Liste des annexes

Annexe 1 : Programmation Mosel Modèle SCND distance

```
!@encoding CP1252
model "Distance totale Multi-produits"
uses "mmxprs"; !gain access to the Xpress-Optimizer solver
!optional parameters section
parameters
	P=3;			! Nombre de dépôts à ouvrir
end-parameters

declarations
Clients=1..170;
Depots=1..18;
Distance:array(Depots,Clients) of integer;
Demand: array(Clients) of integer;
Demand2: array(Clients) of integer
x:array(Clients) of mpvar;
y:array(Depots, Clients) of mpvar;
Objective:linctr;

end-declarations

procedure LoadProblem
	Objective:=sum(d in Depots, c in Clients) Distance (d,c) * (Demand(c) + Demand2(c)) * y(d,c)
	forall (c in Clients) sum (d in Depots) y(d,c)=1
	sum(d in Depots) x(d) = P
	forall (d in Depots, c in Clients) y(d,c) <= x(d)
	forall (d in Depots) x(d) is_binary
	forall (d in Depots, c in Clients) y(d,c) is_binary
end-procedure

procedure Main
!	initializations from 'Datas_Test model 1_Distance_Demand_Multi.txt'
!	initializations from 'Datas_Test model 1_Distance_Demand_Multi_2018.txt'
	initializations from 'Datas_Test model 1_Distance_Demand_Multi_2019.txt'
		Distance
		Demand
		Demand2
	end-initializations
!	writeln("Printing d...");
!	writeln(Demand);
!	writeln (Demand2);
	!writeln("Printing Distances...")
	!forall (w in Warehouses) do
	!			forall (c in Clients) write(Distance (w,c),"\t");
	!			writeln
	!end-do
	LoadProblem
!	setparam("XPRS_VERBOSE",true)
	minimize(Objective)
	writeln ("Total KM: ",getobjval)
	writeln ("Open Depots: ")
	forall (d in Depots | getsol(x(d))>0)
writeln ("Depot(", d,")=",getsol (x(d)))
end-procedure
Main
end-model
```

Annexe 2: Programmation Mosel Modèle SCND Coût

```
!@encoding CP1252
model "Coût total Multi-produits"
uses "mmxprs"; !gain access to the Xpress-Optimizer solver

!optional parameters section

parameters

        P=3;                    ! Nombre de dépôts à ouvrir

end-parameters

declarations

Clients=1..170          ! set of Clients
Depots=1..18            ! set of Depots
Distance:array(Depots,Clients) of integer;
DEMA: array(Clients) of integer;
DEMC: array(Clients) of integer;
CSHIPA: array(Depots, Clients)of real
CSHIPC: array(Depots, Clients)of real
x:array(Depots) of mpvar;
y:array(Depots, Clients) of mpvar;
Objective:linctr;

end-declarations

procedure LoadProblem

        Objective:=sum(d in Depots, c in Clients)((DEMA(c) * CSHIPA(d,c)) + (DEMC(c) * CSHIPC (d,c))) * y(d,c)
        forall (c in Clients) sum (d in Depots) y(d,c)=1
        sum(d in Depots) x(d) = P
        forall (d in Depots, c in Clients) y(d,c) <= x(d)
        forall (d in Depots) x(d) is_binary
        forall (d in Depots, c in Clients) y(d,c) is_binary

end-procedure

procedure Main

!       initializations from 'Datas_Test model 1_Cost_Demand_Multi.txt'
!       initializations from 'Datas_Test model 1_Cost_Demand_Multi_2018.txt'
        initializations from 'Datas_Test model 1_Cost_Demand_Multi_2019.txt'

                DEMA
                DEMC
                CSHIPA
                CSHIPC

        end-initializations
!       writeln("Printing d...");
!       writeln(Demand);
!       writeln (Demand2);
        !writeln("Printing Distances...")
        !forall (w in Warehouses) do
        !                       forall (c in Clients) write(Distance (w,c),"\t");
```

```
        !                       writeln
        !end-do
        LoadProblem
!       setparam("XPRS_VERBOSE",true)
        minimize(Objective)
        writeln ("Total Cost: ",getobjval)
        writeln ("Open Depots: ")
        forall (d in Depots | getsol(x(d))>0)
writeln ("x(",d,")=",getsol (x(d)))

end-procedure

Main

end-model
```

Annexe 3: Programmation Mosel Modèle ACO

```
model AntColonyOptimization_for_VRP
uses "mmxprs","mmsystem","mmjobs";                    !Directive for the optimizer. Do not touch.

parameters
        Inputfile='RandomDistanceMatrix_Week47.txt' !Input file to process
        PheromoneInitialValue=0.5                              !Initial value of the pheromon+
        PheromoneReinforcement=0.02;            !Trace of pheromone deposited
        Pheromone_evaporation_rate=0.01                !pheromone evaporation rate
        Heuristic_information_importance=2          !power for the heuristic information in the ant choice
formula
        AntColonySize=20                                        !the size of the colony includes 10 ants
        MaxRunTime=30                                           !30 sec
        Infinity=1000000000                                     !Infinity
end-parameters

(! ****************** Inputfile Reading Block ************************ !)
declarations
        Number_of_Vertices:integer;
        Number_of_Vehicles:integer;
        VehicleCapacity: integer;
        DepotNumber: integer;
        StartTime:real;
end-declarations

writeln("+Reading ", Inputfile,"...");
initializations from Inputfile
        Number_of_Vertices as 'N'
        Number_of_Vehicles as 'M'
        VehicleCapacity as 'VehicleCapacity'
        DepotNumber as 'DepotNumber'
end-initializations
writeln("+Number of Vertices: ",Number_of_Vertices);
writeln("+Number of Vehicles: ",Number_of_Vehicles);
writeln("+Vehicle Capacity: ",VehicleCapacity);
writeln("+Number of Depots: ",DepotNumber);

declarations
        V=1..Number_of_Vertices;
        DistanceMatrix:array(V,V) of integer;
        Depots=1..DepotNumber;
        DepotVector:array(Depots) of integer;
        DepotSet:set of integer;                    !the same as DepotVector, just more comfortable to use
        CustomerDemand:array(V) of integer;
end-declarations

write("+Initializing distance matrix and demands from input file...");
initializations from Inputfile
        DistanceMatrix
        DepotVector
        CustomerDemand
end-initializations
writeln("done!");
(! End Inputfile Reading Block !)


(! ******************* ACO Block ********************** !)
```

```
declarations
	ColonySize=1..AntColonySize;
	PheromoneMatrix:array(V,V) of real;

	ColonySolution: array(ColonySize, range) of list of integer;
	ColonySolutionValue:array(ColonySize) of integer;
	ToursInColony:array(ColonySize) of integer;

	Best_so_far_Solution:array(range) of list of integer;
	Tours_in_Best_so_far_Solution:integer;
	Best_so_far_Value: real;
end-declarations

procedure ResetPheromoneMatrix
	forall(i in V) PheromoneMatrix(i,i):=0;
	forall(i,j in V | i<j) do
		PheromoneMatrix(i,j):=0.5;
		PheromoneMatrix(j,i):=PheromoneMatrix(i,j);
	end-do
end-procedure

procedure ResetACOVariables
	forall(ant in ColonySize) do
		ColonySolutionValue(ant):=-1;
		ToursInColony(ant):=0;
	end-do
	Best_so_far_Value:=Infinity;
	DepotSet:={}; forall(i in Depots) DepotSet+={DepotVector(i)}; ! I create the DepotSet
	!We reset the pheromone matrix
	ResetPheromoneMatrix;
end-procedure

procedure ConstructVehicleRoutes(ant:integer)
	declarations
		Tour:list of integer;
		AntSolution:array(range) of list of integer;
		CustomersToVisit:set of integer;
		FeasibleCustomersToVisit:set of integer;
		tourcounter:integer;
		ProbabilityVisit:array(V) of real;
		Choose_a_Depot:boolean;
		CurrentNode:integer;
		NextNodeToVisit:integer;
		SelectedDepot:integer;
		CurrentCapacity:integer;
		CurrentSolutionValue:integer;
		!Tour_length:integer;
		r,temp:real;
	end-declarations
	writeln("\n\nAnt ",ant," is constructing a solution to the problem...");
	CurrentSolutionValue:=0;
	CustomersToVisit:={}; forall(i in V-DepotSet) CustomersToVisit+={i};

	tourcounter:=0;

	Choose_a_Depot:=true;
	while(CustomersToVisit<>{}) do
		if(Choose_a_Depot=true) then
```

```
                    Tour:=[];
                    tourcounter+=1;
                    !Tour_length:=0;
                    !pick a depot at random | improvement: pick it in a smarter way
                    SelectedDepot:=integer(round(random*DepotNumber));
                    if(SelectedDepot=0) then SelectedDepot:=1; end-if
                    writeln("Starting tour ",tourcounter," from depot ",SelectedDepot,"...");
                    CurrentNode:=SelectedDepot;
                    Tour:=Tour+[CurrentNode];
                    CurrentCapacity:=VehicleCapacity;
                    writeln("Current capacity of the vehicle: ",CurrentCapacity);
                    Choose_a_Depot:=false;
                end-if
                !Select only the Customers that do not violate the current vehicle capacity
                FeasibleCustomersToVisit:={};
                forall(j in CustomersToVisit | CurrentCapacity - CustomerDemand(j) >0)
FeasibleCustomersToVisit+={j};
                if(FeasibleCustomersToVisit <>{}) then
                    forall(j in FeasibleCustomersToVisit) do
                        ProbabilityVisit(j):=PheromoneMatrix(CurrentNode,j) *
(1/DistanceMatrix(CurrentNode,j))^Heuristic_information_importance / sum(k in FeasibleCustomersToVisit)
PheromoneMatrix(CurrentNode,k) * (1/DistanceMatrix(CurrentNode,k))^Heuristic_information_importance;
                    end-do
                    !Now let's pick a customer
                    r:=random;
                    temp:=0;
                    forall(j in FeasibleCustomersToVisit) do
                        NextNodeToVisit:=j;
                        temp+=ProbabilityVisit(j);
                        if temp >=r then break; end-if
                    end-do
                    writeln("Selected customer: ",NextNodeToVisit);
                    writeln("Demand of the select customer: ",CustomerDemand(NextNodeToVisit));
                    Tour += [NextNodeToVisit];
                    CurrentCapacity:=CurrentCapacity-CustomerDemand(NextNodeToVisit);
                    writeln("Distance
(",CurrentNode,",",NextNodeToVisit,")=",DistanceMatrix(CurrentNode,NextNodeToVisit));
                    CurrentSolutionValue+=DistanceMatrix(CurrentNode,NextNodeToVisit);
                    !Tour_length+=DistanceMatrix(CurrentNode,NextNodeToVisit);
                    CurrentNode:=NextNodeToVisit;
                    CustomersToVisit-={NextNodeToVisit};
                    writeln("Tour ",tourcounter,": ",Tour);
                    writeln("Current capacity of the vehicle: ",CurrentCapacity);
                    writeln("Current SolutionValue: ",CurrentSolutionValue);
                else
                    writeln("No more feasible customer to visit; coming back to depot ",SelectedDepot);
                    Tour := Tour + [SelectedDepot];
                    write("\n\nOverview of Tour ",tourcounter,": ",Tour,"\t\t");
                    CurrentSolutionValue+=DistanceMatrix(CurrentNode,SelectedDepot);
                    !Tour_length+=DistanceMatrix(CurrentNode,SelectedDepot);
                    !writeln("Tour length: ",Tour_length);
                    AntSolution(tourcounter):=Tour;
                    Choose_a_Depot:=true;
                end-if
        end-do

        !add last tour to the ant solution
        if Tour<>[] then
```

```
                Tour := Tour + [SelectedDepot];
                CurrentSolutionValue+=DistanceMatrix(CurrentNode,SelectedDepot);
                !writeln("===Current solution value: ",CurrentSolutionValue);
                !writeln("===Last tour: ",Tour,"\t====> Tour length: ",Compute_length_of_a_Tour(Tour));
                AntSolution(tourcounter):=Tour;

        end-if

        !Now let's add the solution built by the ant to the Colony
        ColonySolutionValue(ant):=CurrentSolutionValue;
        forall(i in 1..tourcounter) ColonySolution(ant,i):=AntSolution(i);
        ToursInColony(ant):=tourcounter;

        writeln("\nAnt ",ant," built a vehicle routing having total cost: ",ColonySolutionValue(ant));
        writeln("Printing the solution...");
        forall(i in 1..tourcounter) writeln("Tour ",i,": ",AntSolution(i));
        !Verify(ColonySolutionValue(ant), tourcounter, AntSolution);

end-procedure

procedure SelectBestSolution_of_the_Iteration
        declarations
                mymin:integer;
                ant_min:integer;
                Tour:array(range) of integer;
                mysize:integer;
        end-declarations
        ant_min:=-1; mymin:=Infinity;
        forall(ant in ColonySize) do
                writeln("Value of the solution found by the ",ant,"th ant: ",ColonySolutionValue(ant));
                if mymin > ColonySolutionValue(ant) then mymin:=ColonySolutionValue(ant); ant_min:=ant; end-if
        end-do
        writeln("\nThe best solution found by the ",ant_min,"th ant has value: ",mymin);

        !Let's check if we found a better primal bound
        if(mymin < Best_so_far_Value) then
                writeln("++New primal bound found having value ",textfmt(mymin,12),"\tTime: ",textfmt(gettime-
StartTime,10)," sec");
                Best_so_far_Value:=mymin;
                Tours_in_Best_so_far_Solution:=ToursInColony(ant_min);
                forall(t in 1..Tours_in_Best_so_far_Solution) Best_so_far_Solution(t):=ColonySolution(ant_min,t);
        end-if

        !Now let's update the pheromone
        forall(i,j in V | i<>j and not(i in DepotSet and j in DepotSet)) do
                PheromoneMatrix(i,j):=(1-Pheromone_evaporation_rate)*PheromoneMatrix(i,j) +
Pheromone_evaporation_rate*PheromoneInitialValue;
        end-do

        !Now reinforce the pheromone on the arcs belonging to the best_so_far_solution or ant_min
        forall(t in 1..Tours_in_Best_so_far_Solution) do
                mysize:=0;
                forall(k in Best_so_far_Solution(t)) do
                        mysize+=1; Tour(mysize):=k;
                end-do
                forall(q in 1..mysize-1) PheromoneMatrix(Tour(q),Tour(q+1))+=PheromoneReinforcement;
        end-do
end-procedure
```

```
procedure Main
	StartTime:=gettime;
	ResetACOVariables;
	writeln("+Starting Ant Colony Optimization, by assuming as a maximum running time ",MaxRunTime,"
seconds...");
	repeat
		forall(ant in ColonySize) ConstructVehicleRoutes(ant);
		SelectBestSolution_of_the_Iteration;
	until (gettime-StartTime > MaxRunTime)
	writeln("+++Best so far value found: ", Best_so_far_Value);
	writeln("Printing best so far solution...");
	forall(t in 1..Tours_in_Best_so_far_Solution) writeln("Tour ",t,": ",Best_so_far_Solution(t));
	writeln("+done!");


	fopen("final_solution.txt",F_OUTPUT)
		writeln("Best:", Best_so_far_Value);
		writeln("Tours: ",Tours_in_Best_so_far_Solution);
		writeln("Solution:[");
		forall(t in 1..Tours_in_Best_so_far_Solution) writeln("(",t,") ", Best_so_far_Solution(t));
		writeln("]")
	fclose(F_OUTPUT)
end-procedure

Main

(! End ACO block !)
end-model
!Stat rosa pristina nomine
!Nomina nuda tenemus
```

Annexe 4 : simulateur de coût de revient d'un véhicule

SIMULATEUR DE PRIX DE REVIENT ***SAISIE DES DONNEES***

Les cellules en jaune sont à documenter. L'unité s'affiche automatiquement. **A saisir**

Commentaires facilitant la saisie

Les véhicules étudiés

Kilométrage annuel moyen d'un véhicule moteur ***(km)***	**110.000 km**
Nombre de véhicules attelés pour un véhicule moteur	**0,0**
Nombre de ***jours*** d'exploitation du véhicule moteur par an	**249,0 j**

1 - Les conditions de détention et de financement du véhicule moteur

Mode de financement du véhicule moteur ***(1 pour Emprunt, 2 pour Crédit Bail, 3 pour Location Financière)***			**2**
Emprunt		Crédit Bail ou Location Financière *	
Valeur du véhicule moteur ***(€)***	**100.000,00 €**	Valeur du véhicule moteur ***(€)***	
Part de l'emprunt dans le renouvellement ***(€)***	**90.000,00 €**	Durée du contrat ***(mois)*** *	**48 mois**
Durée du contrat ***(mois)***	**60 mois**	Périodicité des loyers * ***(1 mensuelle, 3 trimestrielle, 12 annuelle)***	**1**
Périodicité des remboursements ***(1 mensuelle, 3 trimestrielle, 12 annuelle)***	**1**	Montant du loyer ***(€)*** *	**2.000,00 €**
Taux d'intérêt ***(%)***	**4,50 %**	Valeur optionnelle d'achat ***(€)***	
Valeur de revente ***(€)***	**10.000,00 €**	Valeur de revente ***(€)***	

* *pour la location financière, saisir uniquement le loyer et sa périodicité pour le crédit bail saisir toutes les cellules*

Durée d'utilisation du véhicule moteur ***(ans)***	**4,0 an(s)**

2 - Les conditions de détention et de financement du véhicule tracté

Mode de financement du véhicule tracté ***(1 pour Emprunt, 2 pour Crédit Bail, 3 pour Location Financière)***			
Emprunt		Crédit Bail ou Location Financière *	
Valeur du véhicule tracté ***(€)***		Valeur du véhicule tracté ***(€)***	
Part de l'emprunt dans le renouvellement ***(€)***		Durée du contrat ***(mois)*** *	
Durée du contrat ***(mois)***		Périodicité des loyers * ***(1 mensuelle, 3 trimestrielle, 12 annuelle)***	
Périodicité des remboursements ***(1 mensuelle, 3 trimestrielle, 12 annuelle)***		Montant du loyer ***(€)*** *	
Taux d'intérêt ***(%)***		Valeur optionnelle d'achat ***(€)***	
Valeur de revente ***(€)***		Valeur de revente ***(€)***	

* *pour la location financière, saisir uniquement le loyer et sa périodicité pour le crédit bail saisir toutes les cellules*

Durée d'utilisation du véhicule tracté ***(ans)***	

3 - Assurances par véhicule (ou ensemble routier)

Montant annuel de l'assurance RC + VI ***(€)***	**2.500,00 €**
Montant annuel de l'assurance marchandises transportées ***(€)***	**500,00 €**

4 - Taxes par véhicule (ou ensemble routier)

Montant annuel des taxes (taxe à l'essieu et autres) ***(€)***	**274,40 €**

Les coûts directs d'usage de véhicule

1 - Le carburant

Consommation aux 100 Km ***(litre pour 100 km)***	**25,0 l**
Cout moyen du litre (hors toutes taxes récupérables) ***(€)***	**1,400 €**

2 - La maintenance

	Prix d'un pneu ***(€)***	Nombre de pneus	Durée de vie ***(km)***
Véhicule moteur	**180,00 €**	**4**	**40.000 km**
Véhicule tracté			

Contrat pneumatiques global ***(€/km)*** *	

* *Entrez les prix et le nombre de pneus, même en cas de contrat pneumatiques.*

Dépenses annuelles d'entretien-réparations par véhicule ***(€)***	**1.480,00 €**

3 - Péages

Montant annuel moyen des péages par véhicule ***(€)***	**0,00 €**

Coûts de personnel de conduite

Nombre de conducteurs affectés à 1 véhicule	**1,0**
Nombre de ***jours*** d'activité par an d'un conducteur affecté au véhicule	**217,0 j**
Temps de service mensuel ***(heures)***	**198,0 h**
Nombre de ***jours*** travaillés pour un mois de pleine activité	**22,0 j**
Salaire mensuel correspondant ***(€)***	**2.300,00 €**
Primes annuelles ***(€)***	**0,00 €**
Taux de charges patronales ***(%)*** après allègements	**40,0 %**
Frais de déplacement journaliers ***(€/jour)***	**0,00 €**
Nombre de mois payés	**13,0**

les données entrées sous ces 4 lignes doivent être cohérentes avec le kilométrage annuel moyen attendu

Estimation des coûts de gestion et de structure imputable au véhicule

Estimation du montant annuel par véhicule ***(€/an)***	**0,00 €**

SIMULATEUR DE PRIX DE REVIENT ***RECOMPOSITION DU PRIX DE REVIENT***

Recomposition du prix de revient du véhicule étudié en Euros

Coûts kilométriques directs		
Carburant	**0,350 €/km**	34,7%
Pneumatiques	**0,018 €/km**	1,8%
Entretien-réparations	**0,013 €/km**	1,3%
Péages	**0,000 €/km**	0,0%
Total rapporté au kilomètre parcouru	**0,381 €/km**	***37,8%***

Coûts de véhicule rapportés à 1 journée d'exploitation du véhicule		
Coût de détention véhicule moteur	**96,39 €/j**	21,6%
Coût de détention véhicule attelé	**0,00 €/j**	0,0%
Assurances	**12,05 €/j**	2,7%
Taxes	**1,10 €/j**	0,2%
Total par journée d'exploitation	**109,54 €/j**	***24,6%***

Coûts de personnel de conduite rapportés à 1 journée d'exploitation du véhicule		
Salaires et autres éléments de rémunération	**120,08 €/j**	26,9%
Charges sur salaires et autres éléments de rémunération	**48,03 €/j**	10,8%
Frais de déplacement	**0,00 €/j**	0,0%
Total par journée d'exploitation	**168,11 €/j**	***37,7%***

Coûts de structure rapportés à 1 journée d'exploitation du véhicule		
Coûts de structure et autres charges	**0,00 €/j**	***0,0%***

Prix de revient recomposé	**111.094 €**

Formulation Trinôme du prix de revient

			unités d'œuvre correspondantes
Terme kilométrique (1 km parcouru)	CK	**0,381 €/km**	*110.000,0 km*
Hors péages	*CK hp*	***0,381 €/km***	
Terme horaire (1 heure de temps de service)	CC	**21,43 €/h**	*7,8 h/j*
Terme journalier (1 jour d'exploitation du véhicule)	CJ	**109,54 €/j**	*249,0 j*

Application au prix de revient d'1 tournée		
	Nb unités	Coûts
Nombre total de km parcourus	**933 km**	**355,90 €**
Nombre d'heures de travail du conducteur livreur	**18,66 h**	**399,95 €**
Coefficient d'imputation du terme journalier	**4,00**	**438,14 €**
Prix de revient hors TVA		**1.193,99 €**