Improving the Operation of Overnight Intermodal Cargo Terminals Using Simulation and Optimization

Royce Bowden, Jr., PhD, Professor
Department of Industrial Engineering
Mississippi State University

Sriphani Raju Gadiraju, Graduate Research Assistant
Department of Industrial Engineering
Mississippi State University

Gay Reginelli
FedEx Corporation
Memphis, Tennessee

Charles Avery
FedEx Corporation
Memphis, Tennessee

C. Richard Cassady, PhD, Assistant Professor
Department of Industrial Engineering
University of Arkansas

A Report Submitted to the
National Center for Intermodal Transportation: A partnership between the
University of Denver and Mississippi State University

ABSTRACT

This research dealt with determining appropriate terminology, decision variables, parameters, constraints, and performance measures necessary for strategic, tactical, and operation analyses of overnight cargo terminals using simulation. Specially, a methodology was developed for programming a generic discrete-event simulation model in a general-purpose simulation language that could then be customized by its users to simulate specific overnight intermodal cargo terminals. The customized simulation model can then be used to conduct experiments to improve, or optimize, the performance of the terminal. To ensure that the research was well grounded and transferable, our industry partner was Federal Express Corporation.

BACKGROUND

A recent White Paper prepared by the National Center for Intermodal Transportation stated that the goal of the US should be to "create a transportation system that promotes efficiency, safety, mobility, economic growth and trade, national security, protection of the natural environment and enhancement of human welfare" (Bowden et al., 2001). One requirement for the achievement of this goal is that all modes of transportation must be connected to facilitate the convenient, expeditious, and efficient movement of commodities and people. Thus, intermodal terminal operations are a critical factor in the development of the nation's intermodal transportation system.

Some overnight cargo terminal operators have used simulation to improve ramp operations at their intermodal terminals. However, many companies are only beginning to explore the application of simulation to analyze package sorting operations with the goal of minimizing the time required to sort incoming packages such that packages with common destinations are placed in the same shipping containers. Furthermore, recent advances made in combining simulation with sophisticated optimization algorithms are expanding the utility of simulation. However, many academics and practitioners do not understand how to use this new simulation optimization software due to the novelty of the technology. Therefore, there is a need for researchers to develop methodologies that help academics and practitioners to properly formulate and solve intermodal transportation simulation optimization problems.

Building a discrete-event simulation model of a system is a primary activity in a simulation optimization study. In general, a <u>system</u> is a collection of resources that act together toward the accomplishment of some end. The system is the study of the subject of interest, and the focus of this study helps to determine the system boundaries, or scope. For example, if the system in question is the ground control of aircraft at the terminal, the boundaries include the airport's taxiways and ramps and exclude the airport's airspace. A <u>model</u> is a simplified representation, or abstraction, of a system. Here, our models are discrete-event simulation computer programs that "simulate" aspects of the system. This includes the arrival of loaded aircraft, the parking of aircraft on the ramp, the assignment of crews to unload parked aircraft and transport the cargo to the cargo sorting operation and then to load trucks and outbound aircraft (after the cargo sorting operation is completed).

In addition to the functional and logical relationships that describe system behavior, classical discrete-event simulation models have to be modified to include several other components such as decision variables, constraints and an objective function before they can be optimized. <u>Decision variables</u>, or input factors, are the quantities over which the decision-maker (or system manager) has control. Therefore, simulation models must be constructed in a way that allows users to easily modify the values of decision variables and to measure the effect that different values have on key performance measures. Parameters are values over which the decision-maker has no control. Examples of decision variables are the number of tugs used to unload aircraft, or the number of trailers assigned to each tug. Examples of parameters are the arrival rate of loaded aircraft to the terminal, or the arrival rate of loaded trucks to the terminal.

Constraints are any limitations that may be placed on the decision variables. Examples of constraints are area limitations for the number of aircraft parking spaces, budget limitations for operating the terminal, and towing capacity limitations of the tugs used to move cargo. A constraint may limit a single decision variable or a combination of decision variables.

Performance measures are quantities that capture the level to which the system is operating. Examples of performance measures are cargo throughput, waiting times, equipment utilization, operating costs, and inventory levels. An objective function identifies important performance measures and the optimization goal (maximize or minimize) for the measures. For example, an objective function may maximize utilization of tugs, minimize operating costs of a package sorting operation, or maximize the throughput (packages processed per night) of the terminal. In a simulation optimization model, decision variables, parameters, constraints, performance measures, and objective functions are all captured using equations and/or logical relationships.

The remainder of the report describes the FedEx system used to develop and demonstrate a methodology that other terminal operators can use to build simulation models of their system for the purpose of optimizing some aspect of the system. First, the problem is introduced and the scope of the system to be modeled is presented. Next, the terminology is developed to more formally describe the problem and the objectives for the system. From this foundation, we present the modeling methodology that we developed in conjunction with FedEx engineers that resulted in a relatively compact and easy to use simulation model of a rather large system.

PROBLEM STATEMENT

FedEx, our industrial partner, is one of the world's largest package delivery companies. The example system used to develop a modeling methodology is the sorting hub at Memphis, Tennessee. A large part of the sorting function at such hubs is carried out by use of conveyor belts. Therefore, we focused our study on the processing of conveyable packages.

The conveyable package sorting systems at the FedEx Memphis hub is basically a two-stage process, a primary sorting followed by a secondary sorting. After arriving packages are unloaded from inbound aircraft and trucks, they are processed through the sorting system based on their ID number. There are approximately 1,600 different IDs where each ID roughly corresponds to the identifier of the destination that the package is to be flown or trucked to next. Packages are sorted at the primary sorting area (Primary Matrix) and are carried via conveyors to one of 20+ secondary sorting areas. The actual number of secondary sorting areas is confidential. The packages at each of the secondary areas are further sorted and conveyed to one of 78 run-out belts that are connected to load-positions. The system is comprised of approximately 1,200 load-positions, which are used to place packages with common destinations into containers that are then loaded on outbound aircraft and trucks. Thus, the objective of the system is to sort incoming packages based on their destination in order to accumulate packages with common destinations on the appropriate mode of transportation.

In the event that accumulating packages fill a load-position to its capacity because they are arriving to the load position faster than they can be processed, the corresponding run-out belt on which this load-position is located is halted. If the overfill conditions is not quickly corrected by processing the packages from the load-position to outbound containers, then packages will begin queuing up at the secondary sorting areas feeding the load-position's run-out belt. This slows and can even stop the sorting of packages at the secondary, which can lead to packages backing up and congesting the primary-matrix. This results in under-utilization of some secondary areas and overloading of others.

The goal is to develop a modeling methodology that can be used by engineers to develop strategies to better balance the utilization of the sorting system's resources. Specifically, it is desired to find the optimum staffing and package routings, subject to constraints, that maximize the rate that packages are processed through the system (maximize the system's throughput).

Scope of the Model

The scope of the simulation model includes the following components from the FedEx System:

1. Arrival of packages, with IDs assigned to them, to the primary matrix.
2. Sorting of packages at the primary matrix based on package ID. This involves routing the packages to the appropriate secondary sorting areas. This is an automated process that requires little operator intervention.
3. Conveyor belts that transit the packages from the primary sorting matrix to the assigned secondary sorting areas.
4. Secondary slides for the queuing of packages at secondary sort areas.
5. Secondary sort areas for determining which of the 78 run-outs belts to send packages. This process requires an operator to remove packages from the slide queue, read its ID label, and place the package on the conveyor belt designed for the correct run-out belt.
6. Conveyor belts that transit the packages from the secondary sorting areas to run-out belts that circulate the packages to load-positions.
7. Load-position queues for packages sorted by common destination. This is the waiting area for the packages that are ready to be loaded into shipping containers.
8. Load-positions for loading packages into shipping containers.
9. Container staging area for swapping out shipping containers that have been filled at load-positions with empty shipping containers.

The scope of the model ends with the fully loaded shipping container being released for placement on an outbound aircraft or truck.

Project Terminology

<u>Parameters</u> are values over which the system engineer has little, or no, control. For the FedEx system used to demonstrate the methodology developed during this reseach, the parameters include:

1. Arrival rate of packages to the system.
2. The distribution of arriving package IDs.
3. Conveyor speed at which packages are moved between sorting stations.
4. The number of secondary sort areas.
5. The number of run-out belts and load-positions.

Decision variables are the quantities over which the decision-maker (or system engineer) has control. For the FedEx system studied, they included:

1. Number of sorters at each secondary.
2. The number of load-positions assigned to each run-out.
3. Number of checkers (sorters) at each load-position.
4. Routing of the packages with corresponding IDs from the primary matrix to secondary sorting areas and then to load-positions. In short, the assignment of package IDs to secondary sorting areas and load-positions.

Constraints are any limitations that may be placed on the decision variables. In the optimization context, these constraints should be met or satisfied. For the systems studied, constraints included:

1. The maximum number of sorters assigned to secondary sort areas.
2. The maximum number of load-positions on each run-out.
3. The maximum number of checkers (sorters) assigned to each load-positions
4. The possible ID assignments to secondary sort areas and load-positions.

Performance measures are quantities that capture the level at which the system is operating. For this system, the systems engineers identified the following performance measures:

1. Number of packages processed per unit time (system throughput).
2. Utilization of secondary sorting areas.
3. Percent of time run-out belts are filled to capacity, which stops the flow of packages to the run-out.

Objective functions capture the performance goals that the engineers have for the system. Generally, the objective is to maximize, minimize, or achieve a target value of a performance measure. A few objectives for this system are:

1. Maximize the number of packages processed per unit time (system throughput).
2. Balance the utilization of secondary sorting areas.
3. Minimize the percent of time run-out belts become filled to capacity.

Although the objectives are related, FedEx management favors the objective of balancing the utilization of the 20+ secondary sorting areas.

METHODOLOGY

Software Selection

The FedEx engineering staff selected ProModel, a modeling software product by PROMODEL Corporation of Orem, Utah, for modeling different aspects of the Memphis Hub. Therefore, the NCIT researchers used ProModel for the project. ProModel is one of many discrete-event simulation software packages suitable for this project. Although different discrete-event simulation software packages have unique features, they all operate using basically the same underlying technology. Therefore, the modeling approach researched for this project can be adapted for use in some of the other commercially available discrete-event simulation software packages.

ProModel is primarily designed to model discrete event processing systems. The basic modeling elements in ProModel are entities (the items being processed), locations (place where the processing takes place), resources (personnel, equipment, etc. required to process entities), and paths (the routes that entities move). In the context of the FedEx sorting system, entities are the packages in the system that are being sorted; locations are the areas where packages are sorted (primary matrix, secondary areas, run-out belts, and load-positions); resources are the sorting personnel, although they were modeled as capacities of the sorting locations for simplicity; and paths are the conveyor belts that move packages to their destinations in the sorting system.

The modeling methodology researched for this project relied heavily on ProModel's feature that allows for the integration of user written programming routines within the simulation model, ProModel's array construct, and ProModel's interface with Microsoft Excel. These features allowed the research team to develop a novel method for modeling complex systems with many sorting locations and package routing possibilities such that the final model can be easily modified and used without extensive knowledge of the underlying simulation language.

Modeling Approach

The system under consideration is rather large from a simulation-modeling standpoint in that it includes the primary matrix, 20+ secondary sort areas, 78 run-out belts, and 1200 load-positions. Considering that a queuing area for packages is provided for each of these locations, there is a requirement for 2,600+ locations in the model. Furthermore, the sorting system processes approximately 1,600 different package IDs through the system and the number of possible ways package ID can be assigned to different secondary sort areas and load-positions is also quite large. Therefore, modeling the system in a traditional manner would require a large amount of programming time, the program would be difficult to explain and maintain, and the time it would take the computer to complete a simulated sort would be prohibitively long. Thus, a traditional approach to modeling this system was not feasible.

Taking into account the above considerations, the NCIT researchers and FedEx engineers developed a non-traditional approach for building the model using a general-purpose simulation language. A simulation model was developed using a relatively few lines of code that captures the processing logic of the 2,600+ locations using only 71 ProModel locations. The simulation model can also be easily customized for a specific system, or system configuration, by simply changing values of variables that are logically organized in a Microsoft Excel spreadsheet. This modeling approach demonstrates how systems with a large number of processing locations, entities, and entity routings can be efficiently modeled using a simulation language's array construct.

Data and Modeling Requirements

The first segment of the modeling approach involved developing a multi-page Excel spreadsheet to organize key model information such as the assignment of package IDs to various secondary sort areas, run-out belts, and load-positions. The spreadsheet basically controls how packages are routed through the simulated sorting system. The process is such that the simulation program automatically opens and reads the contents of the spreadsheet before it begins simulating the sorting system. Thus, much of the logic represented in the simulation model is controlled by the contents of the spreadsheet. Therefore, users can evaluate alternative configurations of the sorting system by simply changing the contents of the spreadsheet. The alternative approach would require that the simulation program be "re-written" to run simulations of alternative system configurations that change how packages are routed through the system.

The following information was necessary for building the simulation model. It is presented following the order in which the package entities flow through the sorting system.

1. <u>Arrival rate of packages:</u> Data was collected on the rate packages arrive to the primary matrix. The arrival rate of packages varies with time over the course of an overnight sort. To obtain the data necessary for modeling this non-stationary arrival rate, we recorded the number of packages that arrived in contiguous 10-minute time segments for typical overnight sorts.

2. <u>Distribution of Package IDs:</u> Along with the count of the number of packages that arrive to the primary matrix in 10-minute time intervals, data on the IDs of these packages was collected to estimate the distribution of package IDs. A ProModel entity type was defined with an attribute that stores the package's ID. The package entity's attribute is assigned an integer number from 1 to 1,600, where each integer value maps to a unique FedEx ID. The integer ID numbers are referred to as ProModel IDs.

   Information on package arrivals to the primary matrix was organized in an Excel spreadsheet. To vary the percentage of different IDs arriving to the primary matrix, the user simply changes the values in the Excel spreadsheet and runs the model (note again, that no "re-programming' of the simulation model is required). Inserting '99999' in the last row of the Excel table flags the end of the different IDs.

   The sample input sheet provided in Table 1 shows the number of packages arriving to the primary matrix for the first seven 10 minute time intervals for the first 19 ID types. Note that at the beginning of a night's sorting activity that packages slowly trickle into the system as the first aircraft and trucks to arrive are unloaded. The arrival rate of packages to the primary matrix rapidly increases in subsequent time intervals.

Table 1. Excel spreadsheet for programming the frequency of package arrivals to the primary matrix.

		10-Minute Time Intervals						
		2300-2309	2310-2319	2320-2329	2330-2339	2340-2349	2350-2359	0000-0009
ProMod ID	FedEx ID	Count	Count	Count	Count	Count	Count	Count
1	ABCD	1	1	1	1	1	1	1
2	BCDE	1	1	1	1	1	1	1
3	EFGH	1	1	1	1	1	1	1
4	CDEF	1	1	1	1	1	1	1
5	DEFG	1	1	1	1	1	1	1
6	FGHI	0	1	1	1	1	1	1
7	IJKL	1	1	1	1	1	1	1
8	MNOP	1	1	1	1	1	1	1
9	QRST	1	1	1	1	1	1	1
10	UVW	0	1	1	1	1	8	10
11	NOPQ	1	1	1	1	1	1	1
12	OPQR	1	1	1	1	1	1	1

3. Routing of packages based on IDs: To configure the system for the sort, each of the secondary sort areas, run-out belts, and load-positions are assigned a list of IDs for which packages with corresponding IDs will be sent. This determines the route that a package will take through the system.

   Table 2 illustrates a portion of the Excel spreadsheet that contains the routing information. Note that each of the 20+ secondary sort areas, 78 run-out belts, and 1,200 load-positions are assigned a unique integer that corresponds to the FedEx code for that location. The ProModel simulation model conveys package entities from one location to the next by looking up the entities destination in this Excel spreadsheet based on the ProModel ID number assigned to the entity's attribute.

   It would be prohibitively slow for the simulation program to query the spreadsheet each time a package entity is to move to the next location. Therefore, the simulation model was programmed to read the spreadsheets at the beginning of a simulation in order to store the routing information in ProModel array variables. The end of the input to these arrays is flagged by filling the last row of the spreadsheet with '9999'.

Table 2. Excel spreadsheet for the package ID routing table.

ProMod ID	Fed Ex ID	ProMod SEC	FedEx Sec	ProMod RUNOUT	FedEx-Runout	ProMod LP
1	ABCD	12	S2R12	1	RO1	234
2	BCDE	1	S2R1	2	RO2	238
3	EFGH	1	S2R1	3	RO3	242
4	CDEF	2	S2R2	4	RO4	246
5	DEFG	3	S2R3	5	RO5	250
6	FGHI	3	S2R3	1	RO1	112
7	IJKL	12	S2R12	2	RO2	190
8	MNOP	1	S2R1	3	RO3	268
9	QRST	1	S2R1	4	RO4	293
10	UVW	2	S2R2	5	RO5	318
11	NOPQ	3	S2R3	1	RO1	343
12	OPQR	3	S2R3	2	RO2	368

4. Move times from the primary matrix to the secondary sort areas: This is the time the packages spend in transit on the conveyor belt before arriving at the secondary slide queue. Move time values were organized in an Excel spreadsheet, which ProModel reads at the beginning of a simulation.

5. Slide Queue capacity of secondary: Each secondary sort area provides a space, called the slide queue, for packages waiting to be sorted at the secondary. The slide queue capacity is the number of packages that can accumulate before the queue becomes full. This information is programmed in the ProModel simulation model as the capacity of the locations representing the 20+ slide queues.

6. Sorters at each secondary: The number of personnel sorting packages at each secondary sort area. This information is programmed in the ProModel simulation model as the capacity of the location representing the 20+ secondary sort areas.

7. Sorting time at each secondary: This is the time it takes a sorter to handle a package. FedEx engineers collected data on the amount of time it takes a sorter to remove a package from the slide queue, read the package's ID label, and place the package on the conveyor belt that routes the package to the proper run-out belt. The StatFit software package provided with ProModel was used to select a continuous distribution that most closely mimics the sorting time data. The distribution was programmed into the ProModel model to generate the time it takes to sort a package at a secondary.

8. Run-out belt: In the event that accumulating packages fill a load-position to its capacity because they are arriving to the load position faster than they can be processed, the corresponding run-out belt on which this load-position is located is halted. When the run-out belt is halted it stops the flow of packages from the secondary sorting area feeding the run-out belt. Therefore, the model had to be programmed to stop sending packages from a run-out belt to the load-positions attached to it if any of its load-positions fill up to its package holding capacity. An Excel spreadsheet was created, as illustrated in Table 3, that lists the load-positions assigned to each of the 78 run-out belts. The simulation program reads the spreadsheets at the beginning of a simulation in order to store the run-out information in ProModel array variables. The end of the input to these arrays is flagged by '9999' appearing in a spreadsheet cell. An Excel macro was developed to automatically

create this spreadsheet from the data provided in the package ID routing table, Table 2.

Table 3. Excel spreadsheet containing the assignment of load-positions to run-out belts.

Run-outs	Load Positions							
1	234	112	343	410	520	75	8	95	9999
2	238	190	368	435	712	9999	9999	9999	9999
3	242	268	410	21	20	24	30	27	25
4	246	293	452	33	35	36	34	31	9999
5	250	318	494	43	44	45	42	9999	9999
6	51	53	50	54	55	52	9999	9999	9999
7	61	62	67	66	60	63	64	65	58
8	71	72	74	73	69	70	9999	9999	9999
9	79	77	76	81	75	78	80	9999	9999
10	85	84	83	9999	9999	9999	9999	9999	9999

9. Move times from secondary to load-positions: This is the amount of time the packages spend in transit on the conveyor belt moving from the secondary sort area through run-outs and on to load-position slide queues. Move time values were organized in an Excel spreadsheet, which ProModel reads at the beginning of a simulation.

10. Slide Queues at load-position: The packages accumulate at the load-position slide queue waiting to be checked and loaded into outbound containers. The slide queue capacity is the number of packages that can accumulate before the queue becomes full. This information is programmed in the ProModel simulation model.

11. Checkers at each load-position: This is the number of personnel at each load-position checking packages and loading them into outbound containers. This information is programmed in the ProModel simulation model.

12. Checker times at load-positions: This is the time it takes a checker to load packages into outbound shipping containers. FedEx engineers collected data on the amount of time it takes to remove a package from the slide queue, read the package's ID label to verify that it has been properly sorted, and place the package into outbound shipping containers. The StatFit software package provided with ProModel was used to select a continuous distribution that most closely mimics the checking time data. The distribution was programmed into the ProModel model to generate the time it takes to check a package as it is processed at the load position.

13. Shipping container capacity: This is the number of packages that a shipping container can hold. The information is programmed in ProModel to trigger the swapping out of a full container that is ready to be shipped with an empty container that is ready to load at the load-position.

14. Swap time per container: This is the amount of time it takes a checker to exchange (swap) a full container with an empty container at the load position. FedEx engineers collected data on the amount of time it takes a checker to move the

loaded container away from the load-position and then move an empty container into the loaded container's place. The StatFit software package provided with ProModel was used to select a continuous distribution that most closely mimics the swap time. The distribution was programmed into the ProModel model to generate the time it takes to swap containers.

Given the large number of run-out belts and load-positions, a ProModel location for each of the 78 run-out belts and 1,200 load-positions and their corresponding queues is infeasible. Therefore, the second segment of our modeling approach involved the development of a novel technique to model these locations using a single location, which we called the run-out location in the ProModel model. Using this single location and ProModel array variables, a program was designed to simulate the activities of all run-out belts and load-positions. Additional details concerning the innovative modeling technique follow in the next section.

SIMULATION MODEL PROGRAM

In addition to providing an overview of the simulation model, this section describes important segments of the model that were designed not only to capture the complexity and size of overnight cargo sorting facilities but to also provide a relatively easy means for reprogramming the model to evaluate proposed changes to the system or to configure the model for a different sorting facility. See Appendix 1 for modeling details throughout this section as the appendix contains the complete ProModel simulation program.

Overview of Simulation Model

The required variables, arrays, and entity attributes are declared in the appropriate ProModel tables. The packages processed by the sorting facility simulation model are defined as an entity named "Box" in the ProModel Entities table. Box was selected as the entity name because the word contains fewer letters than does the word packages. In the ProModel Locations table, locations are defined for the primary matrix, secondary slide queues, secondary sorting areas, and for moving packages via the network of conveyor belts. A location named Runouts is defined where the logic for the operation of all 78 run-out belts and all 1,200 load-positions is captured.

Through the ProModel Initialization Logic section, the input data entered in the Excel spreadsheet is imported into the simulation model by assigning spreadsheet cell values to the arrays declared for this model. The ProModel Arrivals table contains the code that generates the arrival of packages (Box entities) to the system. The processing of the packages is captured in the ProModel Processing table. The processing logic involves the following steps.

<u>Step 1:</u> The packages (Box entities) arrive at the primary location, which is the entry point for the system being modeled. The Arrivals table contains the code that generates the arrival of packages and their assigned IDs based on the frequency information read from the Excel spreadsheet.

<u>Step 2:</u> The packages move from the primary location to the appropriate secondary queues based on the packages' ID. This routing information is obtained by importing data from the Excel spreadsheet into an array named *Routing_array* when ProModel runs the model's initialization logic (at the beginning of the simulation). The programming statement *LOC(Routing_array[ID_attr,3]+1* determines the routing of the package from

the primary to the secondary queue corresponding to the package's ID, which is stored in the entity's attribute named *ID-attr*. The transit time from the primary location to the secondary queue is captured at the *pri_sec_move* location using ProModel's *wait* statement coupled with the transit time stored in the routing array as given below.

```
wait primary_move[Routing_array[ID_attr,3],1]
```

<u>Step 3:</u> The packages move from the secondary queue to the corresponding secondary sort area where the packages will be sorted by ID and sent to run-out belts feeding the load-positions that correspond to the packages' ID. If a load-position's slide queue fills to its capacity, the run-out belt feeding it is stopped. When this happens, packages in the secondary queue that are destined for any load-position located on that run-out belt are not pulled from the secondary queue until space for the packages becomes available in the load-position's slide queue. The logic is captured as shown below, where the number of packages in queue at each load-position is tracked using the array *LP_QUE_array[Routing_array[ID_attr,7]]*, and the capacity of each load-position is stored in the array *LP_capacity_array[Routing_array[ID_attr,5],2]*, which is set at the beginning of the simulation by reading the capacity values from the Excel spreadsheet

```
 j=2
while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
     DO
     {
    wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
         LP_capacity_array[Routing_array[ID_attr,5],2]
    j=j+1
    }
```

Once the above conditions are satisfied, packages are removed from the secondary slide queue and sorted at the secondary sort area.

<u>Step 4:</u> A package is sorted at the secondary sort area by placing it on a conveyor belt that will move the package to the load-position assigned to process it. The processing of packages at secondary sorting areas is captured using ProModel's "*wait*" statement coupled with a continuous distribution that returns values that mimic the time it takes to sort a package. To protect confidential FedEx information, the ProModel simulation program code appearing in Appendix 1 does not show the continuous distribution that mimics the sorting time. Instead, a constant value of two has been entered.

<u>Step 5:</u> The transit logic of the packages from secondary sort areas to the load positions is captured at the location *sec_LP_move*. The transit time of a package from the secondary sort area to the corresponding load position is captured using ProModel's *wait* statement coupled with the transit time value retrieved from the Excel spreadsheet, which is stored in the array *sec_move[]*.

At *sec_LP_move*, the number of packages waiting in the load-position's slide queue is checked against the capacity of the load position's slide queue just as it was checked in Step 3. If the load position's slide queue is filled to capacity, then the package is queued at the *sec_LP_move* location. In effect, the package remains on the run-out belt feeding the load-position until it can be placed in the load-position's slide queue. The processing logic at sec_LP_move is shown below.

```
wait sec_move[Routing_array[ID_attr,7],1] //Move time from  secondary to load-position
 wait  until  LP_QUE_array[Routing_array[ID_attr,7]]<  LP_capacity_array[Routing_array[ID_attr,5],2]
    //Packages wait until there is  free capacity at the load position
 LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1   //record the number of
    entities in the Loadposition QUEUE.
 MLP_Que_stat      //calls ProModel macro to update statistics on load-position queues.
```

<u>Step 6:</u> The processing of the packages at various runout-belts and load-positions is captured at a single location named *Runouts*. First, the quantity of packages (Boxes) entering into their assigned load-position queues is tracked for each package ID using the statement

```
LPbox_order_array[Routing_array[ID_attr,7]]=LPbox_order_array[Routing_array[ID_attr,7]]+1
```

This quantity is then stored in the package entity's attribute using the statement

```
  LP_arrive_attr=LPbox_order_array[Routing_array[ID_attr,7]]    //Assign variable as attribute.
```

The number of checkers that are busy at a load-position is tracked using the variable *State_LP_array[Routing_array[ID_attr,7]].* If all checkers are busy, the package waits until one of the checkers becomes free. To simulate the processing of packages from load-position slide queues based on a first-in-first-out order, an array called *LPbox_turn_array[Routing_array[ID_attr,7]]* was created to keep track of which package (Box) is to be processed next. The logic is captured using the following lines of code assuming that each load-position is staffed with three checkers.

```
if State_LP_array[Routing_array[ID_attr,7]]=3 then   //Server (Checker) is not available. Wait for server.
    {
    WAIT UNTIL LP_arrive_attr<LPbox_turn_array[Routing_array[ID_attr,7]]+3  //Wait until it is the box's turn to be
        Processed
     }
```

If at least one of the three checkers is available when a package arrives to its load-position, the following code simulates the processing of the package at the load-position.

```
if  State_LP_array[Routing_array[ID_attr,7]]< 3 then    //Server (Checker) is available.
    {
     LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]]-1  //Box leaves the LP
        QUEUE
    MLP_Que_stat        //Update the variables.
     State_LP_array[Routing_array[ID_attr,7]]= State_LP_array[Routing_array[ID_attr,7]]+1
    WAIT checker_time_attr       //Process the entity.
    if LPbox_turn_array[Routing_array[ID_attr,7]] MOD FDE_pieces_attr=0 then wait swap_time   //Swap out
        time
     State_LP_array[Routing_array[ID_attr,7]]= State_LP_array[Routing_array[ID_attr,7]]-1
    LPbox_turn_array[Routing_array[ID_attr,7]]=LPbox_turn_array[Routing_array[ID_attr,7]]+1  //records the
```

```
        number of boxes processed+1
    }
```

The first step in the processing logic above is to update the number of packages waiting in the load-position slide queues and the status of the checkers, which corresponds to the array variables *LP_QUE_array[Routing_array[ID_attr,7]]* and *State_LP_array[Routing_array[ID_attr,7]]*, respectively. Next, the package is processed, which is represented in the model using ProModel's *wait* statement coupled with the box entity attribute *checker_time_attr* that is assigned a value representing the time it takes checkers to remove packages from slide queues and place them in shipping containers. The attribute is assigned a value in the ProModel move logic column at the secondary locations. The array variables *state_LP_array[Routing_array[ID_attr,7]]* and *LPbox_turn_array[Routing_array[ID_attr,7]]* are updated to reflect the process completion of the package (box).

Also, the number of packages an outbound shipping container (called FDE by FedEx) can hold is simulated. This capacity value is stored by the package (box) entity's attribute called *FDE_pieces_attr*. The package entity carries the shipping containers capacity since shipping containers of different sizes are used based on the package IDs being placed in them. The value of *FDE_pieces_attr* is assigned a value in the move logic at the secondary locations. As soon as an outbound container is filled to its capacity, the container is swapped with an empty container. This logic is implemented using ProModel's *MOD* division function in the following statement.

```
if LPbox_turn_array[Routing_array[ID_attr,7]] MOD FDE_pieces_attr=0 then wait swap_time   //Swap out
    time
```

The return of a value of zero by the *MOD* division function signals that the shipping container is full and needs to be replaced (swapped) with an empty shipping container. The swap time is programmed using the *wait* statement coupled with a variable called *swap_time*, which represents the time it takes to swap the containers and is assigned a value when it is declared as a variable in the ProModel Variables table.

The packages exit the system once they finish processing at load-positions (e.g., when full shipping containers are moved away from the load-positions). The total number of packages sorted by the system is tracked using a variable called *EXITS* using the statement.

```
EXITS = EXITS + 1
```

The complete code at this location in the processing logic is given below.

```
LPbox_order_array[Routing_array[ID_attr,7]]=LPbox_order_array[Routing_array[ID_attr,7]]+1
LP_arrive_attr=LPbox_order_array[Routing_array[ID_attr,7]]    //Assign variable as attribute.
if State_LP_array[Routing_array[ID_attr,7]]=3 then   // Server (Checker) is not available. Wait for server.
    {
    WAIT UNTIL LP_arrive_attr<LPbox_turn_array[Routing_array[ID_attr,7]]+1 //Wait until it is the box's turn to be
        processed
    }
```

```
if  State_LP_array[Routing_array[ID_attr,7]]<3 then //Server (Checker)  is available.
    {
     LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]]-1 //Box leaves the
          QUEUE
    MLP_Que_stat        //Update the variables.
     State_LP_array[Routing_array[ID_attr,7]]= State_LP_array[Routing_array[ID_attr,7]]+1
    WAIT checker_time_attr      //Process the entity.
    if LPbox_turn_array[Routing_array[ID_attr,7]] MOD FDE_pieces_attr=0 thenwait swap_time  //Swap out
        time
     State_LP_array[Routing_array[ID_attr,7]]= State_LP_array[Routing_array[ID_attr,7]]-1
    LPbox_turn_array[Routing_array[ID_attr,7]]=LPbox_turn_array[Routing_array[ID_attr,7]]+1  //records the
        number of boxes processed+1
    }
EXITS = EXITS + 1
```

Step 7: The utilization of the secondary is captured using two dummy locations named *Dummy_utl_loc1* and *Dummy_utl_loc2* and a dummy entity named *EntA*. A single *EntA* is created in the ProModel Arrivals table and circulates between the dummy locations every 10 minutes to update the array used to measure the utilization of each of the 20+ secondary sorting areas. The ProModel language includes a more efficient way to implement this logic but it was programmed in this manner because it can be duplicated in other simulation software packages. The piece of code that calculates the utilization of the secondary is shown below

```
wait 10
i=1
while I <= 20+
    Do
    {
    Sec_UTL_array[i]=Number_process_sec_array[i]*100/(SEC_target_cap_array[i,2]*clock(HR))
    i=i+1
    }
```

Model Output
Most discrete-event simulation packages automatically compute statistics for locations, resources, and entities in the simulation model. Typical statistics that are reported for the location element are the utilization of the location, the average number of entities that waited at the location, and the maximum number of entities to wait at the location. The main statistic for resources is usually the utilization of the resource. Entity statistics usually include the average, maximum, and minimum amount of time that entities spend at various locations in the simulated system.

Locations of most interest in this model are those that represent the queues for the secondary sort areas and load-positions. ProModel's location element is used to model the queues as well as the processing locations in the secondary sort area; thus, statistics are automatically provided by ProModel for those locations. Because of the large number of load-positions in the FedEx system, the ProModel location element was not used for load-positions and their respective queues. Instead, the function of these locations was simulated by a cleverly designed logic that greatly improved the efficiency of the simulation model. Therefore, variables were programmed to compute load-position statistics.

SIMULATION OPTIMIZATION

The goal of a simulation study is often to identify the settings for decision variables that will optimize the performance of the system being simulated. Until the arrival of optimization software like SimRunner, the typical practitioner attempted optimization with a somewhat trial-and-error processes of defining different scenarios by assigning different values to the decision variables in the simulation model and then simulating the different scenarios to see which scenario works best. It is usually impractical to evaluate all the possible scenarios because of the amount of time that it would take. The goal of simulation optimization is to automatically generate scenarios, evaluate them, and identify the best performing scenario (optimal or at least near optimal scenario) without evaluating all possible scenarios. The important contribution that optimization packages like SimRunner, OptQuest, Witness Optimizer, and others have made is that they consistently identify better scenarios faster than can be found by most practitioners on their own.

Several simulation optimization software packages, including SimRunner, are based on evolutionary algorithms. Evolutionary algorithms are problem-solving techniques based on the theory of evolution. They have proven to be robust in solving a wide variety of difficult problems. These algorithms seek the optimal solution scenario by processing a population of solution scenarios. As the algorithm identifies the superior solution scenarios, the algorithm makes small changes to the solution scenarios using mechanisms borrowed from the theory of evolution to obtain a new generation of solution scenarios that perform better than the previous generation. This methodology is continued until the algorithm is stopped or the individual solution scenarios in the population converge to the same scenario.

Application of SimRunner Optimization

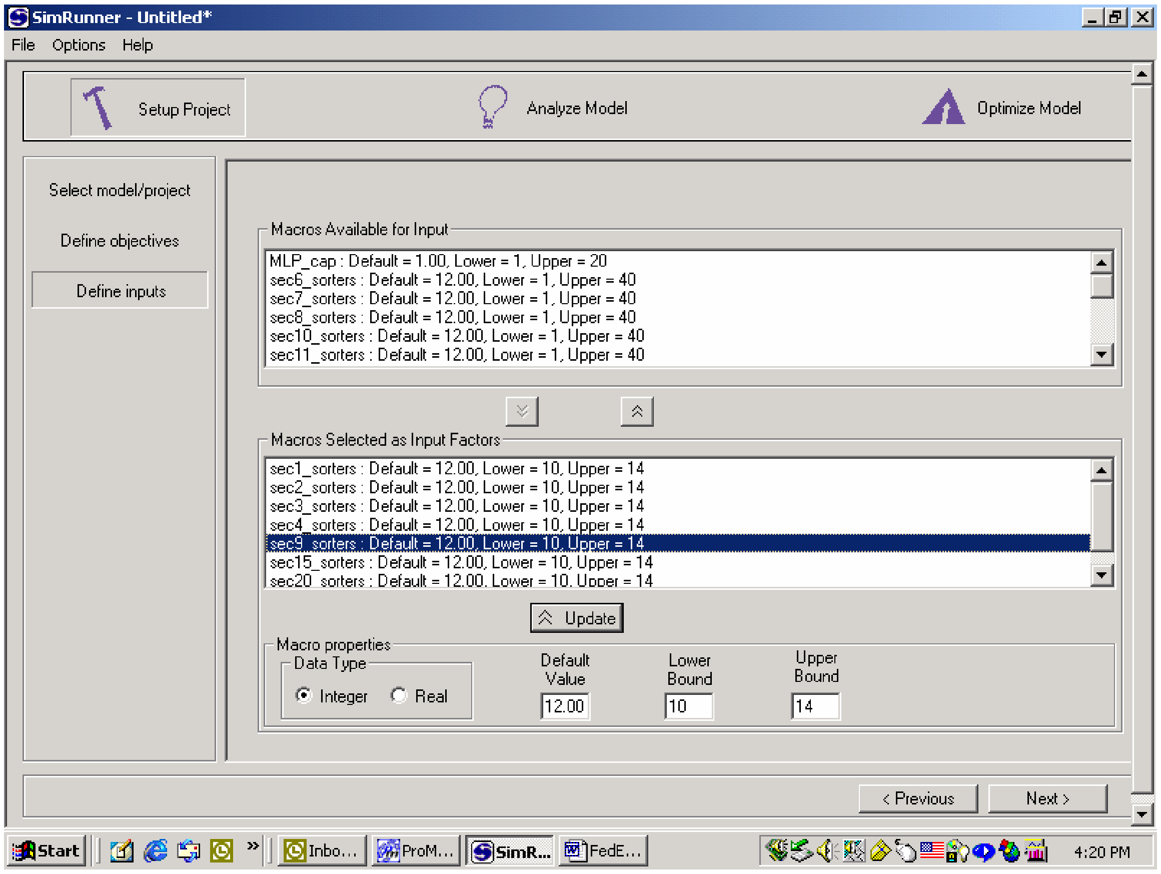
The fundamentals of setting up an optimization project are presented in this section. Although the presentation is tailored toward SimRunner and ProModel it is general in nature to facilitate those using other software products. Therefore, the focus is on process rather than software use.

To conduct optimization projects, the analyst needs to identify the decision variables, constraints, and an objective function for the system. After the analyst identifies the decision variables that might have an influence on the simulated system's performance, the analyst creates a macro for each of the decision variables and includes the macros in the ProModel Runtime Interface for the model. The decision variables to be optimized are referred to as input factors in SimRunner. The analyst defines constraints for the input factors by placing limits on their values. After the objective function is constructed using the performance measures of interest for the system, the analyst invokes the algorithm to search for the best performing solution scenario.

In order for operations like FedEx to be successful, their sorting system must be finely tuned to process the large volume of packages that arrive to the sorting facility each night. For this application example, the simulation model included a variable called *Exits* that records the number of packages sorted and loaded into outbound shipping containers. At the end of the sorting shift, the value of the variable *Exits* should equal the number of packages that arrived to the sorting system, otherwise packages were left behind. This is essentially the system's throughput, which was previously defined as one of the performance measures of interest and will be used to define the optimization project's objective function. Before completing the objective function, however, the input factors (decision variables) are first defined.

The four decision variables previously defined for the system included the number of sorter employees assigned to each secondary sort area. There are 20+ secondary sort areas in the system. Clearly, the number of sorters assigned to the 20+ positions will influence the throughput of the system. The use of too few sorters will prevent the system from completing the sort on time. Too many sorters, while achieving the desired throughput, will add unnecessary cost to the sort. The goal is to determine the optimal number of sorters to assign to each secondary sort area.

Figure 1illustrates the SimRunner screen for setting up the input factors using the macros that were built into the simulation model. The macros used to assign the number of sorters to secondary sort areas one through 20+ are sec1_sorters, sec2_sorters, . . . sec19_sorters, sec20_sorters, . . ., respectively. Figure 1 indicates that the analyst has constrained the number of sorters assigned to a secondary sort are to be no fewer than 10 sorters (lower bound) and no more than 14 sorters (upper bound). For this example, the integer data type, versus the real data type, is appropriately selected for the number of sorters.



Macros Available for Input

MLP_cap : Default = 1.00, Lower = 1, Upper = 20
sec6_sorters : Default = 12.00, Lower = 1, Upper = 40
sec7_sorters : Default = 12.00, Lower = 1, Upper = 40
sec8_sorters : Default = 12.00, Lower = 1, Upper = 40
sec10_sorters : Default = 12.00, Lower = 1, Upper = 40
sec11_sorters : Default = 12.00, Lower = 1, Upper = 40

Macros Selected as Input Factors

sec1_sorters : Default = 12.00, Lower = 10, Upper = 14
sec2_sorters : Default = 12.00, Lower = 10, Upper = 14
sec3_sorters : Default = 12.00, Lower = 10, Upper = 14
sec4_sorters : Default = 12.00, Lower = 10, Upper = 14
sec9_sorters : Default = 12.00, Lower = 10, Upper = 14
sec15_sorters : Default = 12.00, Lower = 10, Upper = 14
sec20_sorters : Default = 12.00, Lower = 10, Upper = 14

Macro properties

Data Type: [✓ Integer] Real

Default Value: 12.00
Lower Bound: 10
Upper Bound: 14

Figure 1. Snapshot of SimRunner screen for specifying input factors to optimize.

During the search, SimRunner generates solution scenarios by varying the values of the input factors according to their data types, lower bound, and upper bound. The number of input factors and the range of possible values affect the size of the search space (number of possible solution scenarios to the problem). Increasing the number of input factors, or their range of values, increases the size of the search space, which can make it more difficult and time-consuming to identify the optimal solution scenario. As a rule, only include those input factors known to significantly affect the output of the simulation model and judiciously define the range of possible values for each input factors. Also, care should be taken when defining the lower and upper bounds of the input factors to ensure that a combination of values will not be created that lead to a solution scenario that was not envisioned when the model was built. For this example, there are 5^{20+} different solution scenarios considering each of the 20+ input factors has five possible values (10, 11, 12, 13, 14), which is an extremely large search space. Thus, the analyst probably needs to be more selective in picking the secondary sort areas to include in the optimization. Perhaps only 7 or 8 of the most critical secondary sort areas in the system should be selected for optimization.

After selecting the input factors, the objective function is constructed to measure the performance of the solution scenarios tested by the optimizer. The objective function is built using terms taken from the output report generated at the end of the simulation run. Objective function terms can be based on entity statistics, location statistics, resource statistics, variable statistics, and so on. The user specifies whether a term is to be minimized or maximized as well as the weighting coefficient of the term in the objective function. Some terms may be more or less important to the user than other terms. One way to think of the weighting coefficient is as a dollar amount in terms of profit (a positive coefficient) or loss (a negative coefficient). As terms are added to the objective function, the complexity of the search space may increase, which makes for a more difficult optimization problem. From a statistical point of view, single-term objective functions are preferable to multiterm objective functions. Therefore, strive to keep the objective function as specific as possible.

For this example, the goal is to achieve a target number of packages processed by the system during the sorting shift. As previously mentioned, the simulation model includes a variable called *Exits*, which represents the number of packages processed completely through the system. For the convenience of including the variable *Exits* as an objective function term, this performance measure can be computed as a proportion by the equation following

$$ThroughputPorportion = \frac{Exits}{Arrivals}$$

where arrivals is a variable programmed in the model to count the number of packages that arrived to the system for sorting. *ThroughputPorportion* represents the proportion of arriving packages to the system that completed the sort in time to be shipped. The equation for the proportion would be programmed into the model's Termination Logic, which is executed at the end of the simulation. *ThroughputPorportion* will range from zero to one, with one being the ideal value. When defining the *ThroughputPorportion* as a variable in ProModel, the analyst will need to indicate that statistics are to be computed for the variable in order for it to be used as an objective function term in SimRunner.

To prevent the optimization algorithm from simply assigning the maximum number of sorters to each secondary sorting area to maximize *ThroughputPortion*, the number of secondary sorters needs to be reflected in the objective function. In reality, the goal is to find the minimum number of secondary sorters that will maximize *ThroughputPorportion*. One way of doing this is to include the number of secondary sorters used in a solution scenario as a penalty function in the objective function. The penalty function serves to reduce the value of the objective function for a solution scenario based on the number of sorters the scenario places in the secondary sort areas. Assuming that eight secondary sort areas are to be optimized, the algorithm can assign a maximum of 14 sorters to each secondary sort area, which would result in the use of 112 sorters. To keep the penalty function in scale with *ThroughputPorportion*, a second proportion is formed that is equal to the total number of sorters assigned to the secondary sort areas divided by 112. In this case, the proportion would range from 0.7143 to 1.0. The value of 0.7143 occurs when the minimum number of 10 sorters are assigned to each of the eight secondary sort areas (80/112). The value of 1.0 occurs when the maximum of 14 sorters are assigned to each of the eight secondary sort areas (112/112). This secondary sorter proportion could be computed in the model's termination logic by

$$SecondarySorterProportion = \frac{sec1\_sorters + sec2\_sorters... + sec8\_sorters}{112}$$

assuming that the number of sorters to be assigned to secondary sorting positions one through eight are selected as input factors. As with the *ThroughputPorportion* variable, the *SecondarySorterProportion* needs to be defined as a variable for which statistics are required in order to include it as an objective function term in SimRunner.

The final format for the objective functions becomes

$$Maximize\ \{(C_1 * ThroughputPorportion) - (C_2 * SecondarySorterProportion)\}$$

where C_1 and C_2 are the weighting coefficients for each proportion. If achieving a high *ThroughputPorportion* is twice as important as achieving a low *SecondarySorterProportion* for example, a C_1 value of 2.0 and a C_2 value of 1.0 would be selected. The objective function will be at its maximum value when all packages arriving to the system are sorted on time using the minimum number of secondary sorters. This is but one possible arrangement of the objective function for this goal.

It is important to understand that the objective function is a random variable, and a set of initial experiments should be conducted to estimate its variability (standard deviation). Note that there is a possibility that the objective function's standard deviation differs from one solution scenario to the next. Therefore, the required number of replications necessary to estimate the expected value of the objective function may change from one solution scenario to the next. Thus, the objective function's standard deviation should be measured for several different solution scenarios and the highest standard deviation recorded used to compute the number of replications necessary to estimate the expected value of the objective function to within an acceptable level of precision. When selecting the set of test solution scenarios, choose solution scenarios that are very different from one another. For example, form solutions by setting the decision variables to their lower bounds, middle values, or upper bounds.

Another approach for controlling the number of replications used to estimate the expected value of the objective function for a given solution scenario would be to incorporate a rule into the model that schedules additional replications until the estimate reaches a desired level of precision. Using this technique can help to avoid running too many replications for some solution scenarios and too few replications for others. See Harrell, Gosh, and Bowden (2004) for additional details.

After completing the input factors, constraints, objective function, and the determination of the number of replications, the next step is to start the algorithm's search for the optimal number of sorters assigned to the secondary sort areas. SimRunner allows the user to select an optimization profile that influences the degree of thoroughness used to search for the optimal solution. The three optimization profiles are aggressive, moderate, and cautious, which correspond to an evolutionary algorithm's population size of small, medium, and large. The aggressive profile generally results in a quick search for locally optimal solutions and is used when computer time is limited. The cautious profile specifies that a more thorough search for the global optimum be conducted and is used when computer time is plentiful.

After the algorithm's search has concluded (or halted due to time constraints), the analyst should carefully analyze the solution scenarios found by the algorithm. A good practice is to run additional replications of the best performing solution scenarios to get better estimates of their true performance before picking one as best.

SUMMARY

The package sorting system under consideration is rather large from a simulation-modeling standpoint. A simulation modeling methodology was developed to efficiently model these large systems. The methodology was used to model a FedEx system with the ProModel simulation language using a relatively few lines of code to capture the processing logic of 2,600+ processing locations using only 71 ProModel locations. Decision variables, parameters, constraints, and performance measures common to large-scale overnight package sorting facilities were defined. Based on the definitions, a simulation optimization project was formulated to optimize the assignment of sorting personnel to various positions in the sorting system.

REFERENCES

Bowden, R. O., Carmichael, G. E., Connery, N. R., Dempsey, P. S., Szyliowicz, J. S., Watt, W. J., and Zhang, Y. Z., 2001, "A New Transportation Agenda for America in the Aftermath of 11 September 2001," National Center for Intermodal Transportation.

Harrell, C.R., Ghosh, B., and Bowden, R.O., 2004, *Simulation Using ProModel,* 2nd ed., McGraw-Hill, New York, NY.

APPENDIX – PROMODEL SIMULATION CODE

```
********************************************************************************
*                                                                              *
*                          Formatted Listing of Model:                         *
*                D:\Mydocuments\project\Finalreport\sim-model.MOD              *
*                                                                              *
********************************************************************************

  Time Units:                         Minutes
  Distance Units:                     Feet
  Initialization Logic:


                                      while i<1200  DO
                                         {
                                            LPbox_turn_array[i]=1   // Initialize the box turn to 1.
                                            State_LP_array[i]=0     // Initialize the state of Loadpositions to 0.
                                            LP_QUE_array[i]=0
                                            i=i+1                   // Increment the loop counter.
                                         }

                                      j=0
                                      While j<25
                                      DO
                                       {
                                        j=j+1
                                        Total_arrivals[j]=0
                                        i=1

                                      DO
                                        {
                                         Total_arrivals[j]=ID_assign_array[i,j+2]+Total_arrivals[j]
                                         i=i+1
                                        }until ID_assign_array[i,j+2]=99999

                                       }
```

```
********************************************************************************
*                                  Locations                                   *
********************************************************************************

  Name           Cap           Units Stats       Rules      Cost
  -------------- ------------- ----- ----------- ---------- ------------
  PRIMARY        inf           1     Time Series Oldest, ,
  SEC1_QUE       300           1     Time Series Oldest, ,
  SEC2_QUE       300           1     Time Series Oldest, ,
  SEC3_QUE       300           1     Time Series Oldest, ,
  SEC4_QUE       300           1     Time Series Oldest, ,
  SEC5_QUE       300           1     Time Series Oldest, ,
  SEC6_QUE       300           1     Time Series Oldest, ,
  SEC7_QUE       300           1     Time Series Oldest, ,
  SEC8_QUE       300           1     Time Series Oldest, ,
  SEC9_QUE       300           1     Time Series Oldest, ,
  SEC10_QUE      300           1     Time Series Oldest, ,
  SEC11_QUE      300           1     Time Series Oldest, ,
  SEC12_QUE      300           1     Time Series Oldest, ,
  SEC13_QUE      300           1     Time Series Oldest, ,
  SEC14_QUE      300           1     Time Series Oldest, ,
  SEC15_QUE      300           1     Time Series Oldest, ,
  SEC16_QUE      300           1     Time Series Oldest, ,
  SEC17_QUE      300           1     Time Series Oldest, ,
  SEC18_QUE      300           1     Time Series Oldest, ,
  SEC19_QUE      300           1     Time Series Oldest, ,
  SEC20_QUE      300           1     Time Series Oldest, ,
        .
        .
        .
  Pri_sec_move   inf           1     Time Series Oldest, ,
  Sec1           sec1_sorters  1     Time Series Oldest, ,
  Sec2           sec2_sorters  1     Time Series Oldest, ,
  Sec3           sec3_sorters  1     Time Series Oldest, ,
  Sec4           sec4_sorters  1     Time Series Oldest, ,
  Sec5           sec5_sorters  1     Time Series Oldest, ,
  Sec6           sec6_sorters  1     Time Series Oldest, ,
  Sec7           sec7_sorters  1     Time Series Oldest, ,
  Sec8           sec8_sorters  1     Time Series Oldest, ,
  Sec9           sec9_sorters  1     Time Series Oldest, ,
  Sec10          sec10_sorters 1     Time Series Oldest, ,
  Sec11          sec11_sorters 1     Time Series Oldest, ,
  Sec12          sec12_sorters 1     Time Series Oldest, ,
  Sec13          sec13_sorters 1     Time Series Oldest, ,
  Sec14          sec14_sorters 1     Time Series Oldest, ,
  Sec15          sec15_sorters 1     Time Series Oldest, ,
  Sec16          sec16_sorters 1     Time Series Oldest, ,
  Sec17          sec17_sorters 1     Time Series Oldest, ,
  Sec18          sec18_sorters 1     Time Series Oldest, ,
  Sec19          sec19_sorters 1     Time Series Oldest, ,
  Sec20          sec20_sorters 1     Time Series Oldest, ,
        .
        .
        .
  sec1_LP_move   inf           1     Time Series Oldest, ,
  sec2_LP_move   inf           1     Time Series Oldest, ,
  sec3_LP_move   inf           1     Time Series Oldest, ,
  sec4_LP_move   inf           1     Time Series Oldest, ,
  sec5_LP_move   inf           1     Time Series Oldest, ,
  sec6_LP_move   inf           1     Time Series Oldest, ,
  sec7_LP_move   inf           1     Time Series Oldest, ,
  sec8_LP_move   inf           1     Time Series Oldest, ,
  sec9_LP_move   inf           1     Time Series Oldest, ,
  sec10_LP_move  inf           1     Time Series Oldest, ,
  sec11_LP_move  inf           1     Time Series Oldest, ,
  sec12_LP_move  inf           1     Time Series Oldest, ,
  sec13_LP_move  inf           1     Time Series Oldest, ,
  sec14_LP_move  inf           1     Time Series Oldest, ,
  sec15_LP_move  inf           1     Time Series Oldest, ,
  sec16_LP_move  inf           1     Time Series Oldest, ,
  sec17_LP_move  inf           1     Time Series Oldest, ,
  sec18_LP_move  inf           1     Time Series Oldest, ,
  sec19_LP_move  inf           1     Time Series Oldest, ,
  sec20_LP_move  inf           1     Time Series Oldest, ,
        .
        .
        .
  Runouts        inf           1     Time Series Oldest, ,
  Dummy_utl_loc1 1             1     Time Series Oldest, ,
  Dummy_utl_loc2 1             1     Time Series Oldest, ,

********************************************************************************
*                                   Entities                                   *
********************************************************************************

  Name       Speed (fpm)  Stats       Cost
  ---------- ------------ ----------- ------------
  BOX        150          Time Series
  EntA       150          Time Series


********************************************************************************
*                                  Processing                                  *
```

```
********************************************************************************

                                   Process                                       Routing

 Entity   Location       Operation            Blk  Output   Destination                    Rule     Move Logic
 -------- -------------- ------------------   ---- -------- ------------------------------ -------  ------------
 BOX      PRIMARY                             1    BOX      Pri_sec_move                   FIRST 1
 BOX      Pri_sec_move   wait primary_move[Routing_array[ID_attr,3],1]
                                              1    BOX      LOC(Routing_array[ID_attr,3]+1 FIRST 1
 BOX      SEC1_QUE       Sec1_Q = Sec1_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                             wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                             j=j+1
                         }

                                              1    BOX      Sec1                           FIRST 1
 BOX      SEC2_QUE       Sec2_Q = Sec2_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                           wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }
                                              1    BOX      Sec2                           FIRST 1
 BOX      SEC3_QUE       Sec3_Q = Sec3_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                            wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                             j=j+1
                         }
                                              1    BOX      Sec3                           FIRST 1
 BOX      SEC4_QUE       Sec4_Q = Sec4_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                           wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
```

```
                           j=j+1
                        }
                                            1    BOX      Sec4                          FIRST 1
 BOX      SEC5_QUE      Sec5_Q = Sec5_Q + 1
                        j=2
                        while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                        DO
                        {
                         wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                          j=j+1
                        }

                                            1    BOX      Sec5                          FIRST 1
 BOX      SEC6_QUE      Sec6_Q = Sec6_Q + 1
                        j=2
                        while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                        DO
                        {
                         wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                          j=j+1
                        }

                                            1    BOX      Sec6                          FIRST 1
 BOX      SEC7_QUE      Sec7_Q = Sec7_Q + 1
                        j=2
                        while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                        DO
                        {
                         wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                          j=j+1
                        }

                                            1    BOX      Sec7                          FIRST 1
 BOX      SEC8_QUE      Sec8_Q = Sec8_Q + 1
                        j=2
                        while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                        DO
                        {
                         wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                          j=j+1
                        }
                                            1    BOX      Sec8                          FIRST 1
 BOX      SEC9_QUE      Sec9_Q = Sec9_Q + 1
```

```
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                          wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }

                                              1    BOX      Sec9                           FIRST 1
 BOX      SEC10_QUE      Sec10_Q = Sec10_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                          wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }

                                              1    BOX      Sec10                          FIRST 1
 BOX      SEC11_QUE      Sec11_Q = Sec11_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                          wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }
                                              1    BOX      Sec11                          FIRST 1
 BOX      SEC12_QUE      Sec12_Q = Sec12_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                          wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }
                                              1    BOX      Sec12                          FIRST 1
 BOX      SEC13_QUE      Sec13_Q = Sec13_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                          wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }


                                             1    BOX      Sec13                          FIRST 1
 BOX      SEC14_QUE      Sec14_Q = Sec14_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                          wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }
                                             1    BOX      Sec14                          FIRST 1
 BOX      SEC15_QUE      Sec15_Q = Sec15_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                          wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }
                                             1    BOX      Sec15                          FIRST 1
 BOX      SEC16_QUE      Sec16_Q = Sec16_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                          wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }

                                             1    BOX      Sec16                          FIRST 1
 BOX      SEC17_QUE      Sec17_Q = Sec17_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                          wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }
```

```
                                          1    BOX      Sec17                          FIRST 1
 BOX      SEC18_QUE      Sec18_Q = Sec18_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                          wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }
                                          1    BOX      Sec18                          FIRST 1
 BOX      SEC19_QUE      Sec19_Q = Sec19_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                          wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }
                                          1    BOX      Sec19                          FIRST 1
 BOX      SEC20_QUE      Sec20_Q = Sec20_Q + 1
                         j=2
                         while  LP_runout_array[Routing_array[ID_attr,5],j]!=9999
                         DO
                         {
                          wait until  LP_QUE_array[LP_runout_array[Routing_array[ID_attr,5],j]]<
LP_capacity_array[Routing_array[ID_attr,5],2]
                           j=j+1
                         }
                                          1    BOX      Sec20                          FIRST 1
        .
        .
        .

 BOX      Sec1           Sec1_Q = Sec1_Q - 1
                         WAIT 2
                         dest_loc=47


                                          1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=47  then
                                                                                                {FDE_pieces_attr=200
                                                                                                 checker_time_attr=0.4
                                                                                        Number_process_sec_array[1]=Number_process_sec_array[1]+1
                                                                                                }
```

```
BOX      Sec2          Sec2_Q = Sec2_Q - 1
                       WAIT 2
                       dest_loc=48
                                 1    BOX      loc(dest_loc)                  FIRST 1  if dest_loc=48  then
                                                                                       {FDE_pieces_attr=200
                                                                                        checker_time_attr=0.4
                                                            Number_process_sec_array[2]= process_sec_array[2]+1
                                                                                       }

BOX      Sec3
                       Sec3_Q = Sec3_Q - 1
                       WAIT 2
                         dest_loc=49
                                 1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=49  then
                                                                                       {FDE_pieces_attr=200
                                                                                        checker_time_attr=0.4
                                                        Number_process_sec_array[3]=Number_process_sec_array[3] + 1
                                                                                       }

BOX      Sec4
                       Sec4_Q = Sec4_Q - 1
                       WAIT 2
                         dest_loc=50
                                 1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=50  then
                                                                                       {FDE_pieces_attr=200
                                                                                        checker_time_attr=0.4
                                                         Number_process_sec_array[4]=Number_process_sec_array[4] + 1
                                                                                       }

BOX      Sec5
                       Sec5_Q = Sec5_Q - 1
                       WAIT 2
                           dest_loc=51
                                 1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=51  then
                                                                                       {FDE_pieces_attr=200
                                                                                        checker_time_attr=0.4
                                                         Number_process_sec_array[5]=Number_process_sec_array[5] + 1
                                                                                       }

BOX      Sec6
                       Sec6_Q = Sec6_Q - 1
                       WAIT 2
                         dest_loc=52
                                 1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=52  then
                                                                                       {FDE_pieces_attr=200
                                                                                        checker_time_attr=0.4
```

```
                                                                                          Number_process_sec_array[6]=Number_process_sec_array[6] + 1
                                                                                                                         }

BOX      Sec7
                        Sec7_Q = Sec7_Q - 1
                        WAIT 2
                            dest_loc=53
                                                1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=53  then
                                                                                                  {FDE_pieces_attr=200
                                                                                                   checker_time_attr=0.4
                                                                                          Number_process_sec_array[7]=Number_process_sec_array[7] + 1
                                                                                                                         }

BOX      Sec8
                        Sec8_Q = Sec8_Q - 1
                        WAIT 2
                           dest_loc=54
                                                1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=54  then
                                                                                                  {FDE_pieces_attr=200
                                                                                                   checker_time_attr=0.4
                                                                                          Number_process_sec_array[8]=Number_process_sec_array[8] + 1
                                                                                                                         }

BOX      Sec9
                        Sec9_Q = Sec9_Q - 1
                        WAIT 2
                           dest_loc=55

                                                1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=55  then
                                                                                                  {FDE_pieces_attr=200
                                                                                                   checker_time_attr=0.4
                                                                                          Number_process_sec_array[9]=Number_process_sec_array[9] + 1
                                                                                                                         }

BOX      Sec10
                        Sec10_Q = Sec10_Q - 1
                        WAIT 2
                          dest_loc=56


                                                1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=56  then
                                                                                                  {FDE_pieces_attr=200
                                                                                                   checker_time_attr=0.4
                                                                                        Number_process_sec_array[10]=Number_process_sec_array[10]+1
                                                                                                                         }

BOX      Sec11
```

```
                              Sec11_Q = Sec11_Q - 1
                              WAIT 2
                                 dest_loc=57
                                                   1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=57  then
                                                                                                         {FDE_pieces_attr=200
                                                                                                          checker_time_attr=0.4
                                                                            Number_process_sec_array[11]=Number_process_sec_array[11]+1
                                                                                                          }

BOX      Sec12
                              Sec12_Q = Sec12_Q - 1
                              WAIT 2
                                dest_loc=58
                                                   1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=58  then
                                                                                                         {FDE_pieces_attr=200
                                                                                                          checker_time_attr=0.4
                                                                             Number_process_sec_array[12]=Number_process_sec_array[12]+1
                                                                                                          }

BOX      Sec13
                              Sec13_Q = Sec13_Q - 1
                              WAIT 2
                                dest_loc=59
                                                   1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=59  then
                                                                                                         {FDE_pieces_attr=200
                                                                                                          checker_time_attr=0.4
                                                                             Number_process_sec_array[13]=Number_process_sec_array[13]+1
                                                                                                          }

BOX      Sec14
                              Sec14_Q = Sec14_Q - 1
                              WAIT 2
                              dest_loc=60          1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=60  then
                                                                                                         {FDE_pieces_attr=200
                                                                                                          checker_time_attr=0.4
                                                                            Number_process_sec_array[14]=Number_process_sec_array[14]+1
                                                                                                          }

BOX      Sec15
                              Sec15_Q = Sec15_Q - 1
                              WAIT 2
                                dest_loc=61        1    BOX      LOC(dest_loc)                  FIRST 1  if dest_loc=61  then
                                                                                                         {FDE_pieces_attr=200
                                                                                                          checker_time_attr=0.4
                                                                           Number_process_sec_array[15]=Number_process_sec_array[15]+1
                                                                                                          }
```

```
BOX      Sec16
                       Sec16_Q = Sec16_Q - 1
                       WAIT 2
                         dest_loc=62       1    BOX      LOC(dest_loc)                 FIRST 1  if dest_loc=62  then
                                                                                                {FDE_pieces_attr=272
                                                                                                 checker_time_attr=0.4
                                                                                  Number_process_sec_array[16]=Number_process_sec_array[16]+1
                                                                                                }

BOX      Sec17
                       Sec17_Q = Sec17_Q - 1
                       WAIT 2
                         dest_loc=63       1    BOX      LOC(dest_loc)                 FIRST 1  if dest_loc=63  then
                                                                                                {FDE_pieces_attr=200
                                                                                                 checker_time_attr=0.4
                                                                                Number_process_sec_array[17]=Number_process_sec_array[17]+ 1
                                                                                                }

BOX      Sec18
                       Sec18_Q = Sec18_Q - 1

                       WAIT 2
                         dest_loc=64       1    BOX      LOC(dest_loc)                 FIRST 1  if dest_loc=64  then
                                                                                                {FDE_pieces_attr=200
                                                                                                 checker_time_attr=0.4
                                                                               Number_process_sec_array[18]=Number_process_sec_array[18]+1
                                                                                                }

BOX      Sec19
                       Sec19_Q = Sec19_Q - 1
                       WAIT 2
                         dest_loc=65       1    BOX      LOC(dest_loc)                 FIRST 1  if dest_loc=65  then
                                                                                                {FDE_pieces_attr=200
                                                                                                 checker_time_attr=0.4
                                                                               Number_process_sec_array[19]=Number_process_sec_array[19]+1
                                                                                                }

BOX      Sec20
                       Sec20_Q = Sec20_Q - 1
                       WAIT 2
                         dest_loc=66       1    BOX      LOC(dest_loc)                 FIRST 1  if dest_loc=66  then
                                                                                                {FDE_pieces_attr=200
                                                                                                 checker_time_attr=0.4
                                                                               Number_process_sec_array[20]=Number_process_sec_array[20]+1
                                                                                                }
      .
      .
      .
BOX      sec1_LP_move   wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.
                                    1             Runouts                        FIRST 1
BOX      sec2_LP_move   wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.

                                     1    BOX      Runouts                        FIRST 1
BOX      sec3_LP_move   wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.

                                     1    BOX      Runouts                        FIRST 1
BOX      sec4_LP_move   wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.

                                     1    BOX      Runouts                        FIRST 1
BOX      sec5_LP_move   wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.
                                     1    BOX      Runouts                        FIRST 1
BOX      sec6_LP_move   wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.

                                      1    BOX      Runouts                        FIRST 1
BOX      sec7_LP_move   wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.
```

```
                                       1    BOX      Runouts                        FIRST 1
BOX      sec8_LP_move   wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.

                                       1    BOX      Runouts                        FIRST 1
BOX      sec9_LP_move   wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.

                                       1    BOX      Runouts                        FIRST 1
BOX      sec10_LP_move  wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.
                                       1    BOX      Runouts                        FIRST 1
BOX      sec11_LP_move  wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.

                                       1    BOX      Runouts                        FIRST 1
BOX      sec12_LP_move  wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.
                                       1    BOX      Runouts                        FIRST 1
BOX      sec13_LP_move  wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.

                                       1    BOX      Runouts                        FIRST 1
BOX      sec14_LP_move  wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                        LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                         //record the number of entities in the Loadposition QUE.
                        MLP_Que_stat                      //Assign the arrays to variables to get stat.
```

```
                                          1    BOX      Runouts                        FIRST 1
 BOX      sec15_LP_move  wait  sec_move[Routing_array[ID_attr,7],1]
                         wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                         LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                          //record the number of entities in the Loadposition QUE.
                         MLP_Que_stat                      //Assign the arrays to variables to get stat.

                                          1    BOX      Runouts                        FIRST 1
 BOX      sec16_LP_move  wait  sec_move[Routing_array[ID_attr,7],1]
                         wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                         LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                          //record the number of entities in the Loadposition QUE.
                         MLP_Que_stat                      //Assign the arrays to variables to get stat.
                                          1    BOX      Runouts                        FIRST 1

 BOX      sec17_LP_move  wait  sec_move[Routing_array[ID_attr,7],1]
                         wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                         LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                          //record the number of entities in the Loadposition QUE.
                         MLP_Que_stat                      //Assign the arrays to variables to get stat.
                                          1    BOX      Runouts                        FIRST 1

 BOX      sec18_LP_move  wait  sec_move[Routing_array[ID_attr,7],1]
                         wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                         LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                          //record the number of entities in the Loadposition QUE.
                         MLP_Que_stat                      //Assign the arrays to variables to get stat.
                                          1    BOX      Runouts                        FIRST 1

 BOX      sec19_LP_move  wait  sec_move[Routing_array[ID_attr,7],1]


                         wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                         LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                          //record the number of entities in the Loadposition QUE.
                         MLP_Que_stat                      //Assign the arrays to variables to get stat.
                                          1    BOX      Runouts                        FIRST 1

BOX      sec20_LP_move  wait  sec_move[Routing_array[ID_attr,7],1]
                        wait until  LP_QUE_array[Routing_array[ID_attr,7]]< LP_capacity_array[Routing_array[ID_attr,5],2]
                         LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]] +1
                                          //record the number of entities in the Loadposition QUE.
                         MLP_Que_stat                      //Assign the arrays to variables to get stat.
                                          1    BOX      Runouts                        FIRST 1
```

```
       .
       .
       .
BOX      Runouts
                       LPbox_order_array[Routing_array[ID_attr,7]]=LPbox_order_array[Routing_array[ID_attr,7]]+1
                       LP_arrive_attr=LPbox_order_array[Routing_array[ID_attr,7]]    //Assign variable as attribute.
                       if State_LP_array[Routing_array[ID_attr,7]]=3 then   //Server is not available. Wait for server.
                          {
                            WAIT UNTIL LP_arrive_attr<LPbox_turn_array[Routing_array[ID_attr,7]]+3
                                        //Wait until it is the box's turn to be processed/
                          }
                        if  State_LP_array[Routing_array[ID_attr,7]]<3 then //Server is available.
                          {
                             LP_QUE_array[Routing_array[ID_attr,7]] = LP_QUE_array[Routing_array[ID_attr,7]]-1
                                                              //Box leaves the QUEUE
                             MLP_Que_stat        //Update the variables.
                             State_LP_array[Routing_array[ID_attr,7]]= State_LP_array[Routing_array[ID_attr,7]]+1
                             WAIT checker_time_attr       //Process the entity.

                                 if LPbox_turn_array[Routing_array[ID_attr,7]] MOD FDE_pieces_attr=0 then
                                  wait swap_time                   //Swap out time
                             State_LP_array[Routing_array[ID_attr,7]]= State_LP_array[Routing_array[ID_attr,7]]-1
                             LPbox_turn_array[Routing_array[ID_attr,7]]=LPbox_turn_array[Routing_array[ID_attr,7]]+1
                                         //records the number of boxes processed+1
                           }
                       EXITS = EXITS + 1


                                            1    BOX      EXIT                           FIRST 1
EntA     Dummy_utl_loc1 wait 10
                       i=1
                        while i<20+
                         Do
                           {
                             Sec_UTL_array[i]=Number_process_sec_array[i]*100/(SEC_target_cap_array[i,2]*clock(HR))
                             i=i+1
                           }
                       Sec1_UTL = Sec_UTL_array[1]
                       Sec2_UTL = Sec_UTL_array[2]
                       Sec3_UTL = Sec_UTL_array[3]
                       Sec4_UTL = Sec_UTL_array[4]
                       Sec5_UTL = Sec_UTL_array[5]
                       Sec6_UTL = Sec_UTL_array[6]
                       Sec7_UTL = Sec_UTL_array[7]
                       Sec8_UTL = Sec_UTL_array[8]
                       Sec9_UTL = Sec_UTL_array[9]
```

```
                         Sec10_UTL = Sec_UTL_array[10]
                         Sec11_UTL = Sec_UTL_array[11]
                         Sec12_UTL = Sec_UTL_array[12]
                         Sec13_UTL = Sec_UTL_array[13]
                         Sec14_UTL = Sec_UTL_array[14]
                         Sec15_UTL = Sec_UTL_array[15]
                         Sec16_UTL = Sec_UTL_array[16]
                         Sec17_UTL = Sec_UTL_array[17]
                         Sec18_UTL = Sec_UTL_array[18]
                         Sec19_UTL = Sec_UTL_array[19]
                         Sec20_UTL = Sec_UTL_array[20]
                              .
                              .
                              .
                                         1    EntA     Dummy_utl_loc2                FIRST 1
 EntA     Dummy_utl_loc2                 1             Dummy_utl_loc1                FIRST 1


********************************************************************************
*                                    Arrivals                                  *
********************************************************************************

  Entity   Location       Qty each   First Time Occurrences Frequency                                  Logic
  -------- -------------- ---------- ---------- ----------- ------------------------------------------ ------------
  BOX      PRIMARY        1          0          inf         10/(Total_arrivals[TRUNC(CLOCK(MIN)/10)+1]) Randm = RAND(100)     //Assign
a random number between 0 and 100
                                                                                                        ID=0
                                                                                                        cum=0
                                                                                                         WHILE cum<Randm
                                                                                                         Do
                                                                                                         {
                                                                                                          ID=ID+1

cum=cum+(ID_assign_array[ID,TRUNC(CLOCK(MIN)/10)+3]*100/Total_arrivals[TRUNC(CLOCK(MIN)/10)+1])
                                                                                                         }
                                                                                                        ID_attr=ID_assign_array[ID,1]
//Assign the ID attribute for each box.
                                                                                                        Box_id = ID_attr      //The ID
of each arriving  box.
                                                                                                        ARRIVALS = ARRIVALS+1



  EntA     Dummy_utl_loc1 1          0          1           inf
```

```
********************************************************************************
*                                  Attributes                                  *
********************************************************************************

  ID                Type         Classification
  ----------------- ------------ --------------
  ID_attr           Integer      Entity
  checker_time_attr Real         Entity
  FDE_pieces_attr   Integer      Entity
  LP_arrive_attr    Integer      Entity


********************************************************************************
*                              Variables (global)                              *
********************************************************************************

  ID         Type         Initial value Stats
  ---------- ------------ ------------- -----------
#
#Stores random number between 0 and 100
  Randm      Real         0             None
#
#This value between 1 and 10 is assigned as attribute
  ID         Integer      0             None
  k          Integer      0             None
#
#Loop counter
  j          Integer      0             None
#
#Que at Loadposition
  LPa_que    Integer      0             Time Series
#
#Que at Loadposition
  LPb_que    Integer      0             Time Series
#
#Que at Loadposition
  LPc_que    Integer      0             Time Series
#
#Que at Loadposition
  LPd_que    Integer      0             Time Series
#
#Que at Loadposition
  LPe_que    Integer      0             Time Series
#
#Que at Loadposition
  LPf_que    Integer      0             Time Series
  LPg_que    Integer      0             Time Series
```

LPh_que	Integer	0	Time Series
LPi_que	Integer	0	Time Series
LPj_que	Integer	0	Time Series
LPk_que	Integer	0	Time Series
LPl_que	Integer	0	Time Series
LPm_que	Integer	0	Time Series
LPn_que	Integer	0	Time Series
LPo_que	Integer	0	Time Series
LPp_que	Integer	0	Time Series
LPq_que	Integer	0	Time Series
LPr_que	Integer	0	Time Series
LPs_que	Integer	0	Time Series
LPt_que	Integer	0	Time Series
LPu_que	Integer	0	Time Series
LPv_que	Integer	0	Time Series
LPw_que	Integer	0	Time Series
LPx_que	Integer	0	Time Series
LPy_que	Integer	0	Time Series
LPz_que	Integer	0	Time Series
LPaa_que	Integer	0	Time Series
LPab_que	Integer	0	Time Series
LPac_que	Integer	0	Time Series
LPad_que	Integer	0	Time Series
LPae_que	Integer	0	Time Series
LPaf_que	Integer	0	Time Series
LPag_que	Integer	0	Time Series
LPah_que	Integer	0	Time Series
LPai_que	Integer	0	Time Series
LPaj_que	Integer	0	Time Series
LPak_que	Integer	0	Time Series
LPal_que	Integer	0	Time Series
LPam_que	Integer	0	Time Series
LPan_que	Integer	0	Time Series
LPao_que	Integer	0	Time Series
LPap_que	Integer	0	Time Series
LPaq_que	Integer	0	Time Series
LPar_que	Integer	0	Time Series
LPas_que	Integer	0	Time Series
LPat_que	Integer	0	Time Series
LPau_que	Integer	0	Time Series
LPav_que	Integer	0	Time Series
LPaw_que	Integer	0	Time Series
LPax_que	Integer	0	Time Series
LPay_que	Integer	0	Time Series
LPaz_que	Integer	0	Time Series
LPba_que	Integer	0	Time Series
LPbb_que	Integer	0	Time Series

LPbc_que	Integer	0	Time Series
LPbd_que	Integer	0	Time Series
LPbe_que	Integer	0	Time Series
LPbf_que	Integer	0	Time Series
LPbg_que	Integer	0	Time Series
LPbh_que	Integer	0	Time Series
LPbi_que	Integer	0	Time Series
LPbj_que	Integer	0	Time Series
LPbk_que	Integer	0	Time Series
LPbl_que	Integer	0	Time Series
LPbm_que	Integer	0	Time Series
LPbn_que	Integer	0	Time Series
LPbo_que	Integer	0	Time Series
LPbp_que	Integer	0	Time Series
LPbq_que	Integer	0	Time Series
LPbr_que	Integer	0	Time Series
LPbs_que	Integer	0	Time Series
LPbt_que	Integer	0	Time Series
LPbu_que	Integer	0	Time Series
LPbv_que	Integer	0	Time Series
LPbw_que	Integer	0	Time Series
LPbx_que	Integer	0	Time Series
LPby_que	Integer	0	Time Series
LPbz_que	Integer	0	Time Series
LPca_que	Integer	0	Time Series
LPcb_que	Integer	0	Time Series
LPcc_que	Integer	0	Time Series
LPcd_que	Integer	0	Time Series
LPce_que	Integer	0	Time Series
LPcf_que	Integer	0	Time Series
LPcg_que	Integer	0	Time Series
LPch_que	Integer	0	Time Series
LPci_que	Integer	0	Time Series
LPcj_que	Integer	0	Time Series
LPck_que	Integer	0	Time Series
LPcl_que	Integer	0	Time Series
LPcm_que	Integer	0	Time Series
LPcn_que	Integer	0	Time Series
LPco_que	Integer	0	Time Series
LPcp_que	Integer	0	Time Series
LPcq_que	Integer	0	Time Series
LPcr_que	Integer	0	Time Series
LPcs_que	Integer	0	Time Series
LPct_que	Integer	0	Time Series
LPcu_que	Integer	0	Time Series
LPcv_que	Integer	0	Time Series
LPcw_que	Integer	0	Time Series

```
  LPcx_que   Integer      0             Time Series
  LPcy_que   Integer      0             Time Series
  LPcz_que   Integer      0             Time Series
  LPda_que   Integer      0             Time Series
  LPdb_que   Integer      0             Time Series
  LPdc_que   Integer      0             Time Series
  LPdd_que   Integer      0             Time Series
  LPde_que   Integer      0             Time Series
  LPdf_que   Integer      0             Time Series
  LPdg_que   Integer      0             Time Series
  LPdh_que   Integer      0             Time Series
  LPdi_que   Integer      0             Time Series
  LPdj_que   Integer      0             Time Series
  LPdk_que   Integer      0             Time Series
  LPdl_que   Integer      0             Time Series
  LPdm_que   Integer      0             Time Series
  LPdn_que   Integer      0             Time Series
  LPdo_que   Integer      0             Time Series
  LPdp_que   Integer      0             Time Series
  LPdq_que   Integer      0             Time Series
  LPdr_que   Integer      0             Time Series
  LPds_que   Integer      0             Time Series
  LPdt_que   Integer      0             Time Series
  LPdu_que   Integer      0             Time Series
  LPdv_que   Integer      0             Time Series
  LPdw_que   Integer      0             Time Series
  LPdx_que   Integer      0             Time Series
  LPdy_que   Integer      0             Time Series
  LPdz_que   Integer      0             Time Series
#
#The ID of each arriving box.
  Box_id     Integer      0             None
#
#Loop counter
  i          Integer      1             None
#
#Cumulative percentage of boxes.
  cum        Real         0             None
#
#Total arrivals
  ARRIVALS   Integer      0             Time Series
#
#Total exit after processing
  EXITS      Integer      0             Time Series
  Sec1_Q     Integer      0             Time Series
  Sec2_Q     Integer      0             Time Series
  Sec3_Q     Integer      0             Time Series
```

```
Sec4_Q     Integer      0             Time Series
Sec5_Q     Integer      0             Time Series
Sec6_Q     Integer      0             Time Series
Sec7_Q     Integer      0             Time Series
Sec8_Q     Integer      0             Time Series
Sec9_Q     Integer      0             Time Series
Sec10_Q    Integer      0             Time Series
Sec11_Q    Integer      0             Time Series
Sec12_Q    Integer      0             Time Series
Sec13_Q    Integer      0             Time Series
Sec14_Q    Integer      0             Time Series
Sec15_Q    Integer      0             Time Series
Sec16_Q    Integer      0             Time Series
Sec17_Q    Integer      0             Time Series
Sec18_Q    Integer      0             Time Series
Sec19_Q    Integer      0             Time Series
Sec20_Q    Integer      0             Time Series
      .
      .
      .
Sec1_UTL   Real         0             Time Series
Sec2_UTL   Real         0             Time Series
Sec3_UTL   Integer      0             Time Series
Sec4_UTL   Integer      0             Time Series
Sec5_UTL   Integer      0             Time Series
Sec6_UTL   Integer      0             Time Series
Sec7_UTL   Integer      0             Time Series
Sec8_UTL   Integer      0             Time Series
Sec9_UTL   Integer      0             Time Series
Sec10_UTL  Integer      0             Time Series
Sec11_UTL  Integer      0             Time Series
Sec12_UTL  Integer      0             Time Series
Sec13_UTL  Integer      0             Time Series
Sec14_UTL  Integer      0             Time Series
Sec15_UTL  Integer      0             Time Series
Sec16_UTL  Integer      0             Time Series
Sec17_UTL  Integer      0             Time Series
Sec18_UTL  Integer      0             Time Series
Sec19_UTL  Integer      0             Time Series
Sec20_UTL  Integer      0             Time Series
      .
      .
      .
dest_loc   Integer      0             Time Series
swap_time  Real         3.0           None
```

```
********************************************************************************
*                                    Arrays                                    *
********************************************************************************

  ID                       Dimensions   Type
  ------------------------ ------------ ------------
  ID_assign_array          2800,38      Integer
  State_LP_array           1200         Integer
  LPbox_turn_array         1200         Integer
  LPbox_order_array        1200         Integer
  LP_QUE_array             1200         Integer
  Routing_array            2800,8       Integer
  LP_capacity_array        1800,6       Integer
  SEC_target_cap_array     25,2         Real
  Sec_UTL_array            25           Real
  Number_process_sec_array 25           Integer
  LP_UTL_array             1200         Real
  LP_target_cap_array      1200         Real
  LP_runout_array          1200,100     Integer
  primary_move             25,5         Real
  sec_move                 1200,4       Real
  process_sec_array        30           Integer
  Total_arrivals           30           Integer
  LP_stats                 200          Integer


********************************************************************************
*                                    Macros                                    *
********************************************************************************

  ID                 Text
  ------------------ ------------
  MLP_Que_stat

                     LPa_que=LP_QUE_array[LP_stats[1]]
                     LPb_que=LP_QUE_array[LP_stats[2]]
                     LPc_que=LP_QUE_array[LP_stats[3]]
                     LPd_que=LP_QUE_array[LP_stats[4]]
                     LPe_que=LP_QUE_array[LP_stats[5]]
                     LPf_que=LP_QUE_array[LP_stats[6]]
                     LPg_que=LP_QUE_array[LP_stats[7]]
                     LPh_que=LP_QUE_array[LP_stats[8]]
                     LPi_que=LP_QUE_array[LP_stats[9]]
                     LPj_que=LP_QUE_array[LP_stats[10]]
                     LPk_que=LP_QUE_array[LP_stats[11]]
                     LPl_que=LP_QUE_array[LP_stats[12]]
                     LPm_que=LP_QUE_array[LP_stats[13]]
```

```
LPn_que=LP_QUE_array[LP_stats[14]]
LPo_que=LP_QUE_array[LP_stats[15]]
LPp_que=LP_QUE_array[LP_stats[16]]
LPq_que=LP_QUE_array[LP_stats[17]]
LPr_que=LP_QUE_array[LP_stats[18]]
LPs_que=LP_QUE_array[LP_stats[19]]
LPt_que=LP_QUE_array[LP_stats[20]]
LPu_que=LP_QUE_array[LP_stats[21]]
LPv_que=LP_QUE_array[LP_stats[22]]
LPw_que=LP_QUE_array[LP_stats[23]]
LPx_que=LP_QUE_array[LP_stats[24]]
LPy_que=LP_QUE_array[LP_stats[25]]
LPz_que=LP_QUE_array[LP_stats[26]]
LPaa_que=LP_QUE_array[LP_stats[27]]
LPab_que=LP_QUE_array[LP_stats[28]]
LPac_que=LP_QUE_array[LP_stats[29]]
LPad_que=LP_QUE_array[LP_stats[30]]
LPae_que=LP_QUE_array[LP_stats[31]]
LPaf_que=LP_QUE_array[LP_stats[32]]
LPag_que=LP_QUE_array[LP_stats[33]]
LPah_que=LP_QUE_array[LP_stats[34]]
LPai_que=LP_QUE_array[LP_stats[35]]
LPaj_que=LP_QUE_array[LP_stats[36]]
LPak_que=LP_QUE_array[LP_stats[37]]
LPal_que=LP_QUE_array[LP_stats[38]]
LPam_que=LP_QUE_array[LP_stats[39]]
LPan_que=LP_QUE_array[LP_stats[40]]
LPao_que=LP_QUE_array[LP_stats[41]]
LPap_que=LP_QUE_array[LP_stats[42]]
LPaq_que=LP_QUE_array[LP_stats[43]]
LPar_que=LP_QUE_array[LP_stats[44]]
LPas_que=LP_QUE_array[LP_stats[45]]
LPat_que=LP_QUE_array[LP_stats[46]]
LPau_que=LP_QUE_array[LP_stats[47]]
LPav_que=LP_QUE_array[LP_stats[48]]
LPaw_que=LP_QUE_array[LP_stats[49]]
LPax_que=LP_QUE_array[LP_stats[50]]
LPay_que=LP_QUE_array[LP_stats[51]]
LPaz_que=LP_QUE_array[LP_stats[52]]
LPba_que=LP_QUE_array[LP_stats[53]]
LPbb_que=LP_QUE_array[LP_stats[54]]
LPbc_que=LP_QUE_array[LP_stats[55]]
LPbd_que=LP_QUE_array[LP_stats[56]]
LPbe_que=LP_QUE_array[LP_stats[57]]
LPbf_que=LP_QUE_array[LP_stats[58]]
LPbg_que=LP_QUE_array[LP_stats[59]]
LPbh_que=LP_QUE_array[LP_stats[60]]
```

```
LPbi_que=LP_QUE_array[LP_stats[61]]
LPbj_que=LP_QUE_array[LP_stats[62]]
LPbk_que=LP_QUE_array[LP_stats[63]]
LPbl_que=LP_QUE_array[LP_stats[64]]
LPbm_que=LP_QUE_array[LP_stats[65]]
LPbn_que=LP_QUE_array[LP_stats[66]]
LPbo_que=LP_QUE_array[LP_stats[67]]
LPbp_que=LP_QUE_array[LP_stats[68]]
LPbq_que=LP_QUE_array[LP_stats[69]]
LPbr_que=LP_QUE_array[LP_stats[70]]
LPbs_que=LP_QUE_array[LP_stats[71]]
LPbt_que=LP_QUE_array[LP_stats[72]]
LPbu_que=LP_QUE_array[LP_stats[73]]
LPbv_que=LP_QUE_array[LP_stats[74]]
LPbw_que=LP_QUE_array[LP_stats[75]]
LPbx_que=LP_QUE_array[LP_stats[76]]
LPby_que=LP_QUE_array[LP_stats[77]]
LPbz_que=LP_QUE_array[LP_stats[78]]
LPca_que=LP_QUE_array[LP_stats[79]]
LPcb_que=LP_QUE_array[LP_stats[80]]
LPcc_que=LP_QUE_array[LP_stats[81]]
LPcd_que=LP_QUE_array[LP_stats[82]]
LPce_que=LP_QUE_array[LP_stats[83]]
LPcf_que=LP_QUE_array[LP_stats[84]]
LPcg_que=LP_QUE_array[LP_stats[85]]
LPch_que=LP_QUE_array[LP_stats[86]]
LPci_que=LP_QUE_array[LP_stats[87]]
LPcj_que=LP_QUE_array[LP_stats[88]]
LPck_que=LP_QUE_array[LP_stats[89]]
LPcl_que=LP_QUE_array[LP_stats[90]]
LPcm_que=LP_QUE_array[LP_stats[91]]
LPcn_que=LP_QUE_array[LP_stats[92]]
LPco_que=LP_QUE_array[LP_stats[93]]
LPcp_que=LP_QUE_array[LP_stats[94]]
LPcq_que=LP_QUE_array[LP_stats[95]]
LPcr_que=LP_QUE_array[LP_stats[96]]
LPcs_que=LP_QUE_array[LP_stats[97]]
LPct_que=LP_QUE_array[LP_stats[98]]
LPcu_que=LP_QUE_array[LP_stats[99]]
LPcv_que=LP_QUE_array[LP_stats[100]]
LPcw_que=LP_QUE_array[LP_stats[101]]
LPcx_que=LP_QUE_array[LP_stats[102]]
LPcy_que=LP_QUE_array[LP_stats[103]]
LPcz_que=LP_QUE_array[LP_stats[104]]
LPda_que=LP_QUE_array[LP_stats[105]]
LPdb_que=LP_QUE_array[LP_stats[106]]
LPdc_que=LP_QUE_array[LP_stats[107]]
```

```
                   LPdd_que=LP_QUE_array[LP_stats[108]]
                   LPde_que=LP_QUE_array[LP_stats[109]]
                   LPdf_que=LP_QUE_array[LP_stats[110]]
                   LPdg_que=LP_QUE_array[LP_stats[111]]
                   LPdh_que=LP_QUE_array[LP_stats[112]]
                   LPdi_que=LP_QUE_array[LP_stats[113]]
                   LPdj_que=LP_QUE_array[LP_stats[114]]
                   LPdk_que=LP_QUE_array[LP_stats[115]]
                   LPdl_que=LP_QUE_array[LP_stats[116]]
                   LPdm_que=LP_QUE_array[LP_stats[117]]
                   LPdn_que=LP_QUE_array[LP_stats[118]]
                   LPdo_que=LP_QUE_array[LP_stats[119]]
                   LPdp_que=LP_QUE_array[LP_stats[120]]
                   LPdq_que=LP_QUE_array[LP_stats[121]]
                   LPdr_que=LP_QUE_array[LP_stats[122]]
                   LPds_que=LP_QUE_array[LP_stats[123]]
                   LPdt_que=LP_QUE_array[LP_stats[124]]
                   LPdu_que=LP_QUE_array[LP_stats[125]]
                   LPdv_que=LP_QUE_array[LP_stats[126]]
                   LPdw_que=LP_QUE_array[LP_stats[127]]
                   LPdx_que=LP_QUE_array[LP_stats[128]]
                   LPdy_que=LP_QUE_array[LP_stats[129]]
                   LPdz_que=LP_QUE_array[LP_stats[130]]
MLP_cap            1
sec1_sorters       12
sec2_sorters       12
sec3_sorters       12
sec4_sorters       12
sec5_sorters       12
sec6_sorters       12
sec7_sorters       12
sec8_sorters       12
sec9_sorters       12
sec10_sorters      12
sec11_sorters      12
sec12_sorters      12
sec13_sorters      12
sec14_sorters      12
sec15_sorters      12
sec16_sorters      12
sec17_sorters      12
sec18_sorters      12
sec19_sorters      12
sec20_sorters      12
      .
      .
      .
```