

# Evaluating Planned Capacities for Public Health Emergency Supply Chain Models

Kathleen King and Jack Muckstadt  
School of Operations Research and Information Engineering  
Cornell University

Technical Report No. 1475

September 15, 2009

## Abstract

To respond effectively in the event of a public health emergency, careful analysis of preparedness plans is essential. This note describes a linear programming model of the Center for Disease Control Strategic National Stockpile (SNS) supply chain for distributing medical supplies under emergency conditions. The purpose of the model is to evaluate the effects that a specified collection of planned resource availabilities would have on the efficacy of the distribution network. This efficacy is measured in terms of the number of patient-hours waited during the emergency response time horizon. The model's constraints describe the physical system, including the relationships between the SNS, the regional distribution warehouse, and the medical supply distribution clinics. In the following note all model constraints and variables are described in detail, and AMPL code implementing the model is given.

## 1 Introduction

In the event of a large-scale public health emergency, like an anthrax attack or an influenza pandemic, a supply chain distribution network must provide the necessary medications and supplies to all affected individuals in a timely manner. This network consists of the federally-controlled Strategic National Stockpile (SNS), state or locally run Receiving Storing and Staging (RSS) warehouses, and local Points of Dispensing (PODs) or other dispensing agencies. For this system to be effective, detailed plans must be made in advance. Furthermore, policy makers must have clear ideas about how material will flow through the system and how the planned capacities will affect this flow and meet the needs of the public. In short, we must ask whether the planned distribution network will function as desired.

Our goal is to construct a simple method for obtaining an approximate answer to this question. In this note, we describe a model that minimizes the delay in service to the affected public while recognizing the physical constraints of the distribution network. While the scope of the model we present in this note addresses all elements of the supply chain, smaller portions of the system could be studied in greater detail using similar models. This is essential because the question of “does it work?” will mean different things to different people working within the network. An RSS manager, for instance, likely does not care about how the SNS moves material to the RSS, as long as it arrives on time, nor does he worry about the distribution of staff at individual PODs. However, the RSS manager must ensure that he has enough staff working at the unloading docks to receive material, space to store the material, and the appropriate number of material handling devices necessary to process the material and move it to its assigned

locations. He may be concerned with different types of workers who are certified to perform different tasks. A POD manager, while uninterested in the detailed operations of the RSS, may instead be concerned with the movement of patients and inventory through a POD, while the truck dispatcher may want to focus on the flow of trucks through the system. It is essential to develop models that focus on different areas of interest to particular users. Again, these models need not be overly complex in order to address the key questions. They would be quite similar to the model presented here in structure and solution methodology, but many details given here will be removed and others added to tailor the new models to their particular applications.

As mentioned above, the model that we will describe here ranges across all elements of the supply chain. The model is rate-based and uses rates such as “the number of patients processed per hour per POD” or “the number of trucks that can be unloaded at the RSS during each time period.” These rates are not calculated by the model, but must be determined separately. For example, one might use a simulation tool such as D-PODS [4], Clinic Generator [1], or ReaOpt [5] to estimate staffing requirements consistent with POD processing rates. Given these effective capacities, the proposed model can help planners to understand more clearly how well their system might perform. Specifically, it will identify bottlenecks in the network, that is, capacities at various points in time that delay the delivery of required material.

For the purpose of evaluating the efficacy of existing plans, the simple model we will describe is based on the assumption that all rates and distribution system parameters are known with certainty. In more complex models the impact of uncertainty on these rates and parameter values is considered explicitly. However, the focus of this model is simply to determine whether, given a fixed set of capacities, our emergency response network will be effective. Thus, for now we assume that patient arrival rates, lead times, and inventory shipments to the SNS are assumed to be known. Below we describe the system model and its assumptions and we explain how the “best” performance is determined. We will then present the mathematical model.

## 2 Model Description

In this model, time is broken down into discrete “periods.” Decisions are assumed to be made at the beginning of each time period. Capacity levels are assumed to be constant during a single time period, but they may change between time periods. The length of a period will depend on the type of emergency. For example, in the case of an anthrax attack, we may want time periods to be an hour long, or perhaps even 15 minutes in length. In responding to pandemic influenza, however, we may want to consider time periods of 8 hours (the standard working shift length) or even days. Shorter time periods allow for more detailed modeling, but they also require more input data from the user.

The model that we will describe consists of the SNS, one RSS, and a set of PODs served by the RSS. We assume that each location may have an initial stock of inventory, and as time goes by, external suppliers resupply the SNS, while the SNS supplies the RSS and the RSS supplies the PODs. It would be easy to add more RSSs to the system, or even other SNS warehouses, but doing so would further complicate our already crowded notation, so we focus on a single RSS for simplicity. In this model, we will consider the following capacities:

- the rates at which patients arrive to each POD,
- the rates at which patients can be served at each POD,
- the rate at which cases of inventory can be loaded onto trucks at the RSS,
- the rate at which pallets can be loaded onto trucks at the SNS,
- the rate at which pallets can be unloaded from trucks at the RSS,
- the number of trucks available to transport inventory from the SNS to the RSS,

- the number of trucks available to transport inventory from the RSS to the PODs, and
- the amount of usable storage space (by volume) at the RSS and PODs.

Let us begin now by describing the constraints on the SNS and its shipments. For simplicity, we assume that trucks are used to transport material from the SNS to the RSS, although the use of aircraft is also an option. The shipping capacity of the SNS determines the maximum number of pallets that can be sent to the RSS each day. As mentioned earlier, we assume that all of these data are known in advance. However, we assume that the SNS has enough capacity to receive all arriving inventory immediately, so arriving delivery trucks to the SNS are not considered in the present model.

A fixed number of trucks is available to move inventory between the SNS and RSS; the total number available in the current time period is given by the total number of trucks in the SNS-RSS system less those that are currently en route to or from the SNS. Under a third party carrier, the number of trucks available to move inventory will almost certainly be more than sufficient, but we include the constraint to emphasize the importance of fully assessing available resources.

Trucks are assumed to return to the SNS after being unloaded at the RSS. The time required to travel between the RSS and the SNS is assumed known and includes the time required for unloading. We assume that it is extremely undesirable to make trucks wait to be unloaded at the RSS; hence, trucks are not sent to the RSS from the SNS unless it is known that the RSS will be able to receive them during the time period of their arrival. In a nondeterministic setting, where travel times and the RSS unloading capacities are not known precisely, we would relax such an assumption; but, in the deterministic model, this requirement is achievable and does not change the optimal solution.

Each truck has a fixed volume capacity, in terms of pallets, so the total volume of the inventory loaded onto each truck must not exceed this value. And, of course, inventory cannot be shipped out to the RSS until it has arrived to the SNS. The SNS has some initial supply of inventory, and additional materials arrive to the SNS from external sources over time. All inventory at the SNS is managed in terms of pallets. Note that there may be many different inventory types, and the number of units per case, the size of each case, and the number of cases per pallet may vary by type.

At the RSS, there is a limit on the number of pallets of inventory that can be received during each time period, and there is a fixed amount of usable warehouse space available for storing inventory. Inventory cannot be unloaded and stored in the warehouse unless space is available. Inventory is sent to PODs in units of cases, and there is a limit on the rate at which cases can be loaded onto trucks to be sent to PODs. Depending on the RSS organization, more inventory storage constraints may be required. For example, there may be constraints on where, within the warehouse, certain types of inventory may be stored. However, for the illustrative purposes of this model, we do not include these details.

A fixed number of trucks is available to move inventory to the PODs. These trucks travel on routes that include one or more PODs. For now we assume that each POD lies on exactly one route, but this assumption could be relaxed easily if it would be useful to allow some PODs to lie on multiple routes. The PODs along a route are always visited in the same order, so the lead time to reach each POD is known, as well as the total route time. We assume that these times include the time required to unload material at the PODs. We do not limit the rates at which inventory can be received at each POD because we assume that these values will normally be somewhat small and so the task of unloading material will not require significant time or resources at the PODs. This capacity could be restricted if necessary. However, the model will indicate how much capacity is needed and when to unload the incoming trucks.

At the PODs, the patient arrival rate may change over time, but within a single time period the rate is assumed to be constant. Similarly, the rate at which a POD can serve patients may change over time, but is constant during each individual time period. So, if we used short time periods, then we could model very detailed patient arrival patterns and changing POD service capacities.

The model recognizes that patients can be served only after they have arrived and only when the POD has available service capacity and sufficient inventory. In the model, arriving patients who cannot be served immediately wait until the POD has service capacity and inventory available to process them. Different patient types may require different types of medical supplies. The amount of inventory used for each patient may also change over time; for example, during the first days of an emergency response to an anthrax attack, when inventory might be severely limited, it may be desirable to distribute unit-of-use bottles containing only 10-day pill regimens. Once more inventory has arrived, patients would return to PODs to receive the remainder of their prescribed regimens.

### 3 Model Objective

To determine the optimal performance of this supply chain, we first must decide what we mean by “optimal.” The most important goal of the system is to serve as many people as possible, as quickly as possible. Therefore, the model objective is to minimize the number of person-hours of delay experienced by patients who arrive to PODs seeking service. If desired, we could weight patients by type, by location, or by time period. For example, during an influenza pandemic, it may be more important to serve small children and immunocompromised individuals before the rest of the population. During an anthrax attack, there may be a larger penalty for making patients wait for antibiotics near the end of the time horizon, when patients have a higher chance of becoming symptomatic. Thus, we will assign a cost for each patient who is waiting at a POD for service at the end of each time period, and we will minimize the sum of these costs. Note that these “costs” are not monetary, but rather parameters that can be tuned to measure the effects of different strategies for serving incoming streams of patients.

The objective we have described is an efficacy-based goal. Another possible objective would be to consider balancing efficacy and efficiency, by calculating the operating expenses of the system as well as the penalties for patient delays. In such a case, we could assign monetary costs to the SNS and RSS trucks used, to the inventory stored at the SNS, RSS, and PODs, and to the capacity at each location over time. These costs could be calculated using a spreadsheet tool such as RASCAL [3]. The goal of the model would then be to minimize the sum of all of these costs in addition to the penalty costs for patient delays. Alternatively, we could consider a model in which prescribed service constraints must be met, and our objective would be to achieve the desired level of service with the lowest possible cost. To implement such a model, we would again assign monetary costs to each type of capacity and then minimize the total cost. That is, the level of desired efficacy would be given, and the model would have an efficiency-based objective. Such a model could be implemented in a manner similar to the one discussed below.

### 4 Linear Programming Model

The problem that we have described above can be written as a linear program [2]. That is, we can define a linear objective function and linear constraints which enforce the conditions described in earlier sections. We now define in detail the variables, parameters, and constraints

described above. The values of the variables will be determined by the linear program, but all of the parameter values must be calculated or estimated by the POD designers.

## 4.1 Definition of Variables

### Sets

Note that the elements of set  $\mathcal{N}$  are indexed by  $n$ , the elements of  $\mathcal{T}$  are indexed by  $t$ , and so forth. We let

$\mathcal{N}$  = the set of PODs;

$\mathcal{T}$  = the set of time periods;

$\mathcal{K}$  = the set of inventory types; (These might include such items as unit-of-use bottles of doxycycline and ciprofloxacin in response to an anthrax attack. In an influenza pandemic, we may be concerned with antivirals, masks, and N-95 respirators.)

$\mathcal{R}$  = the set of truck routes; and (Each route consists of an ordered sequence of PODs from the set  $\mathcal{N}$ .)

$\mathcal{P}$  = the set of patient types. (Patient types might include such categories as children, mobility-impaired adults, or symptomatic individuals. Different patient types may require different resources and arrive at different rates over time.)

### Decision Variables

We let

$s_{npt}$  = number of patients of type  $p$  at POD  $n$  who are served during time period  $t$ ;

$q_{kt}^p$  = number of pallets of inventory type  $k$  sent to the RSS during time period  $t$ ;

$q_{knt}^c$  = number of cases of inventory type  $k$  sent to POD  $n$  during time period  $t$ ;

$u_t^S$  = number of trucks dispatched from the SNS to the RSS during time period  $t$ ;

$u_{rt}$  = number of trucks dispatched from the RSS to PODS on route  $r$  during time period  $t$ ;

$v_{kt}$  = number of pallets of inventory type  $k$  loaded at the SNS in time period  $t$ ; and

$w_t$  = number of trucks that are loaded at the SNS in time period  $t$ .

### State Variables

We let

$I_{kt}^S$  = inventory of type  $k$  on-hand at the SNS at the beginning of time period  $t$ ;

$I_{kt}^R$  = inventory of type  $k$  on-hand at the RSS at the beginning of time period  $t$ ;

$I_{knt}^P$  = inventory of type  $k$  on-hand at POD  $n$  at the beginning of time period  $t$ ;

$U_t^S$  = number of trucks available to be used at the SNS at the beginning of time period  $t$ ;

$U_t^R$  = number of trucks available to be used at the RSS at the beginning of time period  $t$ ;

$P_{npt}$  = number of patients of type  $p$  at POD  $n$  waiting to be served at the beginning of time period  $t$ ;

$V_t$  = number of pallets already loaded and ready to send out from the SNS at the beginning of time period  $t$ ; and

$W_t$  = number of trucks already loaded and ready to send out from the SNS at the beginning of time period  $t$ .

### Model Parameters

We let

$D_{npt}$  = number of patients of type  $p$  who arrive to POD  $n$  for service in time period  $t$ ;

$Q_{kt}$  = number of pallets of inventory type  $k$  that arrive to the SNS in time period  $t$ ;

$\Delta_t^S$  = number of trucks that are removed from or added to service at the SNS in time period  $t$ ;

$\Delta_t^R$  = number of trucks that are removed from or added to service at the RSS in time period  $t$ ;

- $\xi^S$  = initial number of trucks available to ship between the SNS and RSS;
- $\xi^R$  = initial number of trucks available to ship from the RSS to the PODs;
- $\zeta_k^S$  = initial inventory of type  $k$  at the SNS;
- $\zeta_k^R$  = initial inventory of type  $k$  at the RSS;
- $\zeta_{kn}$  = initial inventory of type  $k$  at POD  $n$ ;
- $\gamma^R$  = volume of storage space available at the RSS;
- $\gamma_n^P$  = volume of storage space available at POD  $n$ ;
- $\tau^S$  = travel time from the SNS to the RSS;
- $\tau_n$  = lead time to reach POD  $n$  from the RSS;
- $\tau_r^R$  = total time to travel route  $r$ ;
- $\sigma_t^S$  = maximum number of pallets that can be loaded onto trucks in time period  $t$  at the SNS;
- $\sigma_t^{RA}$  = maximum number of pallets that can be unloaded from arriving SNS delivery trucks in time period  $t$  at the RSS;
- $\sigma_t^{RD}$  = maximum number of cases that can be loaded onto departing POD delivery trucks in time period  $t$  at the RSS;
- $\eta_k^p$  = units of inventory type  $k$  in a pallet;
- $\eta_k^c$  = units of inventory type  $k$  in a case;
- $\beta_k^v$  = volume of a unit of inventory type  $k$ ;
- $\rho^{vS}$  = volume capacity of a truck that ships inventory from the SNS to the RSS;
- $\rho^{vR}$  = volume capacity of a truck that ships inventory from the RSS to the PODs;
- $\alpha_{nt}$  = total POD service capacity available;
- $\phi_{kpt}$  = units of resource type  $k$  needed to treat a patient of type  $p$  in time period  $t$ ; and
- $c_{npt}$  = penalty for making one patient of type  $p$  wait at POD  $n$  during time period  $t$ .

## 4.2 Constraints

Using the notation defined above, we will now define the mathematical constraints corresponding to the model described in section 2.

1. Inventory at the SNS in period  $t+1$  is equal to the inventory from the previous time period plus inventory received, minus inventory loaded onto trucks to go to the RSS:

$$I_{k,t+1}^S = I_{kt}^S + \eta_k^p(Q_{kt} - v_{kt}).$$

2. The number of pallets of inventory already loaded onto trucks at the SNS in period  $t+1$  is equal to the number of pallets already loaded in the previous time period, plus the newly loaded pallets from the current time period, minus the pallets sent to the RSS:

$$V_{k,t+1} = V_{kt} + v_{kt} - q_{kt}^p.$$

3. The number of loaded trucks at the SNS in period  $t+1$  is equal to the number of trucks already loaded in the previous time period, plus the newly loaded trucks from the current time period, minus the trucks sent to the RSS:

$$W_{t+1} = W_t + w_t - u_t^S.$$

4. The number of trucks available at the SNS in period  $t+1$  is equal to the number available during the previous time period, minus those that were sent to the RSS, plus those that returned from previous trips to the RSS, plus the change in the total number of available trucks:

$$U_{t+1}^S = U_t^S - u_t^S + \Delta_t^S \quad \text{for all } t = 1, \dots, 2\tau^S, \text{ and}$$

$$U_{t+1}^S = U_t^S - u_t^S + u_{t-2\tau^S} + \Delta_t^S \quad \text{for all } t = 2\tau^S + 1, \dots, T - 1.$$

5. The maximum number of pallets that can be loaded during each time period from the SNS is limited by a known capacity:

$$\sum_{k \in \mathcal{K}} v_{kt} \leq \sigma_t^S.$$

6. Pallets of inventory can only be loaded onto trucks at the SNS if there is inventory available:

$$\eta_k^p v_{kt} \leq I_{kt}^S.$$

7. Trucks can only be loaded at the SNS if there are trucks available:

$$w_t \leq U_t^S.$$

8. The trucks can only hold a limited amount of volume:

$$\sum_{k \in \mathcal{K}} \beta_k^v \eta_k^p v_{kt} \leq \rho^{vS} w_t.$$

$$\sum_{k \in \mathcal{K}} \beta_k^v \eta_k^p q_{kt}^p \leq \rho^{vS} u_t.$$

9. Trucks from the SNS are not sent to the RSS until the RSS has available unloading capacity:

$$u_t^S \leq \sigma_t^{RA}.$$

10. Inventory at the RSS in period  $t+1$  is equal to the inventory from the previous time period plus inventory received, minus inventory sent to the PODs:

$$I_{k,t+1}^R = I_{kt}^R - \sum_{n \in \mathcal{N}} \eta_k^c q_{knt}^c \quad \text{for all } t = 1, \dots, \tau^S, \text{ and}$$

$$I_{k,t+1}^R = I_{kt}^R - \sum_{n \in \mathcal{N}} \eta_k^c q_{knt}^c + \eta_k^p q_{k,t-\tau^S}^p \quad \text{for all } t = \tau^S + 1, \dots, T - 1.$$

11. The number of trucks available at the RSS in period  $t+1$  is equal to the number available during the previous time period, minus those that were sent to the PODs, plus those that returned from previous trips to the PODs:

$$U_{t+1}^R = U_t^R + \sum_{r \in \mathcal{R}: \tau_r < t} (u_{r,t-\tau_r} - u_{rt}) + \Delta_t^R.$$

12. Inventory can only be received if there is space (volume) available at the RSS:

$$\sum_{k \in \mathcal{K}} \beta_k^v (\eta_k^p q_{k,t-\tau^S}^p + I_{kt}^R) \leq \gamma^R.$$

13. Inventory can only be sent out to the PODs if there is inventory available at the RSS:

$$\eta_k^c \sum_{n \in \mathcal{N}} q_{knt}^c \leq I_{kt}^R.$$

14. Trucks can only be sent out to the PODs if there are trucks available at the RSS:

$$\sum_{r \in \mathcal{R}} u_{rt} \leq U_t^R.$$

15. There is a limit on the number of cases that can be loaded onto trucks headed to PODs during each time period:

$$\sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{N}} q_{knt}^c \leq \sigma_t^{RD}.$$

16. The trucks that transport inventory from the RSS to the PODs can only hold a limited amount of volume:

$$\sum_{k \in \mathcal{K}} \sum_{n \in \mathcal{R}} \beta_k^v \eta_k^c q_{knt}^c \leq \rho^{vR} u_{rt}.$$

17. Inventory at POD  $n$  in period  $t+1$  is equal to the inventory from the previous time period plus inventory received, minus inventory given out to patients:

$$I_{kn,t+1} = I_{knt} + \eta_k^c q_{nk,t-\tau_n}^c - \sum_{p \in \mathcal{P}} \phi_{kpt} s_{npt}.$$

18. The number of patients of type  $p$  waiting to be served at POD  $n$  in period  $t+1$  is equal to the number of patients waiting at the beginning of the previous time period plus those that arrived during the previous time period, minus those that were served:

$$P_{np,t+1} = P_{npt} + D_{npt} - s_{npt}.$$

19. Inventory can only be received at the POD if there is capacity to store it:

$$\sum_{k \in \mathcal{K}} \beta_k^v (\eta_k^c q_{kn,t-\tau_n}^c + I_{knt}) \leq \gamma_n^P.$$

20. Patients can only be served if there is available inventory, people, and service capacity:

$$\begin{aligned} s_{npt} &\leq D_{npt} + P_{npt}, \\ \sum_{p \in \mathcal{P}} \phi_{kpt} s_{npt} &\leq I_{knt}, \text{ and} \\ \sum_{p \in \mathcal{P}} s_{npt} &\leq \alpha_{nt}. \end{aligned}$$

21. The state variables must be initialized in the first time period:

$$\begin{aligned} U_1^S &= \xi^S, \\ U_1^R &= \xi^R, \\ I_{k1}^S &= \zeta_k^S, \\ I_{k1}^R &= \zeta_k^R, \\ I_{kn1} &= \zeta_{kn}, \text{ and} \\ P_{pn1} &= 0, \\ W_1 &= 0, \text{ and} \\ V_{k1} &= 0. \end{aligned}$$

22. All state and decision variables must be nonnegative.

### 4.3 Objective Function

As discussed previously, our objective is to minimize the total patient delay over the time horizon. Thus, our objective function is given by:

$$\min \sum_{t \in \mathcal{T}} \sum_{n \in \mathcal{N}} \sum_{p \in \mathcal{P}} c_{npt} (P_{npt} + D_{npt} - s_{npt}).$$

## References

- [1] K. Aaby, R.L. Abbey, J.W. Herrmann, M. Treadwell, C.S. Jordan, and K. Wood. Embracing computer modeling to address pandemic influenza in the 21st century. *Journal of Public Health Management Practices*, 12(4):365–372, 2006.
- [2] D. Bertsimas and J.N. Tsitsiklis. *Introduction to Linear Programming*. Athena Scientific, Belmont, MA, 1997.
- [3] C. Caggiano and J. Muckstadt. *Regional Antibiotic Storage Cost Analysis and Locator (RASCAL)*. Cornell University College of Engineering, 2009.
- [4] M. Castorena, C. Hawkins, N. Hupert, K. King, C. Lee, J. Muckstadt, and W. Xiong. Uncertainty and operational considerations in mass prophylaxis workforce planning. *Disaster Medicine and Public Health Preparedness*, To appear.
- [5] E.K. Lee, J. Mason, and W. Glisson. Large-scale dispensing for emergency response to bioterrorism and infectious disease outbreak. *Interfaces*, 36(6):591–607, 2006.

## Appendix: AMPL Model

There are a number of ways in which this model can be implemented. One of the most efficient and convenient is to describe the model in AMPL, a mathematical programming modeling language. We can then use CPLEX, or another linear programming solver, to read and solve the AMPL model, given a data set also described in AMPL. Below we show an implementation of the model described above.

```
### System parameters:
param N >= 1;
param T >= 1;

### Sets
set PODS := 1..N;
set TIME := 1..T;
set PATTYPES;
set INVTYPES;
set LOCS := 0..N; #0 is nowhere
set ROUTES in {LOCS,LOCS,LOCS,LOCS,LOCS};
#routes can have at most 5 stops, not including the RSS
#(assume first and last stops are the RSS)

### System Parameters
param PODdemand {PODS, PATTYPES, TIME} >= 0;
param PODstoragecap {PODS} >= 0;
param PODservicecap {PODS, TIME} >= 0;
param PODinitinv {INVTYPES, PODS} >= 0;
param PODinitpats {PODS, PATTYPES} >= 0;

param RSSloadingcap {TIME} >= 0;
param RSSunloadingcap {TIME} >= 0;
param RSSstoPODtime {PODS} >= 1;
param RSSroutetime {ROUTES} >= 1;
param RSSstrucks >= 1;
param RSSstoragecap >= 1;
param RSSstruckvolcap >= 0;
param RSSinitinv {INVTYPES} >= 0, <=RSSstoragecap;
param RSSstrucksadded {TIME};
```

```

param SNSinitinv {INVTYPE} >= 0;
param SNSDeliveries {INVTYPE, TIME} >= 0;
param SNSLoadingcap {TIME} >= 0;
param SNStorSStime >= 1;
param SNStrucks >= 1;
param SNStruckvolcap >= 0;
param snstrucksadded {TIME};

param invvol {INVTYPE} >= 0;
param unitspercase {INVTYPE} >= 0;
param unitsperpallet {INVTYPE} >= 0;
param unitsperpat {INVTYPE, PATTYPE, TIME};

### Cost parameters
param HoldingCost {PODS} >= 0;
param BackorderCost {PODS} >= 0;

### Decision Variables
var shippedtorss {INVTYPE, TIME} >= 0;
var shippedtopod {INVTYPE, PODS, TIME} >= 0;
var served {PODS, PATTYPE, TIME} >= 0;
var truckssenttorss {TIME} >= 0;
var truckssenttopod {ROUTES, TIME} >= 0;
var palletsloaded {INVTYPE, TIME} >= 0;
var trucksloaded {TIME} >= 0;

### State Variables
var snstrucksavail {TIME} >=0, <=SNStrucks;
var rsstrucksavail {TIME} >=0, <=RSStrucks;
var snsinventory {INVTYPE, TIME} >= 0;
var rssinventory {INVTYPE, TIME} >= 0, <=RSSstoragecap;
var inventory {INVTYPE, PODS, TIME} >= 0;
var waitingpats {PODS, PATTYPE, TIME} >= 0;
var readypallets {INVTYPE, TIME} >= 0;
var readytrucks {TIME} >= 0;

### Objective Function
minimize TotalCost:
    sum{k in INVTYPE, n in PODS, t in TIME} (HoldingCost[n]*inventory[k,n,t]
        + BackorderCost[n]*inventory[k,n,t]);

### Constraints
subject to

#1
SNSUpdateInventory{k in INVTYPE, t in 1..T-1}:
    snsinventory[k,t+1] = snsinventory[k,t] + unitsperpallet[k]*(SNSDeliveries[k,t]
        - shippedtorss[k,t]);

#2
SNSUpdateReadyPallets{k in INVTYPE, t in 1..T-1}:
    readypallets[k,t+1] = readypallets[k,t] + palletsloaded[k,t] - shippedtorss[k,t];

```

```

#3
SNSUpdateReadyTrucks{t in 1..T-1}:
  readytrucks[t+1] = readytrucks[t] + trucksloaded[t] - truckssenttorss[t];

#4
SNSUpdateTrucksA{t in 1..2*SNStoRSStime}:
  snstrucksavail[t+1] = snstrucksavail[t] - truckssenttorss[t] + SNStrucksadded[t];

SNSUpdateTrucksB{t in 2*SNStoRSStime+1..T-1}:
  snstrucksavail[t+1] = snstrucksavail[t] - truckssenttorss[t]
    + truckssenttorss[t-2*SNStoRSStime] + SNStrucksadded[t];

#5
SNSPalletLimit{t in TIME}:
  sum{k in INVYPES} palletsloaded[k,t] <= SNSloadingcap[t];

#6
SNSInventoryLimit{k in INVYPES, t in TIME}:
  unitsperpallet[k]*palletsloaded[k,t] <= snsinventory[k,t];

#7
SNSTrucksLimit{t in TIME}:
  trucksloaded[t] <= snstrucksavail[t];

#8a
SNSTruckLoadedVol{t in TIME}:
  sum{k in INVYPES} unitsperpallet[k]*invvol[k]*palletsloaded[k,t]
    <= SNStruckvolcap*truckssenttorss[t];

#8b
SNSTruckSentVol{t in TIME}:
  sum{k in INVYPES} unitsperpallet[k]*invvol[k]*shippedtorss[k,t]
    <= SNStruckvolcap*truckssenttorss[t];

#9
RSSUnloadingLimit{t in TIME}:
  truckssenttorss[t] <= RSSunloadingcap[t];

#10
RSSUpdateInventoryA{k in INVYPES, t in 1..SNStoRSStime}:
  rssinventory[k,t+1] = rssinventory[k,t]
    - sum{n in PODS} unitspercase[k]*shippedtopod[k,n,t];

RSSUpdateInventoryB{k in INVYPES, t in SNStoRSStime+1..T-1}:
  rssinventory[k,t+1] = rssinventory[k,t]
    + unitsperpallet[k]*shippedtorss[k,t-SNStoRSStime]
    - sum{n in PODS} unitspercase[k]*shippedtopod[k,n,t];

#11
RSSUpdateTrucks{t in 1..T-1}:
  rsstrucksavail[t+1] = rsstrucksavail[t] + RSSTrucksadded[t]
    - sum{(a,b,c,d,e) in ROUTES} truckssenttopod[a,b,c,d,e,t]
    + sum{(a,b,c,d,e) in ROUTES: RSSroutetime[a,b,c,d,e] < t}
      truckssenttopod[a,b,c,d,e,t-RSSroutetime[a,b,c,d,e]];

```

```

#12
RSSStorageA{t in 1..SNStoRSStime}:
    sum{k in INVYPES} invvol[k]*rssinventory[k,t] <= RSSstoragecap;

RSSStorageB{t in SNStoRSStime+1..T}:
    sum{k in INVYPES} invvol[k]*(unitsperpallet[k]*shippedtorss[k,t-SNStoRSStime]
        + rssinventory[k,t]) <= RSSstoragecap;

#13
ConserveInventory {k in INVYPES, t in TIME}:
    unitspercase[k]* sum{n in PODS} shippedtopod[k,n,t] <= rssinventory[k,t];

#14
RSSTrucksLimit{t in TIME}:
    sum{(a,b,c,d,e) in ROUTES} truckssenttopod[a,b,c,d,e,t] <= rsstrucksavail[t];

#15
RSSLoadingLimit{t in TIME}:
    sum{k in INVYPES, n in PODS} shippedtopod[k,n,t] <= RSSloadingcap[t];

#16
RSSTruckVol{t in TIME, (a,b,c,d,e) in ROUTES}:
    sum{k in INVYPES, n in PODS: n=a or n=b or n=c or n=d or n=e}
        unitspercase[k]*invvol[k]*shippedtopod[k,n,t]
        <= RSSTruckvolcap*truckssenttopod[a,b,c,d,e,t];

#17
UpdateInventory{k in INVYPES, n in PODS, t in 1..T-1}:
    inventory[k,n,t+1] = inventory[k,n,t] + unitspercase[k]*shippedtopod[k,n,t]
        - sum{p in PATYPES} unitsperpat[k,p,t]*served[n,p,t];

#18
UpdatePats{n in PODS, p in PATYPES, t in 1..T-1}:
    waitingpats[n,p,t+1] = waitingpats[n,p,t] + PODdemand[n,p,t] - served[n,p,t];

#19
PODStorage{n in PODS, t in SNStoRSStime+1..T}:
    sum{k in INVYPES} invvol[k]*(unitspercase[k]*shippedtopod[k,n,t-SNStoRSStime]
        + inventory[k,n,t]) <= PODstoragecap[n];

#20a
PatientsLimit {n in PODS, p in PATYPES, t in TIME}:
    served[n,p,t] <= waitingpats[n,p,t] + PODdemand[n,p,t];

#20b
InventoryLimit {k in INVYPES, n in PODS, t in TIME}:
    sum{p in PATYPES} unitsperpat[k,p,t]*served[n,p,t] <= inventory[k,n,t];

#20c
ServiceCapacity {n in PODS, t in TIME}:
    sum{p in PATYPES} served[n,p,t] <= PODservicecap[n,t];

#21

```

```
InitSNSTrucks:
  snstrucksavail[1] = SNSTrucks;

InitRSSTrucks:
  rsstrucksavail[1] = RSSTrucks;

InitSNSInv{k in INVTYPES}:
  snsinventory[k,1] = SNSinitinv[k];

InitRSSInv{k in INVTYPES}:
  rssinventory[k,1] = RSSinitinv[k];

InitPODInv{k in INVTYPES, n in PODS}:
  inventory[k,n,1] = PODinitinv[k,n];

InitPats{n in PODS, p in PATTYPES}:
  waitingpats[n,p,1] = PODinitpats[n,p];

InitLoadedTrucks:
  trucksloaded[1] = 0;

InitLoadedPallets{k in INVTYPES}:
  palletsloaded[k,1] = 0;
```