time to total flying time;  Add load flying time to delivery time;  Sigma = Load flying time;  }  δint (e)  {  **case** phase:  Transporting:  Phase = Unloading;  Sigma = 30 minutes;  Add 30 minutes to delivery time;  Unloading:  Phase = Return;  Sigma = Load flying time;  Return:  Add load flying time to total flying time;  Current Load = {};  Phase = Waiting if no maintenance to be performed, or maintenance;  Send aircraft available status once any maintenance is completed;  Delivery Time = 0;  Sigma = ∞ (waiting) or 2 days if maintenance;  Maintenance:  Phase = Waiting;  Sigma = ∞;  Total flying time = 0;  }  λ(s)  {  If phase = unloading then send Current Load to port Out;  If aircraft busy send aircraft busy status;  } | |

**Experimentation Strategy:**

1. Aircraft only processes loads which belong to it;
2. Aircraft flying time and delivery time are accurate; and
3. Aircraft go into maintenance when they are supposed to.

### Destination = <X, Y, S, ta, δext, δint, λ>

|  |  |
| --- | --- |
|  | |
| **State Variables**  Sigma = ∞  Phase = Waiting  Current Load = {} <Aircraft Load> | **Formal Specifications**  X = {Aircraft Load ∈ <N, Load>}  Y = {Received Message ∈ <string>}  S = {{Sigma, Phase, Current Load }} |
| δext (Load, e, x <Aircraft Load>)  {  Add x as current load;  Phase = processing;  Sigma = destination processing time;  }  δint (e)  {  **case** phase:  Processing:  Phase = Waiting;  Sigma = ∞;  Current Load = {};  }  λ(s)  {  If phase = processing  Send received message to port Out;  } | |

**Experimentation Strategy:**

1. Ensure load messages are sent when the load is received.

## Coupled Models

### Loading Process = <*X*, *Y*, D, {Mi}, IC, EIC, EOC, select>

|  |  |
| --- | --- |
| X = | {Pallet ∈ <N, R>} |
| Y = | {Expired Pallets ∈ <Pallet>,  Aircraft Load ∈ <N, Load>} |
| D = | {Packing Facility,  Aircraft Loader,  Aircraft[iAircraft]} |
| IC = | {(PackingFacility.Load, AircraftLoader.Load),  (AircraftLoader.AircraftLoad, Aircraft.AircraftLoad),  (PackingFacility.AircraftStatus, Aircraft[iAircraft].AircraftStatus),  (AircraftLoader.AircraftStatus, Aircraft[iAircraft].AircraftStatus)} |
| EIC = | {(PackingFacility.Pallet, LoadingProcess.InputPallet)} |
| EOC = | {(PackingFacility.ExpiredPallets, LoadingProcess.OutputExpiredPallets), (Aircraft.AircraftLoad, LoadingProcess.OutputAircraftLoad)} |
| Select = | {Packing Facility, Aircraft Loader, Aircraft} = Aircraft  {Packing Facility, Aircraft Loader} = Aircraft Loader |

Notes: iAircraft can represent any number of aircraft.

**Experimentation Strategy:**

1. Ensure loads are taken by all aircraft.

### Strategic Airlift = <*X*, *Y*, D, {Mi}, IC, EIC, EOC, select>

|  |  |
| --- | --- |
| X = | {} |
| Y = | {Expired Pallets ∈ <Pallet>,  Received Message ∈ <string>} |
| D = | {Pallet Generator,  Loading Process,  Destination} |
| IC = | {(PalletGenerator.Pallet, LoadingProcess.InputPallet),  (LoadingProcess.AircraftLoad, Destination.AircraftLoad)} |
| EIC = | {} |
| EOC = | {(LoadingProcess.OutputExpiredPallets, StrategicAirlift.OutputExpiredPallets), (Destination.RecievedMessage,StrategicAirlift.OutputRecievedMessage)} |
| Select = | {Pallet Generator, Loading Process, Destination} = Destination  {Pallet Generator, Loading Process} = Loading Process |

# Part 3 – Model Experiments

## Notes

1. Input test cases are included with the source code and not in this document. This was done to allow the document to be clear and concise for the reader.
2. Tests were performed at the atomic level starting with the pallet generator and proceeding through the model flow. After atomic tests were performed, coupled tests were executed.
3. Program output is in italics.

## Atomic Experiments

### Pallet Generator (PalletGen\_Test.exe)

Input Files for Test

1. N/A

Description

1. The pallet generator requires no inputs and simply produces pallets (people and goods) to transport on aircraft. The test simulation is set for 4 days (5760 minutes) since the pallets are generated every 4 to 8 hours. This test will provide a reasonable check that the system is working as intended. Given this scenario there should be between 12 and 24 pallets produced.
2. When pallets are produced, they should have an expiration time between 1 and 10 days.

Results

1. When I ran the test, I received 45 pallets as noted by the output line:

*14100/1*

*[PalletGen\_defs::outPallet: {Pallet: 45 Expires At: 12 Days (18120 Minutes)}] generated by model myPalletGen*

1. From the 45 pallets I calculated the expiration days for each one as shown in the table below. These all fall within the 1 to 10 day range.

|  |  |
| --- | --- |
| **Pallet Id** | **Expiration Days** |
| 1 | 4 |
| 2 | 3 |
| 3 | 1 |
| 4 | 3 |
| 5 | 7 |
| 6 | 2 |
| 7 | 4 |
| 8 | 9 |
| 9 | 1 |
| 10 | 10 |
| 11 | 8 |
| 12 | 3 |
| 13 | 1 |
| 14 | 10 |
| 15 | 5 |
| 16 | 2 |
| 17 | 1 |
| 18 | 6 |
| 19 | 5 |
| 20 | 3 |
| 21 | 4 |
| 22 | 1 |
| 23 | 7 |
| 24 | 10 |
| 25 | 6 |
| 26 | 1 |
| 27 | 5 |
| 28 | 10 |
| 29 | 8 |
| 30 | 4 |
| 31 | 5 |
| 32 | 10 |
| 33 | 7 |
| 34 | 3 |
| 35 | 1 |
| 36 | 10 |
| 37 | 6 |
| 38 | 8 |
| 39 | 8 |
| 40 | 7 |
| 41 | 7 |
| 42 | 4 |
| 43 | 8 |
| 44 | 10 |
| 45 | 3 |

### Packing Facility (LoadPacking\_Test.exe)

Input Files for Test

1. LoadPackingTest\_Pallet.txt – Simulated input pallets from the Pallet Generator
2. LoadPackingTest\_AC.txt – Simulated status updated from the Aircraft

Description

1. Verify loads are generated after the 4th pallet arrives at time 40 minutes.
2. Verify pallets 6 and 7 expire once the simulation time is greater than 100 minutes.
3. Verify pallets get queued while waiting for the next available aircraft.
4. Verify an aircraft load is generated as soon as an aircraft is available and there are 4 pallets.

Results

1. Pallet Generated at minute 40 after 4th pallet arrives

*40/1*

*[LoadPacking\_defs::outLoad: {Load: 1 Expires At: 1 Days (2000 Minutes) Flight Time: 23 Hours (1380 Minutes)}, LoadPacking\_defs::outExpired: {}] generated by model myLoadPacking*

1. Pallet 6 and 7 expire at minute 110.

*110/1*

*[LoadPacking\_defs::outLoad: {Load: 2 Expires At: 1 Days (2000 Minutes) Flight Time: 21 Hours (1260 Minutes)}, LoadPacking\_defs::outExpired: {Pallet: 6 Expires At: 1 Days (1900 Minutes), Pallet: 7 Expires At: 1 Days (1900 Minutes)}] generated by model myLoadPacking*

1. Aircraft 1 becomes busy at minute 60 and does not become available until minute 110. At minute 94, 7 pallets are in the queue waiting for an aircraft.

*94/1*

*[cadmium::basic\_models::pdevs::iestream\_input\_defs<oPallet>::out: {Pallet: 11 Expires At: 1 Days (2300 Minutes)}] generated by model input\_reader\_genpallet*

1. At minute 110 there is an aircraft available and at least 4 pallets in the queue.

*110/1*

*[LoadPacking\_defs::outLoad: {Load: 2 Expires At: 1 Days (2000 Minutes) Flight Time: 21 Hours (1260 Minutes)}, LoadPacking\_defs::outExpired: {Pallet: 6 Expires At: 1 Days (1900 Minutes), Pallet: 7 Expires At: 1 Days (1900 Minutes)}] generated by model myLoadPacking*

### Aircraft Loader (AircraftLoader\_Test.exe)

Input Files for Test

1. AircraftLoaderTest\_Load.txt – Simulated loads from the Packing Facility
2. AircraftLoaderTest\_AC.txt – Simulated status updated from the Aircraft

Description

1. Verify when a load arrives, it is sent to the first available aircraft.
2. Verify when an aircraft becomes active it takes the next available load.

Results

1. Loads arrive at minute 10 and 20. They are put on aircraft 1 and 2 immediately upon arrival.

*10/1*

*[AircraftLoader\_defs::outLoad: {Aircraft Load: 1 Carrying Load: 1 Expires At: 3 Days (4320 Minutes) Flight Time: 5 Hours (300 Minutes)}] generated by model myAircraftLoad*

*20/1*

*[AircraftLoader\_defs::outLoad: {Aircraft Load: 2 Carrying Load: 2 Expires At: 3 Days (4320 Minutes) Flight Time: 5 Hours (300 Minutes)}] generated by model myAircraftLoad*

1. As soon as aircraft 1 becomes available again at minute 50 it is loaded.

*50/1*

*[cadmium::basic\_models::pdevs::iestream\_input\_defs<oLoad>::out: {Load: 5 Expires At: 3 Days (4320 Minutes) Flight Time: 5 Hours (300 Minutes)}] generated by model input\_reader\_getload*

*[cadmium::basic\_models::pdevs::iestream\_input\_defs<oAircraftStatus>::out: {Aircraft Status (1): Avaliable}] generated by model input\_reader\_acstatus*

*50/1*

*[AircraftLoader\_defs::outLoad: {Aircraft Load: 1 Carrying Load: 3 Expires At: 3 Days (4320 Minutes) Flight Time: 5 Hours (300 Minutes)}] generated by model myAircraftLoad*

### Aircraft (Aircraft\_Test.exe)

Input Files for Test

1. AircraftTest\_Load.txt – Simulated aircraft loads from the Aircraft Loader

Description

1. Verify aircraft only processes a load that belongs to it.
2. Verify aircraft flying time and delivery times are accurate.
3. Verify aircraft goes into maintenance when appropriate.

Results

1. Aircraft 1 ignores 1st load at minute 10 destine for aircraft 2. It then selects the correct load at minute 20.

*20/1*

*State for model input\_reader\_getload is next time: 80/1*

*State for model myAircraft is Aircraft: 1 Transporting ...*

1. The first load is 10 minutes of flying time one way. This should result in a total flying time of 20 minutes and delivery time of 40 minutes.

*60/1*

*State for model input\_reader\_getload is next time: 80/1*

*State for model myAircraft is Aircraft: 1 Completed Trip, Delivery Time: 40*

*70/1*

*State for model input\_reader\_getload is next time: 80/1*

*State for model myAircraft is Aircraft: 1 Waiting, Total Flying Time: 20*

1. Using a 6000 minute round trip flight at minute 200, it should go into maintenance mode around minutes 6200.

*6230/1*

*State for model input\_reader\_getload is next time: 6800/1*

*State for model myAircraft is Aircraft: 1 in Maintenance ...*

### Destination (Destination\_Test.exe)

Input Files for Test

1. DestinationTest\_Load.txt – Simulated aircraft loads from the Aircraft

Description

1. Verify load messages are sent when the load is received.

Results

1. Loads are received at minute 10, 20, and 100.

*10/1*

*State for model input\_reader\_getload is next time: 10/1*

*State for model myDestination is Destination: Processing Shipment 1 from Aircraft: 2*

*20/1*

*State for model input\_reader\_getload is next time: 80/1*

*State for model myDestination is Destination: Processing Shipment 2 from Aircraft: 1*

*100/1*

*State for model input\_reader\_getload is next time: inf*

*State for model myDestination is Destination: Processing Shipment 3 from Aircraft: 1*

## Coupled Experiments

### Loading Process (LoadingProcess\_Test.exe)

Input Files for Test

1. LoadingProcessTest\_Pallets.txt – Simulated pallets to transport

Description

1. Verify loads are taken by all aircraft.

Results

1. Loads generated by the Packing Facility were only transported by Aircraft 1. This requires a re-evaluation of the model to correct the error.
   1. Resolution required updating how aircraft are being tracked in the Packing Facility Code and revising sending aircraft status in the Aircraft code. Both changes are highlighted in red in the respective atomic models in Part 2.
   2. The second test was successful with both aircraft transporting loads correctly.

*70/1*

*State for model input\_reader\_getpallets is next time: 20/1*

*State for model myLoadPacking is Load Packing - Waiting*

*Load Packing - Aircraft Avaliable: 1*

*State for model myAircraftLoad is Aircraft Loader - Waiting*

*State for model myAircraft1 is Aircraft: 1 Transporting ...*

*State for model myAircraft2 is Aircraft: 2 Taking Off ...*

## Model Experiments

### Overview

The Strategic Airlift model is self contained and generates its own input pallets. The user can change many of the parameters as described below to experiment with the model. Defaults are shown in brackets.

LOG\_DIR (“../simulation\_results/”)

Allows the user to specify the directory to store the log files.

INPUT\_DIR (“../input\_data/”)

Allows the user to specify the directory for input files.

SCENARIO\_STOP (14400 minutes – 10 Days)

The scenario can run for any length of time. This option allows the user to specify the number of minutes the scenario should run for.

PALLET\_GEN\_MAX\_EXPIRE (10 days)

Used to set the maximum number of days for a pallet to expire.

PALLET\_GEN\_MIN\_NEW (4 hours)

The lower limit in hours to wait before the next pallet is generated.

PALLET\_GEN\_MAX\_NEW (8 hours)

The upper limit in hours to wait before the next pallet is generated.

LOAD\_EXPIRE\_BUFFER (LOAD\_FLIGHT\_MAX\*HOURS\_IN\_MIN)

The buffer time required for a load waiting to be transported to reach its destination. This value is used to determine if a pallet has expired.

LOAD\_AC\_SIZE (4 pallets)

Number of pallets an aircraft can carry.

LOAD\_FLIGHT\_MIN (3 hours)

The lower limit in hours for a flight to a destination.

LOAD\_FLIGHT\_MAX (30 hours)

The upper limit in hours for a flight to a destination.

AIRCRAFT\_NUM (2 aircraft)

Number of aircraft in the scenario. Changes to the TOP model will be required if this value changes.

AIRCRAFT\_MAINT\_TIME (2 days)

The number of days on average aircraft spend in maintenance when they reach a fixed number of flying hours.

AIRCRAFT\_MAINT\_LIMIT (100 hours)

The number of flying minutes before the next scheduled maintenance period.

DEST\_PROCESS\_TIME (1 hour)

The average processing time for a load at the destination.

### Experiments

Below are 2 of many experiments that could be run against this model.

#### E1 – Expired Pallet Generation Experiment

Input Files for Test

1. StratAirlift\_E1A.txt – Pallets generated between 1 and 5 hours
2. StratAirlift\_E1B.txt – Pallets generated between 5 and 9 hours
3. StratAirlift\_E1C.txt – Pallets generated between 9 and 12 hours

Description

1. Determine the effects of pallet generation frequency on the number of expired pallets produced over a simulation month. This allows multiple opportunities for expiration of pallets between 1 and 10 days.

Output Log Files

1. /Experiments/StratAirlift\_E1A\_messages.txt and StratAirlift\_E1A\_state.txt
2. /Experiments/StratAirlift\_E1B\_messages.txt and StratAirlift\_E1B\_state.txt
3. /Experiments/StratAirlift\_E1C\_messages.txt and StratAirlift\_E1C\_state.txt

Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trial** | **Pallets Generated** | **Pallets Expired** | **Percentage Loss** | **Expired Pallets** |
| A | 281 | 8 | 3% | 3, 12, 16, 24, 26, 29, 30, 44 |
| B | 109 | 8 | 7% | 3, 6, 12, 17, 18, 24, 29, 30 |
| C | 68 | 11 | 16% | 3, 6, 11, 12, 15, 16, 20, 22, 29, 30, 31 |

Based on the preliminary results it appears as though the number of pallets lost remains relatively consistent over the trials. This is most likely due to the Packing Facility prioritizing pallets with the soonest expiration.

It might be better to look at this problem through a different lens and reframe it as a network throughput problem or an optimization problem instead of a capacity issue. This has resulted in different questions to ask, such as experiment E2.

#### E2 – Pallet Expiration Time Experiment

Input Files for Test

1. StratAirlift\_E2A.txt – Pallets expiration between 1 and 5 days
2. StratAirlift\_E2B.txt – Pallets expiration between 1 and 10 days
3. StratAirlift\_E2C.txt – Pallets expiration between 1 and 15 days

Description

1. Following on from experiment E1, determine the effects of pallet expiration on the number of expired pallets produced over a simulation month. This allows multiple opportunities for expiration of pallets between 1 and 15 days.

Output Log Files

1. /Experiments/StratAirlift\_E2A\_messages.txt and StratAirlift\_E2A\_state.txt
2. /Experiments/StratAirlift\_E2B\_messages.txt and StratAirlift\_E2B\_state.txt
3. /Experiments/StratAirlift\_E2C\_messages.txt and StratAirlift\_E2C\_state.txt

Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trial** | **Pallets Generated** | **Pallets Expired** | **Percentage Loss** | **Expired Pallets** |
| A | 130 | 24 | 18% | 3, 6, 7, 12, 15, 16, 17, 20, 21, 22, 24, 26, 27, 31, 33, 35, 40, 44, 45, 46, 50, 51, 52, 55 |
| B | 130 | 8 | 6% | 3, 6, 12, 17, 18, 24, 29, 30 |
| C | 129 | 3 | 2% | 3, 20, 26 |

Based on the refined results from this experiment, it confirms that lower expiration time results in higher percentage loss of pallets.

The result of this finding is exploring an alternative hypothesis to the capacity question as a network throughput or load optimization problem. Both areas have been researched in the past and could contribute to future iterations of this model.

If future work continues with the capacity question it would need to add assumptions around the loads already being optimized and focus specifically on throughput. A reframed version of the question could be: “How many pallets could be delivered given N aircraft.”. This could be matched with a destination demand requirement and determine how many aircraft would be required to provide the supply.

1. Government of Canada (2019). Define Strategic Airlift. [online] TERMIUM. Available at: https://www.btb.termiumplus.gc.ca/tpv2alpha/alpha-eng.html?lang=eng&i=1&srchtxt=strategic+airlift&index=alt&codom2nd\_wet=1#resultrecs [Accessed 9 Oct. 2019]. [↑](#footnote-ref-1)