## Ethanol Transport Model

We present an optimization model to determine over a repeated time horizon the configuration of leased and purchased cars that will provide the least capital and operating costs given a forecast for a decision period and a set of realized costs due to actual demands in a recourse stage occurring after the procurement decision. It consists of three nested parts: a Scheduling Model, a Provisioning Model, and a Horizon Model. We will explain the timing in terms of our case; Scheduling is stochastic and run for an operating cycle, in our case two weeks; Provisioning is run for each decision cycle, for us quarter of 3 months, comprising 6 Scheduling cycles; and the Horizon model is run to plan over a horizon of multiple quarters; in our case the horizon is 5 decision cycles, or 15 months, since this is the term of a typical railcar lease.

In the Scheduling Model, we use a linear (integer) program to schedule the available cars into unit trains to meet realized demand during the operating cycle, which reflects the time required for the cars to go to the destination and return to the base to be used again. For our example this time is two weeks. The objective function calculates the operating cost of assigning the proper number of cars in unit trains to meet demand. Costs are measured per car per trip, and include maintenance, operating expenses, a risk factor reflecting the probability that a car may not finish the trip, inspection costs per trip, and car movement charges. Other costs in the objective are a charge per car per trip for idle cars (not used in this period because they cannot be put in a unit train), and a cost, higher than operating costs, for outsourcing a carload for one trip, incurred if demand cannot fit into the unit trains. We assume demand cannot be backlogged, so outsourcing transport allows us to fill it exactly.

There are several constraints. Unit trains must contain between a minimum and maximum number of cars. If demand cannot be fit into the unit trains, it must be outsourced. We cannot load more cars than are available. Available cars are those in our leased pool, and those in our purchased pool, less those that have been subleased for this time period (in Figure 1, the blue parts only). There is a cost associated with idle cars; storage space, insurance, maintenance, and security, as well as tracking.

Scheduling is run multiple times over all possible demands from 10 to 150, so we can obtain the expected cost over a distribution of demand. For our example we presumed demand is Poisson distributed with mean calculated from the forecast by weighting each operating period by an appropriate ‘seasonal’ factor; experience shows that realized demand does not occur evenly across the six two-week periods in our decision period of 3 months. We truncated the Poisson distributions at a minimum of 10 carloads and maximum of 150 in a two week cycle. The output from the Scheduling model is therefore the expected value of the recourse cost, calculated as the sum of the cost from each run at a specific potential demand weighted by its appropriate Poisson likelihood and summing. Similarly we calculate the weighted average number and costs of idle and outsourced cars. It would also be possible to produce exact schedules for unit train cars, but this is not necessary for the overall goal of determining the least cost portfolio of leased and purchased cars.

The Provisioning Model uses a simple integer program to determine the number of leased and purchased cars to make available at the start of a three month provisioning period. Leased and purchased cars are acquired in different provisioning periods, and so have different acquisition costs. Thus the pools are broken into sub-pools of identical cost per car. In addition, subleasing may occur only at the start of a provisioning period; however there may be cars subleased at different rates. The objective includes the fixed costs per car of the leased, purchased, and (negative, since it is revenue) subleased cars.

The model constraints include assuring that the cars in the sub-pools add to the total number of cars of each type; the subleased cars in each sub-pool do not exceed the number of leased plus purchased cars in the sub-pool; and the number of cars must be within a lower and an upper bound. The minimum constraint can be initialized at the minimum number of cars in a unit train, since there is no point in having cars unless we can make at least one unit train; the maximum constraint can be initialized at a number large enough to at least cover the largest expected demand in any single two-week period.

However, the minimum constraint will be adjusted as we cycle through the Scheduling and Provisioning models to zero in on the minimum cost provisioning configuration.

The model algorithm for a single provisioning (three-month) decision period is then as follows:

1. Initialize with the current number of leased, purchased, and subleased cars in each sub-pool.
2. Obtain the Forecast of demand over the three month decision period.
3. Determine demand probability distribution parameters for 6 operating cycles, proportioning the forecasted decision period demand by a seasonal factor for each cycle.
4. Repeat
   1. For each operating cycle:
      1. Solve Scheduling for each possible demand state.
      2. Average costs over possible demand states to obtain expected recourse cost.
      3. Determine average number of idle cars and outsourced cars.
      4. Determine average idle cost and outsource cost.
   2. Sum expected recourse costs, idle costs, outsource costs, idle cars, and outsourced cars over operating cycles in decision cycle
   3. Adjust inputs to Provisioning by changing total number of cars. (1)
   4. Solve Provisioning for leased, subleased and purchased cars and provisioning cost including the expected recourse costs of Scheduling.
   5. Compare average idle cost to average outsource cost.
      1. If idle cost > outsource cost, (1) reduce total number of cars.
      2. If outsource cost >= idle cost, (1) increase total number of cars.
   6. If direction of inequality has changed once or more, update minimum expected recourse costs, idle costs, outsource costs, idle cars, and outsourced cars, and minimum provisioning cost, and minimum number of leased, purchased, and subleased cars. (2)
   7. If direction of inequality has changed twice, stop repeating.
5. Report minimum expected recourse costs, idle costs, outsource costs, idle cars, and outsourced cars, and minimum provisioning cost, and minimum number of leased, purchased, and subleased cars.

Notes:

1. Search on the number of cars to find the optimum number. Increase number of cars if idle cost < outsource cost, and decrease the number of cars if outsource costs>= idle costs.
2. Since the sum of idle costs and outsource costs is a convex function, there is a unique number of cars that minimizes the sum of outsource costs and idle costs, or balances outsource cost against idle cost as well as possible. Therefore we search till the direction switches, then search back, recording minimum value. The number of cars yielding lowest total cost is then optimal for the decision cycle. Once the direction switches twice we know the minimum is trapped between the switch points.

Output is a provisioning policy (number of leased, purchased, and subleased cars) for the decision period, and its expected cost for the provisioning period. With this in hand the user can easily calculate the adjustments to the prior period provisioning to achieve it. The routine above should be run for each provisioning decision period of three months, before the start of the period, and after the decision period forecast is available.

Next we embed the provisioning period problem in a multi-period model for a horizon of five periods. 15 months is chosen in our example because that is the standard length of lease. Other horizons could be studied and would obtain different results. Multi-period problems are sensitive to the ‘edge’ effects created by initial conditions and target status at the end of the horizon.

This Horizon Model allows determination of when to change the number of cars available, and by how much of each type. Figure 1 shows the possible changes to the leased, purchased, and subleased pools.

Figure Dynamics of railcar portfolio adjustment

Purchased

Cars (P)

Subleased

Cars

Leased

Cars (L)

New/ add-on Lease (1)

New Purchase

(2)

Recall from sublease (6L and 6P)

Sublease (3L)

Cancel Lease (4)

Sell (5)

Sublease (3P)

Railcars fall into one of three states: In figure 1 we show how the state may change. Cars are either leased for a period of 15 months, purchased, or the firm has subleased them to another party for a period, assumed to be 6 months (blue and orange inventory boxes). The firm’s portfolio of cars for a given Quarter (90 days) consists of the leased cars and purchased cars that are not subleased to someone else. The subleased cars generate revenue, though they do have some operating costs, calculated in the Recourse model. To add cars, the firm can purchase cars (arrow 2), or newly lease cars or add to an existing lease (arrow 1); these are red (cost money). If there are cars already subleased, they can be recalled from subleases at the appropriate time (arrows 6, also red, since the cash flow will be negative). To reduce the portfolio of cars, the firm can cancel the lease on a car (arrow 4, green, saves money) or sell the car (arrow 5, also green). The firm can also reduce the number of available cars without losing control of them for later by subleasing them (arrows 3, green because net cash flow will probably be positive). In the figure, the Purchased and Leased car inventories are shown with two colored segments, since the subleased cars come from either the Leased inventory or the Purchased inventory. The orange Subleased inventory is identical with the union of the two orange triangles inside the Leased and Purchased inventories. It represents a group of cars which are partially available, a backup inventory of controlled cars which could be pressed into service later, but in the meantime are generating revenue. We have assumed the firm writes only subleases for 6 months, so that cars committed could be available six months later for transport. This safety inventory is more productive than simply idling the cars; but the savings comes with a downside that they are only available 6 months from inception.

Here is my current thinking on this problem.

We draw from the lot sizing literature on replenishment in the face of irregular or lumpy demand. The Silver-Meal heuristic shows a very good record of practical applicability. It chooses a replenishment schedule to minimize the average cost per period of the sum of the setup costs (major and minor, particular to each item) and the carrying costs of having extra (idle) cars available for multiple periods. Such an objective is locally concave so that there is likely a number of periods which minimizes the cost.

We wish to determine at what points over the horizon to modify the portfolio of cars by adding or subtracting cars. Adds can be made by new purchases, adding a car to a lease, or recalling a leased or purchased car from a sublease, if there are any eligible. Subtracts can happen through disposing of a purchased car by sale, disposing of a leased car by taking it off the lease, or by subleasing a purchased or leased car to another firm. These transactions are accompanied by several costs. A *major setup cost* is a processing cost for any transaction of any type regardless of size, which includes cost of finding a second party, completing an agreement, and processing the order. A *minor setup cost* is a form of processing cost that is particular to the type of transaction, but otherwise occurs once for each transaction, not per car. Examples are cleaning and preparation costs, the cost of moving or shunting cars to make them available for the new owner, and any maintenance, inspection, or risk associated costs particular to the batch of cars. There is a minor setup cost for each of the 8 different add or subtract actions.

The *carrying cost* of a car consists of two parts, particular to the type of car; the capital cost, and the expected cost per unit of idle cars. The last two figures are outputs available from the Scheduling model. The capital cost is an input to the Provisioning model; for leased and purchased cars it is a cost, for subleased cars it is the net of the sublease revenue and the lease or purchase cost, and may be negative or positive.

## Our Replenishment model assumes we choose to replenish up to a particular period h in the overall horizon from 1 to H. In this period the replenishment cost is the sum of the setup costs (occurring in period 1) and the carrying costs (occurring in periods t in 1 through h). Transaction types are denoted by b. B\_b is a two valued variable +1 for an add and -1 for a subtract; Y\_bt is a binary variable indicating whether a particular transaction is used or not in a period.

RC(h) = A + sum\_{b} a\_b + sum\_ {t in 1 .. h} sum\_{b} Delta\_bh B\_b Y\_bh (ECQ/EQ + K\_b)

The objective we wish to minimize is then

RCUT(h) = RC(h) / h

The Silver-Meal heuristic proceeds by

Let h := 1.

Let Last\_RCUT := Infinity

For {b in B} Let Last\_Delta\_b := 0

Repeat {

Calculate RCUT(h)

If RCUT(h) > Last\_RCUT then break; #RCUT should be descending with increasing h

If h := H then break;

Let Last\_RCUT := RCUT(h);

For { b in B } let Last\_Delta\_b := Delta\_bh

Let h := h + 1;

}

If RCUT(h) <= Last\_RCUT then

Let h\* := h

Let RCUT(h\*) := RCUT(h)

For {b in B} Let Delta\_b\* := Delta\_bh

;

If RCUT(h) > Last\_RCUT then

Let h\* := h-1

Let RCUT(h\*) := Last\_RCUT(h-1)

For {b in B} Let Delta\_b\* := Last\_Delta\_b

;

Since RCUT is concave, and setup costs are being spread over more periods, RCUT will decline till the minimum is past, and the optimal number of periods is on one or the other side of the upturn.

The result should tell us that we should make the changes in portfolio allocation in period h\*, and the Delta\_b\* will be the changes to make; the cost will be RCUT(h\*). Once h\* is determined, the decision is frozen for h\* periods, the changes Delta\_b\* are made, and the decision is reevaluated for H future periods when the change ‘expires’.

## Appendix

Model file for Scheduling Subproblem

/\* MODEL FOR SUBPROBLEM (RECOURSE)

Ethanol Transport Model Simple version

A bin packing problem with constraints. \*/

#function gsl\_cdf\_poisson\_P;

set TRAINS;

param F\_demand >= 0 integer; #Forecasted demand in carloads

param E\_demand >= 0 integer; #Realized demand in carloads

param Mmax >=0 integer; #max length of a unit train

param Mmin >= 0 <=Mmax, integer; #min length of a unit train

param Nmax integer >=0 ; #max no of unit trains ;

param Nmin integer >=0 ; #minimum no of unit trains

#given number of cars

param L ;

param P ;

param S ; #starting no of L, P, S cars

#costs of car types

param CL >= 0 ;

param CP >=0 ;

param CO >= 0 ;

param CS >=0; #costs of car types

param CQ >=0; #costs of unused car

param CY {TRAINS} >=0; #ordering cost for trains

var XL {i in TRAINS} >=0 integer; #no of L cars in ith train

var XP {i in TRAINS} >=0 integer; #no of P cars in ith train

var O >= 0 integer ;

var Q >=0 integer; #no outsourced, no not used

var N >=0 integer; # number of trains actually used

var Y {i in TRAINS} >=0 <=1 binary; # binary, 1 if used and 0 if not

minimize RCost: CL\*sum {i in TRAINS} XL[i] + CP\*sum {i in TRAINS} XP[i] + CO\*O + CQ\*Q + sum {i in TRAINS} CY[i]\*Y[i]

;

#costs to run L, P cars

#cost to outsource, idle

#fill order cost parameters

subject to notrains: N = sum {i in TRAINS} Y[i] ; #no of trains used

subject to mintrains: sum {i in TRAINS} Y[i] >= Nmin ; #at least Nmin trains

subject to maxtrains: sum {i in TRAINS} Y[i] <= Nmax ; #no more than Nmax trains

subject to trainmin {i in TRAINS}: (XL[i] + XP[i]) >= Mmin \* Y[i] ; #must have Mmin in a train

subject to trainmax {i in TRAINS}: (XL[i] + XP[i]) <= Mmax \* Y[i] ; #no more than Mmax in train

subject to useallcars: sum{i in TRAINS} ( XL[i] + XP[i] ) + Q = L + P - S ; #define Q the idle cars

subject to outsource: sum{i in TRAINS} ( XL[i] + XP[i] ) + O = E\_demand ; #define O the outsourced cars

subject to Lcars: sum{i in TRAINS} XL[i] <= L ; #cant exceed the no of L cars available

subject to Pcars: sum{i in TRAINS} XP[i] <= P ; #cant exceed the no of P cars

Data file for Scheduling Problem

/\* Ethanol Transport Data Simple version

\*/

#param F\_demand := 200 ; #Forecasted demand in carloads, not needed in recourse model

param E\_demand := 100 ; #Realized demand in carloads

param Mmax := 50 ; #max length of a unit train

param Mmin := 20 ; #min length of a unit train

param Nmax := 5 ; #E\_demand/Mmin ;

param Nmin := 0 ; #E\_demand/Mmax ;

set TRAINS := 1 2 3 4 5 ; #no of hypothetical trains

#given number of cars

param L ;

param P ;

param S ; #starting no of L, P, S cars

#costs of car types

param CL := 10 ;

param CP := 9 ;

param CO := 30 ;

#costs of car types

param CQ := 50 ; #costs of unused car (and setup cost of unit train not here)

param: CY :=

1 1

2 2

3 3

4 4

5 5

; #ordering cost for trains

Provisioning Problem Model

/\* MASTER MODEL FOR QUARTER

Ethanol Transport Period Model (Quarters) \*/

**param** NLpool >= 0 **integer** **default** 3; #size of price pools

**param** NPpool >= 0 **integer** **default** 3;

**param** NSpool >= 0 **integer** **default** 3;

**set** LPOOL ;

**set** PPOOL ;

**set** SPOOL ;

**set** PDS ;

**param** F ; # this is the forecasted demand for the 3 month period.

**param** NQ ; #number of stages in each period (6 semimonths in a quarter)

#param seas {PDS} >=0 <=1; #seasonal factors to spread F over the stages (months) in the period (qtr)

**param** EDQ {PDS} ;

**param** KL {LPOOL} >= 0 **integer**; #monthly ownership costs of each pool

**param** KP {PPOOL} >= 0 **integer**;

**param** RS {SPOOL} >= 0 **integer**; #this one will be a revenue from sublease

#param Mmax >=0 integer default 50; #max length of a unit train

#param Mmin >= 0 <=Mmax integer default 20; #min length of a unit train

#param Nmax integer >=0 default 5; #max no of unit trains ;

#param Nmin integer >=0 default 2; #minimum no of unit trains

**var** La {LPOOL} >=0 **integer**;

**var** Pa {PPOOL} >=0 **integer**;

**var** Sa {SPOOL} >=0 **integer**;

**var** Lop ;

**var** Pop ;

**var** Sop ;

**param** ERecourseCost ; # recourse costs from stages (months)

**minimize** Qcost: **sum** {i **in** LPOOL} KL[i]\*La[i] + **sum** {i **in** PPOOL} KP[i]\*Pa[i] - **sum** {i **in** SPOOL} RS[i]\*Sa[i]

+ ERecourseCost ;

**subject** **to** lcars: **sum**{i **in** LPOOL} La[i] = Lop ;

# the pools add up to LL, PP, SS

**subject** **to** pcars: **sum** {i **in** PPOOL} Pa[i] = Pop ;

**subject** **to** scars: **sum** {i **in** SPOOL} Sa[i] = Sop ;

**subject** **to** slim {i **in** LPOOL}: Sa[i] <= La[i] + Pa[i];

**subject** **to** cars: **sum** {i **in** LPOOL} La[i] + **sum** {i **in** PPOOL} Pa[i] - **sum** {i **in** SPOOL} Sa[i] >= 20;

**subject** **to** totcars: **sum** {i **in** LPOOL} La[i] + **sum** {i **in** PPOOL} Pa[i] + **sum** {i **in** SPOOL} Sa[i] >= F / **card**(PDS);

Data file for Provisioning Model

# DATA FOR MASTER PROBLEM ETQ

**param** NLpool := 2;

**param** NPpool := 2;

**param** NSpool := 2;

**set** LPOOL := 1 2 ;

**set** PPOOL := 1 2 ;

**set** SPOOL := 1 2 ;

**param** NQ := 3 ;

**set** PDS := 1 2 3 4 5 6;

**param** KL := 1 300 2 240 ; #monthly ownership costs of each pool

**param** KP := 1 600 2 650 ;

**param** RS := 1 125 2 200 ; #this one will be a revenue from sublease

#param F ;

#param seas := 1 0.25 2 0.40 3 0.35;

Run file for Logic of Provisioning Period

#master program for period

load amplgsl.dll;

function gsl\_cdf\_poisson\_P;

**option** solver cplex;

**option** solver\_msg 0;

**model** EthanolTransportStageModel.mod;

**data** EthanolTransportStageData.dat;

**model** ETQ.mod;

**data** ETQ.dat;

**problem** Recourse: XL,XP,O,Q,N,Y,RCost,

notrains, mintrains, maxtrains, trainmin, trainmax, useallcars,

outsource, Lcars, Pcars

;

**option** relax\_integrality 1;

**problem** Quarter: La, Pa, Sa, Lop, Pop, Sop, Qcost,

lcars, pcars, scars, slim, cars, totcars

;

**let** PDS := 1 ..6 **by** 1;

**let** F := 400 ;

**param** seas {PDS} ;

**let** seas[1] := .20; **let** seas[2] := .30; **let** seas[3] := .10 ;

**let** seas[4] := .15; **let** seas[5] := .10; **let** seas[6] := .15 ;

**param** Qcostold;

**let** Qcostold := Infinity ;

**param** Linit; **param** Pinit; **param** Sinit;

**let** Linit := 60; **let** Pinit := 20; **let** Sinit := 5; #initial values

**set** EDR;

**param** Pr {EDR} ;

**param** PPr {EDR} ;

**param** PPr\_mean ;

**param** PPr\_trunc\_const ;

**param** edr\_obj {EDR} ;

**param** edr\_dist {EDR} ;

**param** Expected\_Cost {PDS} ;

**param** Recourse\_Costs ;

**param** edr\_O {EDR};

**param** edr\_Q {EDR};

**param** edr\_XP {EDR,1..5}; **param** edr\_XL {EDR,1..5};

**param** Emin := 10 ; **param** Emax := 150 ; **param** Estep := 1 ;

#param L ; param P; param S;

**param** LL; **param** PP; **param** SS;

**let** L := Linit;

**let** P := Pinit;

**let** S := Sinit;

**repeat** {

**printf** "\n%20s \n", "Starting Subproblems" ;

**for** {q **in** PDS} {

**let** EDQ[q] := F\*seas[q];

**let** EDR := Emin .. Emax **by** Estep ;

**for** {i **in** EDR} {

**let** E\_demand := i;

#option send\_statuses 1;

**solve** Recourse > scratchfile.out;

#printf "%10.0f %10.0f %10.0f %10.0f %10.0f\n", E\_demand, RCost, O, Q, N;

#let Pr[i] := 1 / card{EDR};

**let** PPr\_trunc\_const := gsl\_cdf\_poisson\_P(Emax,EDQ[q]) - gsl\_cdf\_poisson\_P(Emin-1,EDQ[q]) ;

**let** PPr[i] := ( gsl\_cdf\_poisson\_P(i,EDQ[q]) - gsl\_cdf\_poisson\_P(i-1,EDQ[q]) ) / PPr\_trunc\_const;

**let** edr\_obj[i] := RCost ;

**let** edr\_dist[i] := edr\_obj[i] \* PPr[i] ;

**let** edr\_O[i] := O ;

**let** edr\_Q[i] := Q ;

**for** {j **in** 1..5} {

**let** edr\_XP[i,j] := XP[j]; **let** edr\_XL[i,j] := XL[j];

}

}

**let** Expected\_Cost[q] := **sum** {i **in** EDR} edr\_dist[i];

#let Expected\_Q := sum {i in EDR} edr\_Q[i] ;

#let Expected\_O := sum {i in EDR} edr\_O[i] ;

#display edr\_obj, PPr, edr\_O, edr\_Q;

#display "Start", q;

#printf "%15s %1d\n", "Results for q=",q;

**printf** "%5s %2d %15s %5d %5d %5d %10s %10.3f\n", "q=",q,"(L P S)=",L, P, S,"ERcost=",Expected\_Cost[q];

#display L, P, S, edr\_expected;

#display edr\_XP, edr\_XL ;

}

**let** Recourse\_Costs := **sum** {q **in** PDS} Expected\_Cost[q] ;

**printf** "%15s: $%10.2f\n\n", "Total Recourse cost",Recourse\_Costs;

**printf** "%15s %10.0f %10.0f %10.0f \n\n", "Duals =", Lcars, Pcars, useallcars;

**let** ERecourseCost := Recourse\_Costs;

**printf** "%30s %10.3f\n", "Starting Master Problem with recourse cost = ",ERecourseCost;

**solve** Quarter;

**printf** "\n%10s %10.2f %10s %10.0f %10.0f %10.0f\n\n", "Qcost=", Qcost,"(Lop Pop Sop)=", Lop, Pop, Sop;

**printf** "%15s %10.3f %10.3f\n", "Dual Prices: ", lcars.dual, pcars.dual;

**let** L := Lop; **let** P := Pop; **let** S := Sop;

**if** Qcostold - Qcost >0.00001

**then** { **let** Qcostold := Qcost ; **printf** "%15s\n", "About to try again."; }

**else** {**printf** "%15s\n","About to break!";

**break**; }

};

**for** {q **in** PDS} {

**let** EDQ[q] := F\*seas[q];

**let** PPr\_mean := EDQ[q] ;

**let** EDR := Emin .. Emax **by** Estep ;

# printf "%15s %10s %10s %10s %10s %10s\n", "","E","Cost","O","Q", "N";

**for** {i **in** EDR} {

**let** E\_demand := i;

#option send\_statuses 1;

**solve** Recourse > scratchfile.out;

#printf "%10.0f %10.0f %10.0f %10.0f %10.0f\n", E\_demand, RCost, O, Q, N;

#let Pr[i] := 1 / card{EDR};

**let** PPr\_trunc\_const := gsl\_cdf\_poisson\_P(Emax,PPr\_mean) - gsl\_cdf\_poisson\_P(Emin-1,PPr\_mean) ;

**let** PPr[i] := ( gsl\_cdf\_poisson\_P(i,PPr\_mean) - gsl\_cdf\_poisson\_P(i-1,PPr\_mean) ) / PPr\_trunc\_const;

**let** edr\_obj[i] := RCost ;

**let** edr\_dist[i] := edr\_obj[i] \* PPr[i] ;

**let** edr\_O[i] := O ;

**let** edr\_Q[i] := Q ;

**for** {j **in** 1..5} {

**let** edr\_XP[i,j] := XP[j]; **let** edr\_XL[i,j] := XL[j];

}

}

**let** Expected\_Cost[q] := **sum** {i **in** EDR} edr\_dist[i];

#display edr\_obj, PPr, edr\_O, edr\_Q;

#display "Start", q;

#printf "%15s %1d\n", "Results for q=",q;

**printf** "%5s %2d %10s %5d %5d %5d %10s %10.3f\n", "q=",q,"(L P S)=",L, P, S, "ERcost=",Expected\_Cost[q];

#display L, P, S, edr\_expected;

#display edr\_XP, edr\_XL ;

}

**let** Recourse\_Costs := **sum** {q **in** PDS} Expected\_Cost[q] ;

**printf** "%15s: $%10.2f\n", "Total Recourse cost",Recourse\_Costs;

**printf** "%15s: %10d %10d %10d\n", "Lop Pop Sop", Lop, Pop, Sop;