```
! ****************************************** !
!   Master's thesis SPRING 2013
!   Tonnage Allocation Problem
!   Øydis Kristine Flateby

! ****************************************** !

model TonnageAllocation

options explterm
options noimplicit

uses "mmxprs", "mmive", "mmsystem", "mmodbc";

parameters
    TurboRouterFile = "3T_withTC.txt";
    DataFile = "DataFile_3T.txt";

    MODELNAME = 'TonnageAllocation_problem';
    DATETIME =  " "+ getday(datetime(SYS_NOW)) + "-" + getmonth(datetime(SYS_NOW))
                + "-" + getyear(datetime(SYS_NOW)) + " " + gethour(datetime(SYS_NOW))
                + "-" + getminute(datetime(SYS_NOW)) ;
end-parameters


declarations
    Time:               real;                           ! Time when performing optimization
    IPRECUT:            integer;                        ! Parameter for optimization
    ICPU:               integer;                        ! Parameter for optimization

    Ship:               set of integer;                 ! Set of ships
    TradeM:             set of integer;                 ! Set of trades that are mandatory
    TradeTC:            set of integer;                 ! Set of trades that are optional, TC-trades
    TradeDD:            set of integer;                 ! Set of dry-docking "trades"
    Trade:              set of integer;                 ! All of the trades, the union of the three above
    Way:                set of integer;                 ! Set of ways of performing a voyage

    NumShip:            integer;                        ! Number of ships
    NumTradeM:          integer;                        ! Number of mandatory trades
    NumTradeTC:         integer;                        ! Number of optional, TC, trades
    NumTradeDD:         integer;                        ! Number of dry-docking "trades", how many ships need drydocking
    NumTrade:           integer;                        ! Total number of trades
    NumWay:             integer;                        ! Number of ways to perform a voyage
end-declarations

initializations from  TurboRouterFile
    NumShip;
    NumTradeM;
    NumTradeTC;
    NumTradeDD;
end-initializations

initializations from DataFile
    NumWay;
    IPRECUT;
    ICPU;
end-initializations

Time := gettime;
```

```
Ship := 1..NumShip;

if not (NumTradeM = 0) then
    TradeM  := 1..NumTradeM;
end-if
if not (NumTradeTC = 0) then
    TradeTC := NumTradeM+1..NumTradeM+NumTradeTC;
end-if
if not (NumTradeDD = 0) then
    TradeDD := NumTradeM+NumTradeTC+1..NumTradeM+NumTradeTC+NumTradeDD;
end-if

Way := 1..NumWay;

finalize(Ship);
finalize(TradeM);
finalize(TradeTC);
finalize(TradeDD);
finalize(Way);

NumTrade:= NumTradeM + NumTradeTC + NumTradeDD;
Trade   := 1..NumTrade;

finalize(Trade);

declarations
    maxVoyage:          integer;                                    ! Maximum number of voyages in one of the trades
    Voyage:             set of integer;                             ! Set of voyage for trade
    NumVoyage:          dynamic array(Trade)        of integer;     ! Number of voyages on each trade

    ShipNames:          array(Ship)                 of string;      ! Names of ships
    TradeMNames:        array(TradeM)               of string;      ! Names of the mandatory trades
    TradeTCNames:       array(TradeTC)              of string;      ! Names of TC-trades, optional trades
    TradeDDNames:       array(TradeDD)              of string;      ! Names of dry-docking "trades"
    TradeNames:         array(Trade)                of string;      ! Names of all trades
    TradeStartNames:    array(Trade)                of string;      ! Names of start ports of the trades
    TradeStopNames:     array(Trade)                of string;      ! Names of stop ports of the trades

    TradeDDShip:        dynamic array(TradeDD,Ship) of integer;     ! Compatibility between TradeDD and Ship
    StartPort:          array(Ship)                 of integer;     ! The start port for ship in this timeperiod.

end-declarations

initializations from  TurboRouterFile
    NumVoyage;
    StartPort;
    TradeDDShip;
    ShipNames;
    TradeMNames;
    TradeTCNames;
    TradeDDNames;
    TradeStartNames;
    TradeStopNames;
end-initializations

maxVoyage := max (rr in Trade) NumVoyage(rr);       ! Find the largest number of voyages
Voyage    := 1..maxVoyage;
```

```
finalize(Voyage);

declarations
    !Parameters
    CostBallast:         dynamic array(Ship,Trade,Trade)                 of real;    ! BallastCost between two trades for ship
    CostBallastStart:    dynamic array(Ship,Trade)                       of real;    ! BallastCost from start to first trade
    CostPortStart:       dynamic array(Ship,Trade,Trade)                 of real;    ! BallastCost from startport to startport
    CostTrade:           dynamic array(Ship,Trade,Way)                   of real;    ! CostTrade for ship, regardless of voyage
    CostTradeSpot:       dynamic array(Trade)                            of real;    ! Cost of hiring a spotship to do trade
    IncomeTrade:         dynamic array(Ship,Trade,Way)                   of real;    ! IncomeTrade for ship, regardless of voyage
    TimeBallast:         dynamic array(Ship,Trade,Trade)                 of real;    ! BallastTime between two trades for ship
    TimeBallastStart:    dynamic array(Ship,Trade)                       of real;    ! BallastTime from start to first trade
    TimeEarliest:        dynamic array(Trade,Voyage)                     of real;    ! Earliest start for voyage of trade
    TimeEarliestStart:   dynamic array(Ship)                             of real;    ! Earliest start for ship
    TimeLatest:          dynamic array(Trade,Voyage)                     of real;    ! Latest start for voyage of trade
    TimeTradeSpread:     dynamic array(Trade)                            of real;    ! Time Spread for each trade
    TimePortStart:       dynamic array(Ship,Trade,Trade)                 of real;    ! BallastTime from startport to startport
    TimeTrade:           dynamic array(Ship,Trade,Way)                   of real;    ! TimeTrade for ship, regardless of voyage

    !Variables
    s_ri:                dynamic array(Trade,Voyage)                       of mpvar;   ! = 1 if spotship is used on (r,i)
    t_vov:               dynamic array(Ship)                               of mpvar;   ! When ship starts from origin
    t_ri:                dynamic array(Trade,Voyage)                       of mpvar;   ! When voyage (r,i) starts
    x_vovdv:             dynamic array(Ship)                               of mpvar;   ! = 1 if ship is not in use
    x_vovri:             dynamic array(Ship,Trade,Voyage)                  of mpvar;   ! = 1 if ship sails from origin to (r,i)
    x_vridvw:            dynamic array(Ship,Trade,Voyage,Way)              of mpvar;   ! = 1 if (r,i) is last for ship in period
    x_vriujw:            dynamic array(Ship,Trade,Voyage,Trade,Voyage,Way) of mpvar;   ! = 1 if ship goes from (r,i) to (u,j)
    y_vri:               dynamic array(Ship,Trade,Voyage)                  of mpvar;   ! = 1 if ship v serves (r,i)

    ! Constraints
    FlowStart:           dynamic array(Ship)                               of linctr;
    FlowStop:            dynamic array(Ship)                               of linctr;
    FlowBalance:         dynamic array(Ship,Trade,Voyage)                  of linctr;
    VoyageServed:        dynamic array(Trade,Voyage)                       of linctr;
    TimeBetweenTrades:   dynamic array(Ship,Trade,Voyage,Trade,Voyage,Way) of linctr;
    TimeStart:           dynamic array(Ship,Trade,Voyage)                  of linctr;
    TimeSpread:          dynamic array(Trade,Voyage)                       of linctr;
    TimeStarting:        dynamic array(Ship,Trade)                         of linctr;
    ServeTrade:          dynamic array(Ship,Trade,Voyage)                  of linctr;
    WayServe:            dynamic array(Ship,Trade,Voyage)                  of linctr;

    ! Objective value
    Profit:                                                                linctr;

    ! PARAMETERS FOR TIMESPREAD-CONSTRAINT
    A:                                                          real;      ! Constant: middle of first voyage-TW
    B:                                                          real;      ! Constant: middle of last voyage-TW
    Factor:              dynamic array(Trade)                   of real;   ! Faktor, UsedTime divided by TotalTime

    ! Way Voyage
    WayVoyage:           dynamic array(Trade,Voyage)            of integer; ! Parameter: how many ways a voyage can be sailed
    WayExists:           dynamic array(Ship,Trade,Voyage,Way)   of integer; ! Does an alternative way exist?

    ! Change Time window
    ChangeWindow:        real;                                  ! Parameter to change time window
end-declarations

initializations from DataFile
    Factor;
```

```
    WayVoyage;
end-initializations

initializations from  TurboRouterFile
    CostTradeSpot;
    TimeEarliestStart;
    CostTrade;
    IncomeTrade;
    TimeTrade;
    TimeEarliest;
    TimeLatest;
    CostBallast;
    TimeBallast;
    CostPortStart;
    TimePortStart;
end-initializations


! Finding out if voyage ii of trade rr can be done in an alternative way I
forall(rr in Trade, ii in Voyage) do
    if (exists(WayVoyage(rr,ii))) then
        WayVoyage(rr,ii) := WayVoyage(rr,ii);
    else
        WayVoyage(rr,ii) := 1;
    end-if
end-do

! Finding out if voyage ii of trade rr can be done in an alternative way II
forall(vv in Ship,rr in Trade,ii in Voyage, ww in Way) do
    if (exists(TimeTrade(vv,rr,ww)) and exists(IncomeTrade(vv,rr,ww)) and exists(CostTrade(vv,rr,ww))
    and WayVoyage(rr,ii) >= ww) then
        WayExists(vv,rr,ii,ww) := 1;
    else
        WayExists(vv,rr,ii,ww) := 0;
    end-if
end-do

! Setting names in TradeNames from TradeM, TradeTC and TradeDD
forall(rr in Trade) do
    if (rr <= NumTradeM) then
        TradeNames(rr) := TradeMNames(rr);
    elif (rr > NumTradeM and rr <= (NumTradeM+NumTradeTC)) then
        TradeNames(rr) := TradeTCNames(rr);
    elif (rr > (NumTradeM+NumTradeTC)) then
        TradeNames(rr) := TradeDDNames(rr);
    end-if
end-do

! Finding time and cost for ballast travel from startport for ship v
forall(vv in Ship, rr in Trade) do
    if (StartPort(vv) > NumTrade) then
        TimeBallastStart(vv,rr) := TimeBallast(vv,StartPort(vv)-NumTrade,rr);
        CostBallastStart(vv,rr) := CostBallast(vv,StartPort(vv)-NumTrade,rr);
    elif (StartPort(vv) <= NumTrade) then
        TimeBallastStart(vv,rr) := TimePortStart(vv,StartPort(vv),rr);
        CostBallastStart(vv,rr) := CostPortStart(vv,StartPort(vv),rr);
    end-if
end-do
```

```
! Calculating time spread between voyages
forall (rr in Trade) do
    if (NumVoyage(rr) > 1) then
        A := TimeEarliest(rr,1) + 0.5 * (TimeLatest(rr,1) - TimeEarliest(rr,1));
        B := TimeEarliest(rr,NumVoyage(rr)) + 0.5 *(TimeLatest(rr,NumVoyage(rr)) - TimeEarliest(rr,NumVoyage(rr)));
        TimeTradeSpread(rr) := (B - A) / (NumVoyage(rr)-1);
    end-if
end-do

! ************************************************************************************************
! CREATING VARIABLES
! ************************************************************************************************

!Creates variable x_vovdv, binary, equals 1 if ship vv can choose not to be used. For ship that needs
!dry-docking, this variable is not created.
forall(vv in Ship) do
    if(sum(rr in TradeDD) TradeDDShip(rr,vv) = 0)then
        create(x_vovdv(vv));
        x_vovdv(vv) is_binary;
    end-if
end-do

!Creates variable x_vovri if ship makes it to start of (r,i) before the timewindow ends (NB! CAN WAIT!)
forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr) | not (rr in TradeDD and TradeDDShip(rr,vv) = 0)) do
    if (TimeEarliestStart(vv) + TimeBallastStart(vv,rr) <= TimeLatest(rr,ii)) then
        create(x_vovri(vv,rr,ii));
        x_vovri(vv,rr,ii) is_binary;
    end-if
end-do

! Creates variable x_vridvw, binary, equals 1 if (rr,ii) is the last voyage in timeperiod
forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr), ww in Way| not (rr in TradeDD and TradeDDShip(rr,vv) = 0)) do
    if (WayExists(vv,rr,ii,ww) > 0.5) then
        create(x_vridvw(vv,rr,ii,ww));
        x_vridvw(vv,rr,ii,ww) is_binary;
    end-if
end-do

!Creates variable x_vriujw if ship makes it from (r,i) to (u,j) before the timewindow closes.
!Ship can come and wait.
forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr),uu in Trade, jj in 1..NumVoyage(uu), ww in Way
| not (rr in TradeDD and TradeDDShip(rr,vv) = 0) and not (uu in TradeDD and TradeDDShip(uu,vv) = 0) ) do
    if (TimeEarliest(rr,ii) + TimeTrade(vv,rr,ww) + TimeBallast(vv,rr,uu) <= TimeLatest(uu,jj)
    and WayExists(vv,rr,ii,ww) > 0.5) then
        create(x_vriujw(vv,rr,ii,uu,jj,ww));
        x_vriujw(vv,rr,ii,uu,jj,ww) is_binary;
    end-if
end-do

!Time to start from origin is later than earliest possible start for ship v
forall(vv in Ship) do
    create(t_vov(vv));
    t_vov(vv) >= TimeEarliestStart(vv);
end-do

!Voyage (r,i) has to start within the timewindow
forall(rr in Trade, ii in 1..NumVoyage(rr)) do
    create(t_ri(rr,ii));
    t_ri(rr,ii) >= TimeEarliest(rr,ii);
```

```
    t_ri(rr,ii) <= TimeLatest(rr,ii);
end-do


!Creates spotship-variable for voyages of trades, except dry-docking,
!those have to be done by their own ships.
forall(rr in Trade, ii in 1..NumVoyage(rr) | not rr in TradeDD ) do
    create(s_ri(rr,ii));
    s_ri(rr,ii) is_binary;
end-do

!Creates binaryvariable, equals 1 if ship v does voyage and trade (rr,ii)
forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr) | exists(x_vovri(vv,rr,ii))) do
    if not (rr in TradeDD and TradeDDShip(rr,vv) = 0) then
        create(y_vri(vv,rr,ii));
    end-if
end-do

! ***************************************************************************************************
! CONSTRAINTS
! ***************************************************************************************************

! FLOW CONSTRAINTS
!All ships have to leave origin or they are not in use
forall(vv in Ship) do
    FlowStart(vv) :=
    x_vovdv(vv) + sum(rr in Trade,ii in 1..NumVoyage(rr)) x_vovri(vv,rr,ii) = 1;
end-do

!All ships have to reach destination or they are not in use
forall(vv in Ship) do
    FlowStop(vv) :=
    x_vovdv(vv) + sum(rr in Trade,ii in 1..NumVoyage(rr),ww in Way) x_vridvw(vv,rr,ii,ww) = 1;
end-do

!Nodebalance - every ship that sails a voyage has to come from a place and finish it.
forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr)) do
    FlowBalance(vv,rr,ii) :=
    sum(uu in Trade, jj in 1..NumVoyage(uu), ww in Way) x_vriujw(vv,rr,ii,uu,jj,ww) + sum(ww in Way) x_vridvw(vv,rr,ii,ww) -
    sum(uu in Trade, jj in 1..NumVoyage(uu), ww in Way) x_vriujw(vv,uu,jj,rr,ii,ww) - x_vovri(vv,rr,ii) = 0;
end-do

! Constraint to get y_vri
forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr)) do
    ServeTrade(vv,rr,ii) :=
    y_vri(vv,rr,ii) = sum(uu in Trade, jj in 1..NumVoyage(uu),ww in Way) x_vriujw(vv,rr,ii,uu,jj,ww) +
    sum(ww in Way) x_vridvw(vv,rr,ii,ww);
end-do


! VOYAGE CONSTRAINTS: THREE CONSTRAINTS HAVE BEEN PUT TOGETHER TO ONE.
!Every "mandatory" voyage has to be done either with their own ship or by a spotship. The TC(optional)-trades
!can be done (not with spotship). The ships that need dry dock has to complete the dry-dock voyage. s_ri(rr,ii) for TC-ships
!are defined, but the cost and income is 0 and it has therefore no influence on the objective function. s_ri(rr,ii) for
!dry-dockings are not defined.
forall(rr in Trade, ii in 1..NumVoyage(rr)) do
    VoyageServed(rr,ii) :=
    sum(vv in Ship) y_vri(vv,rr,ii) + s_ri(rr,ii) = 1;
end-do
```

```
! TIME CONSTRAINTS
! Start time for voyage (u,j) cannot be before (r,i) is completed if ship v is doing both
forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr), uu in Trade, jj in 1..NumVoyage(uu), ww in Way |
exists(x_vriujw(vv,rr,ii,uu,jj,ww))) do
    TimeBetweenTrades(vv,rr,ii,uu,jj,ww) :=
    t_ri(rr,ii) + TimeTrade(vv,rr,ww) + TimeBallast(vv,rr,uu) - t_ri(uu,jj) <=
    ((TimeLatest(rr,ii) + TimeTrade(vv,rr,ww) + TimeBallast(vv,rr,uu) - TimeEarliest(uu,jj)) *
    (1 - x_vriujw(vv,rr,ii,uu,jj,ww)));
end-do

! Start time for voyage (r,i) cannot be before ship reaches the startpoint from origin
forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr) | exists(x_vovri(vv,rr,ii))) do
    TimeStart(vv,rr,ii) :=
    t_vov(vv) + TimeBallastStart(vv,rr) - t_ri(rr,ii) <=
    ((TimeEarliestStart(vv) + TimeBallastStart(vv,rr)) * (1 - x_vovri(vv,rr,ii)));
end-do

! Starttime for voyage (r,i) has to be after starttime for voyage (r,i-1) and the voyages have to fairly evenly spread
forall(rr in Trade, ii in 2..NumVoyage(rr)) do
    TimeSpread(rr,ii) :=
    t_ri(rr,ii) >= t_ri(rr,ii-1) + Factor(rr) * TimeTradeSpread(rr);
end-do


! Ensuring only sailing one way
forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr)) do
    WayServe(vv,rr,ii) :=
    sum(uu in Trade, jj in 1..NumVoyage(uu),ww in Way) x_vriujw(vv,rr,ii,uu,jj,ww) + sum(ww in Way) x_vridvw(vv,rr,ii,ww) <= 1;
end-do

! ************************************************************************************************
! Objective function - maximizing profit
! ************************************************************************************************

Profit :=
sum(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr), uu in Trade, jj in 1..NumVoyage(uu), ww in Way)
(IncomeTrade(vv,rr,ww) - CostTrade(vv,rr,ww) - CostBallast(vv,rr,uu)) * x_vriujw(vv,rr,ii,uu,jj,ww)
+ sum(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr), ww in Way)
(IncomeTrade(vv,rr,ww) - CostTrade(vv,rr,ww)) * x_vridvw(vv,rr,ii,ww)
- sum(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr)) (CostBallastStart(vv,rr) * x_vovri(vv,rr,ii))
- sum(rr in Trade, ii in 1..NumVoyage(rr)) (CostTradeSpot(rr) * s_ri(rr,ii));



! ************************************************************************************************
! Setting parameters of the optimization
! ************************************************************************************************

setparam("XPRS_VERBOSE",TRUE);

if (IPRECUT = 0) then
    setparam("XPRS_PRESOLVE",0);
    setparam("XPRS_CUTSTRATEGY",0);
end-if

if (ICPU <> 0) then
    !setparam("XPRS_MAXTIME", -30);
```

```
    setparam("XPRS_MAXTIME", ICPU);
end-if

! ************************************************************************************************
! Optimization : Maximizing profit
! ************************************************************************************************

maximize(Profit);



! ************************************************************************************
! WRITING OUT TO FILE
! ************************************************************************************

declarations
    x_vovdvValues:      dynamic array(Ship)                             of real;
    t_vovValues:        dynamic array(Ship)                             of real;
    x_vovriValues:      dynamic array(Ship,Trade,Voyage)                of real;
    x_vridvwValues:     dynamic array(Ship,Trade,Voyage)                of real;
    x_vriujwValues:     dynamic array(Ship,Trade,Voyage,Trade,Voyage)   of real;
    y_vriValues:        dynamic array(Ship,Trade,Voyage)                of real;
    s_riValues:         dynamic array(Trade,Voyage)                     of real;
    t_riValues:         dynamic array(Trade,Voyage)                     of real;

    xTime_start:        dynamic array(Ship,Trade,Voyage)                of real;
    xTime_flow:         dynamic array(Ship,Trade,Voyage)                of real;
    xTime_last:         dynamic array(Ship,Trade,Voyage)                of real;
    xTime_spot:         dynamic array(Trade,Voyage)                     of real;

    TimeSpreadTrade:    dynamic array(Trade,Voyage)                     of real;
    Duration:           dynamic array(Trade,Voyage)                     of real;
    ProfitTrade:        dynamic array(Trade,Voyage)                     of real;
    CostBallastTrade:   dynamic array(Trade,Voyage)                     of real;
    TimeBallastTrade:   dynamic array(Trade,Voyage)                     of real;
    PreviousTrade:      dynamic array(Trade,Voyage)                     of string;

    IncomeTradeWay:     dynamic array(Ship,Trade)                       of real;
    CostTradeWay:       dynamic array(Ship,Trade)                       of real;
    TimeTradeWay:       dynamic array(Ship,Trade)                       of real;
    WayChosen:          dynamic array(Ship,Trade)                       of real;

    BestBound:          real;
    Gap:                real;
    ProfitValue:        real;
    SolutionTime:       real;

    StartTime:          dynamic array(Ship)                             of real;
    FirstTrade:         dynamic array(Ship)                             of integer;
    FirstVoyage:        dynamic array(Ship)                             of integer;
    LastTrade:          dynamic array(Ship)                             of integer;
    LastVoyage:         dynamic array(Ship)                             of integer;
    FinishTime:         dynamic array(Ship)                             of real;
    ShipDurationOnSea:  dynamic array(Ship)                             of real;
    EndTimeOfTrade:     dynamic array(Trade,Voyage)                     of real;
    DaysOutOfPeriod:    dynamic array(Trade,Voyage)                     of real;
    ProfitOutOfPeriod:  dynamic array(Ship)                             of real;
    TimeOfTrade:        dynamic array(Trade,Voyage)                     of real;
```

```
    StartTrade:                 integer;
    StartVoyage:                integer;
    FinishTrade:                integer;
    FinishVoyage:               integer;
    TotalDurationOnSea:         real;
    ProfitPerDayOnSea:          real;
    StartTimeOfPeriod:          real;
    EndDayOfPeriod:             real;
    ProfitTooLate:              real;
    RealProfitInPeriod:         real;
    TotalTimeOfTrade:           real;
    TotalNumberOfShip:          real;
    ProfitPerShip:              real;
    TotalBallastCosts:          real;
    TotalBallastTime:           real;
    TotalIncome:                real;
    TotalCosts:                 real;
    TotalSpotCosts:             real;
    TotalTradeCosts:            real;
    PeriodLength:               real;
    TotalNumberOfTC:            real;
    NumberOfDD:                 real;
    TotalBallastTimeDD:         real;
    TotalNumberOfSpotShip:      real;
    TotalBallastCostsDD:        real;
    TotalNumberOfVoyages:       real;
    TotalDaysOutOfPeriod:       real;
    ProfitTooLatePerDay:        real;
    RealProfitInPeriodPerShip:  real;

    temp:           string;
    FileString:     string;
end-declarations

ProfitValue := getobjval;
BestBound   := getparam("XPRS_bestbound");

forall(vv in Ship) do
    x_vovdvValues(vv) := getsol(x_vovdv(vv));
end-do

forall(vv in Ship) do
    t_vovValues(vv) := getsol(t_vov(vv));
end-do

forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr))do
    x_vovriValues(vv,rr,ii) := getsol(x_vovri(vv,rr,ii));
end-do

forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr))do
    x_vridvwValues(vv,rr,ii) := sum(ww in Way) getsol(x_vridvw(vv,rr,ii,ww));
end-do

forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr), uu in Trade, jj in 1..NumVoyage(uu)) do
    x_vriujwValues(vv,rr,ii,uu,jj) := sum(ww in Way) getsol(x_vriujw(vv,rr,ii,uu,jj,ww));
end-do

forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr))do
    y_vriValues(vv,rr,ii) := getsol(y_vri(vv,rr,ii));
```

```
end-do

forall(rr in Trade, ii in 1..NumVoyage(rr)) do
    s_riValues(rr,ii) := getsol(s_ri(rr,ii));
end-do

forall(rr in Trade, ii in 1..NumVoyage(rr))do
    t_riValues(rr,ii) := getsol(t_ri(rr,ii));
end-do

forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr))do
    if (x_vovriValues(vv,rr,ii) > 0.5) then
        xTime_start(vv,rr,ii) := getsol(t_ri(rr,ii));
    end-if
end-do

forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr), uu in Trade, jj in 1..NumVoyage(uu)) do
    if (x_vriujwValues(vv,rr,ii,uu,jj) > 0.5) then
        xTime_flow(vv,rr,ii) := getsol(t_ri(rr,ii));
    end-if
end-do

forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr)) do
    if (x_vridvwValues(vv,rr,ii) > 0.5) then
        xTime_last(vv,rr,ii) := getsol(t_ri(rr,ii));
    end-if
end-do

forall(rr in Trade, ii in 1..NumVoyage(rr))do
    if (s_riValues(rr,ii) > 0.5) then
        xTime_spot(rr,ii) := getsol(t_ri(rr,ii));
    end-if
end-do

forall(rr in Trade, ii in 2..NumVoyage(rr)) do
    TimeSpreadTrade(rr,ii) := t_riValues(rr,ii) - t_riValues(rr,ii-1);
end-do

forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr), uu in Trade, jj in 1..NumVoyage(uu)) do
    if (y_vriValues(vv,uu,jj) > 0.5) then
        if (x_vriujwValues(vv,rr,ii,uu,jj) > 0.5) then
            CostBallastTrade(uu,jj) := CostBallast(vv,rr,uu);
            TimeBallastTrade(uu,jj) := TimeBallast(vv,rr,uu);
        elif (x_vovriValues(vv,uu,jj) > 0.5) then
            CostBallastTrade(uu,jj) := CostBallastStart(vv,uu);
            TimeBallastTrade(uu,jj) := TimeBallastStart(vv,uu);
        end-if
    end-if
end-do

forall(vv in Ship,rr in Trade,ii in 1..NumVoyage(rr),uu in Trade, jj in 1..NumVoyage(uu)) do
    if (x_vriujwValues(vv,rr,ii,uu,jj) > 0.5) then
        PreviousTrade(uu,jj) := TradeNames(rr)+" "+ii;
    end-if
end-do

forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr),uu in Trade, jj in 1..NumVoyage(uu),ww in Way) do
    if (getsol(x_vriujw(vv,rr,ii,uu,jj,ww)) > 0.5) then
        IncomeTradeWay(vv,rr)   := IncomeTrade(vv,rr,ww);
```

```
        CostTradeWay(vv,rr)     := CostTrade(vv,rr,ww);
        TimeTradeWay(vv,rr)     := TimeTrade(vv,rr,ww);
        WayChosen(vv,rr)        := ww;
    elif (getsol(x_vridvw(vv,rr,ii,ww)) > 0.5) then
        IncomeTradeWay(vv,rr)   := IncomeTrade(vv,rr,ww);
        CostTradeWay(vv,rr)     := CostTrade(vv,rr,ww);
        TimeTradeWay(vv,rr)     := TimeTrade(vv,rr,ww);
        WayChosen(vv,rr)        := ww;
    end-if
end-do

forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr),uu in Trade, jj in 1..NumVoyage(uu),ww in Way) do
    if (getsol(x_vriujw(vv,rr,ii,uu,jj,ww)) > 0.5 or getsol(x_vridvw(vv,rr,ii,ww)) > 0.5) then
        Duration(rr,ii) := t_riValues(rr,ii) + TimeTradeWay(vv,rr);
        ProfitTrade(rr,ii) := IncomeTradeWay(vv,rr) - CostTradeWay(vv,rr) - CostBallastTrade(rr,ii);
    elif (s_riValues(rr,ii) > 0.5) then
        ProfitTrade(rr,ii) := -CostTradeSpot(rr);
    end-if
end-do

forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr)) do
    if (x_vovriValues(vv,rr,ii) > 0.5) then
        FirstTrade(vv) := rr;
        FirstVoyage(vv) := ii;
    end-if
    if (x_vridvwValues(vv,rr,ii) > 0.5) then
        LastTrade(vv) := rr;
        LastVoyage(vv) := ii;
    end-if
end-do


forall(vv in Ship) do
    if (x_vovdvValues(vv) < 0.5) then
        StartTrade := FirstTrade(vv);
        StartVoyage := FirstVoyage(vv);
        StartTime(vv) := t_riValues(StartTrade,StartVoyage) - TimeBallastStart(vv,StartTrade);
        FinishTrade := LastTrade(vv);
        FinishVoyage := LastVoyage(vv);
        FinishTime(vv) := t_riValues(LastTrade(vv),LastVoyage(vv)) + TimeTradeWay(vv,LastTrade(vv));
        ShipDurationOnSea(vv) := FinishTime(vv) - StartTime(vv);
    end-if
end-do

forall(vv in Ship) do
    if (ShipNames(vv) = 'PI') then
        StartTimeOfPeriod := t_vovValues(vv);
    end-if
end-do

! The length of period is an input here. Since the analysis is conducted on the 90 days instances,
! the length is automatically set to 90 days. If considering different time periods, this should have been
! an input to the model.

PeriodLength := 90;
EndDayOfPeriod := StartTimeOfPeriod + PeriodLength;

forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr)) do
    if(y_vriValues(vv,rr,ii) > 0.5) then
```

```
        EndTimeOfTrade(rr,ii) := t_riValues(rr,ii) + TimeTradeWay(vv,rr);
        ! PROFIT THAT IS NOT IN THIS PERIOD: HAVE TO GET PER DAY!!!
        DaysOutOfPeriod(rr,ii) := EndTimeOfTrade(rr,ii) - EndDayOfPeriod;
        ! TOTAL TIME OF TRADE INCLUDED BALLASTVOYAGE BEFORE
        TimeOfTrade(rr,ii) := TimeTradeWay(vv,rr) + TimeBallastTrade(rr,ii);
    end-if
end-do

TotalDaysOutOfPeriod := 0;
forall(rr in Trade, ii in 1..NumVoyage(rr)) do
    if (DaysOutOfPeriod(rr,ii) > 0) then
        TotalDaysOutOfPeriod := TotalDaysOutOfPeriod + DaysOutOfPeriod(rr,ii);
    end-if
end-do

TotalNumberOfShip := 0;
forall(vv in Ship) do
    if (x_vovdvValues(vv) < 0.5) then
        TotalNumberOfShip := TotalNumberOfShip + 1;
    end-if
end-do

TotalNumberOfTC := 0;
forall(vv in Ship, rr in TradeTC, ii in 1..NumVoyage(rr)) do
    if (y_vriValues(vv,rr,ii) > 0.5) then
        TotalNumberOfTC := TotalNumberOfTC + 1;
    end-if
end-do

TotalNumberOfVoyages := 0;
forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr)) do
    if (y_vriValues(vv,rr,ii) > 0.5) then
        TotalNumberOfVoyages := TotalNumberOfVoyages + 1;
    end-if
end-do



TotalIncome := 0;
TotalCosts := 0;
TotalTradeCosts := 0;
TotalBallastCosts := 0;
TotalBallastTime := 0;
forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr)) do
    if (y_vriValues(vv,rr,ii) > 0.5) then
        TotalIncome := TotalIncome + IncomeTradeWay(vv,rr);
        TotalCosts := TotalCosts + CostTradeWay(vv,rr)+CostBallastTrade(rr,ii);
        TotalTradeCosts := TotalTradeCosts + CostTradeWay(vv,rr);
        TotalBallastCosts := TotalBallastCosts + CostBallastTrade(rr,ii);
        TotalBallastTime := TotalBallastTime + TimeBallastTrade(rr,ii);
    end-if
end-do

TotalSpotCosts := 0;
TotalNumberOfSpotShip := 0;
forall(rr in TradeM, ii in 1..NumVoyage(rr)) do
    if (s_riValues(rr,ii) > 0.5) then
        TotalSpotCosts := TotalSpotCosts + CostTradeSpot(rr);
        TotalNumberOfSpotShip := TotalNumberOfSpotShip + 1;
```

```
    end-if
end-do

NumberOfDD := 0;
TotalBallastCostsDD := 0;
TotalBallastTimeDD := 0;
forall(vv in Ship, rr in TradeDD, ii in 1..NumVoyage(rr)) do
    if (y_vriValues(vv,rr,ii) > 0.5) then
        NumberOfDD := NumberOfDD + 1;
        TotalBallastCostsDD := TotalBallastCostsDD + CostBallastTrade(rr,ii);
        TotalBallastTimeDD := TotalBallastTimeDD + TimeBallastTrade(rr,ii);
    end-if
end-do


TotalTimeOfTrade := sum(rr in Trade, ii in 1..NumVoyage(rr)) TimeOfTrade(rr,ii);
TotalDurationOnSea := sum(vv in Ship)ShipDurationOnSea(vv);
ProfitPerDayOnSea := ProfitValue/TotalDurationOnSea;
ProfitTooLate := ProfitPerDayOnSea * TotalDaysOutOfPeriod;
RealProfitInPeriod := ProfitPerDayOnSea * (TotalDurationOnSea-TotalDaysOutOfPeriod);
RealProfitInPeriodPerShip := RealProfitInPeriod / TotalNumberOfShip;
ProfitPerShip := ProfitValue / TotalNumberOfShip;

! ***************************************************************************************************
! WRITING TO FILE
! ***************************************************************************************************

!Creating a table which is written to a .csv file to make it compatible with Excel
!Write case info
FileString := " "+MODELNAME;
FileString += " "+DATETIME;
FileString += " "+DataFile;

! Calculating solution time
SolutionTime := gettime - Time;

makedir("output/"+FileString);

fopen("output/"+FileString+"/ flow.csv", F_OUTPUT);

writeln("ECONOMIC ANALYSIS: ");
!Writing information about timeperiod
writeln("TIME OF PERIOD: ");
writeln("StartTimeOfPeriod: ,",StartTimeOfPeriod, " days");
writeln("Duration: , " , PeriodLength, " days");    ! This analysis is foremost for the 90 day period an
writeln("EndTimeOfPeriod: ,", EndDayOfPeriod, " days");
writeln("");

!Writing objective value, BestBound and Gap
writeln("Profit: ,","$ ",ProfitValue);

if (ProfitValue <> 0) then
    Gap := (1 - BestBound/ProfitValue) * 100;
else
    Gap := 0;
end-if

writeln("Best bound: ,","$ ", BestBound);
writeln("Gap: ,","" , Gap, " %");
```

```
!Writing solution time
writeln("Solution time: ,","", SolutionTime, " sec");

!Writing profit calculations
writeln("");
writeln("PROFIT CALCULATIONS: ");
writeln("ProfitValue: , $ ",ProfitValue);
writeln("TotalDurationOnSea: ,",TotalDurationOnSea, ", days");
writeln("Total Time of Trade: ,",TotalTimeOfTrade, ", days");
writeln("ProfitPerDayOnSea: , $ ", ProfitPerDayOnSea);
writeln("TotalDaysOutOfPeriod: ,", TotalDaysOutOfPeriod, ", days");
writeln("ProfitTooLate: , $ " ,ProfitTooLate);
writeln("RealProfit in Period: , $ ",RealProfitInPeriod);
writeln("");
writeln("TotalNumberOfShips: ,",TotalNumberOfShip);
writeln("TotalNumberOfTC-trades: ,",TotalNumberOfTC);
writeln("TotalNumberOfSpotShip: ," , TotalNumberOfSpotShip);
writeln("TotalNumberOfVoyages: ,",TotalNumberOfVoyages);
writeln("TotalNumberOfVoyages/on sea: ,",TotalNumberOfVoyages/TotalDurationOnSea);
writeln("TotalNumberOfVoyages/daysperiod: ,",TotalNumberOfVoyages/PeriodLength);
writeln("TotalNumberOfVoyages/total time of trade: ,",TotalNumberOfVoyages/TotalTimeOfTrade);
writeln("");
writeln("RealProfitInPeriodPerShip: , $ ",RealProfitInPeriodPerShip);
writeln("ProfitPerShip: , $ ",ProfitPerShip);
writeln("");
writeln("Total Income: , $ " ,TotalIncome);
writeln("Total Costs: , $ " , TotalCosts);
writeln("Total Trade Costs: , $ ",TotalTradeCosts);
writeln("Total Spot Costs: , $ ",TotalSpotCosts);
writeln("Total Ballast Costs: , $ ",TotalBallastCosts);
writeln("Total Ballast Time: ,", TotalBallastTime, ", days");

writeln("");
writeln("DRY DOCKINGS: ");
writeln("Number of dry dockings : , ", NumberOfDD);
writeln("TotalBallastCostsDD: , $ ", TotalBallastCostsDD);
writeln("TotalBallastTimeDD: ," ,TotalBallastTimeDD, ", days");
writeln("");


writeln("SHIP AND TRADE MATRIX");
temp := "Ship,Use,Start-up time,";
forall(rr in Trade, ii in 1..NumVoyage(rr))do
    temp += "" +TradeNames(rr)+ii+",";
end-do

!Writing solution to file
writeln(temp);
forall(vv in Ship) do
    temp := ""+ShipNames(vv)+",";
    if (x_vovdvValues(vv) > 0.5) then
        temp += "NOT IN USE,";
    else
        temp += "IN USE,";
    end-if
    temp += "" +t_vovValues(vv)+",";
    forall(rr in Trade, ii in 1..NumVoyage(rr)) do
        if (exists(xTime_start(vv,rr,ii))) then
```

```
            temp += ""+xTime_start(vv,rr,ii);
        elif (exists(xTime_flow(vv,rr,ii))) then
            temp += ""+xTime_flow(vv,rr,ii);
        elif (exists(xTime_last(vv,rr,ii))) then
            temp += ""+xTime_last(vv,rr,ii);
        end-if
        temp += ",";
    end-do
    writeln(temp);
end-do

!Writing spot solutions to file
temp := "StartTime for SpotShip,,,";
forall(rr in TradeM, ii in 1..NumVoyage(rr)) do
    if (exists(xTime_spot(rr,ii))) then
        temp += ""+xTime_spot(rr,ii);
    end-if
    temp+= ",";
end-do
writeln(temp);

!Writing time spread to file
temp := "Actual Time spread,,,";
forall(rr in Trade) do
    temp+= ",";
    forall(ii in 2..NumVoyage(rr)) do
        temp+= ""+TimeSpreadTrade(rr,ii);
        temp+= ",";
    end-do
end-do
writeln(temp);

writeln("");
writeln("");

writeln("TRADESPECIFIC");
temp := "Trade,VoyageNumber,ShipName,StartTime,TimeTrade,StopTime,IncomeTrade,CostTrade,CostBallastTrade,
TimeBallastTrade,ProfitTrade,Factor,WayChosen";

!Writing solution to file
writeln(temp);
forall(rr in Trade, ii in 1..NumVoyage(rr) | not rr in TradeTC) do
    temp := ""+TradeNames(rr)+"," +ii+",";
    if (s_riValues(rr,ii) > 0.5) then
            temp += ""+"SpotShip"+",";
            temp += ",,,,"+CostTradeSpot(rr)+",,,"+ProfitTrade(rr,ii)+",";
    end-if
    forall(vv in Ship) do
        if (y_vriValues(vv,rr,ii) > 0.5) then
            temp += ""+ShipNames(vv)+",";
            temp += ""+t_riValues(rr,ii)+",";
            temp += ""+TimeTradeWay(vv,rr)+",";
            temp += ""+Duration(rr,ii)+",";
            temp += ""+IncomeTradeWay(vv,rr)+",";
            temp += ""+CostTradeWay(vv,rr)+",";
            temp += ""+CostBallastTrade(rr,ii)+",";
            temp += ""+TimeBallastTrade(rr,ii)+",";
            temp += ""+ProfitTrade(rr,ii)+",";
            temp += ""+Factor(rr)+",";
```

```
            temp += ""+WayChosen(vv,rr)+",";
        end-if
    end-do
    writeln(temp);
end-do

forall(rr in TradeTC, ii in 1..NumVoyage(rr)) do
    forall(vv in Ship) do
        if (y_vriValues(vv,rr,ii) > 0.5) then
            temp := ""+TradeNames(rr)+"," +ii+",";
            temp += ""+ShipNames(vv)+",";
            temp += ""+t_riValues(rr,ii)+",";
            temp += ""+TimeTradeWay(vv,rr)+",";
            temp += ""+Duration(rr,ii)+",";
            temp += ""+IncomeTradeWay(vv,rr)+",";
            temp += ""+CostTradeWay(vv,rr)+",";
            temp += ""+CostBallastTrade(rr,ii)+",";
            temp += ""+TimeBallastTrade(rr,ii)+",";
            temp += ""+ProfitTrade(rr,ii)+",";
            temp += ""+Factor(rr)+",";
            temp += ""+WayChosen(vv,rr)+",";
            writeln(temp);
        end-if
    end-do
end-do

writeln("");
writeln("");


writeln("SHIPSPECIFIC");

forall(vv in Ship) do
    temp := "ShipName: ,"+ShipNames(vv)+",";
    writeln(temp);
    temp := "Start: ,"+t_vovValues(vv)+",";
    writeln(temp);
    temp := "TradeName,VoyageNumber,PreviousTrade,StartTime,TimeTrade,StopTime,IncomeTrade,CostTrade,CostBallastTrade,
    TimeBallastTrade,ProfitTrade,Factor,WayChosen";
    writeln(temp);
    forall(rr in Trade, ii in 1..NumVoyage(rr)) do
        if (y_vriValues(vv,rr,ii) > 0.5) then
            temp := ""+TradeNames(rr)+",";
            temp += ""+ii+",";
            temp += ""+PreviousTrade(rr,ii)+",";
            temp += ""+t_riValues(rr,ii)+",";
            temp += ""+TimeTradeWay(vv,rr)+",";
            temp += ""+Duration(rr,ii)+",";
            temp += ""+IncomeTradeWay(vv,rr)+",";
            temp += ""+CostTradeWay(vv,rr)+",";
            temp += ""+CostBallastTrade(rr,ii)+",";
            temp += ""+TimeBallastTrade(rr,ii)+",";
            temp += ""+ProfitTrade(rr,ii)+",";
            temp += ""+Factor(rr)+",";
            temp += ""+WayChosen(vv,rr)+",";
            writeln(temp);
        end-if
    end-do
    writeln("");
```

```
end-do

writeln("");
writeln("DURATION ON SEA: ");
forall(vv in Ship) do
    temp := "ShipName: ,"+ShipNames(vv)+",";
    writeln(temp);
    temp := "StartTime: ,"+StartTime(vv)+",";
    writeln(temp);
    temp := "FinishTime: ,"+FinishTime(vv)+",";
    writeln(temp);
    temp := "Duration on Sea: ,"+ShipDurationOnSea(vv)+",";
    writeln(temp);
end-do

writeln("");
writeln("");
writeln("OUT OF PERIOD");

temp := "TradeName, VoyageNumber, StartTime, TimeBallastTrade, TimeTrade, StopTime, TotalTimeOnTrade, DaysOutOfPeriod";
writeln(temp);
forall(vv in Ship, rr in Trade, ii in 1..NumVoyage(rr)) do
    if(y_vriValues(vv,rr,ii) > 0.5) then
        temp := ""+TradeNames(rr)+",";
        temp += ""+ii+",";
        temp += ""+t_riValues(rr,ii)+",";
        temp += ""+TimeBallastTrade(rr,ii)+",";
        temp += ""+TimeTradeWay(vv,rr)+",";
        temp += ""+Duration(rr,ii)+",";
        temp += ""+TimeOfTrade(rr,ii)+",";
        temp += ""+DaysOutOfPeriod(rr,ii)+",";
        writeln(temp);
    end-if
end-do

writeln("");
writeln("");


writeln("SPOTSHIP");

temp := "TradeName, VoyageNumber, ProfitTrade";
writeln(temp);
forall(rr in TradeM, ii in 1..NumVoyage(rr)) do
    if (s_riValues(rr,ii) > 0.5) then
        temp := ""+TradeNames(rr)+",";
        temp += ""+ii+",";
        temp += ""+ProfitTrade(rr,ii)+",";
        writeln(temp);
    end-if
end-do

fclose(F_OUTPUT);


end-model
```