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 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

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.

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 to various secondary sort areas, run-out belts, and load-positions. The spreadsheet

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 E Tc-rrri

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					

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

Table 3.

```
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.
```

Step 6: 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

if State_LP_array[Routing_array[ID_a

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

(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 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

Another approach for controlling the number of replications used to estimate the

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  LW1                 j=2
```

```
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
eO                       f3]3T*( eO 9 BOr=          5a5-)k1                                                      +1
                                                                                          }


                          dest_loc=49
                                    1    BOX      LOC                                                            +1
                                   FDE_pieces_attr=200


                          dest_loc=49
                                    1    BOX      LOC                                                            +1
                                                                                          }
```

```
                                                  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
```

```
                         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     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.
```

```
        .
        .
        .
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                   3            8aaaaaai        wap_time                   3          8aaaaaai
```

```
                         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_attr    ID_attr    ID_attr    ID
                                                                              TRUNC(CLOCK(MIN)/10)+1])
                                                                              TRUNC(CLOCK(MIN)/10)+1])
```

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]]
```

```
                   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
      .
      .
      .
```