

6.2 - L'algorithme de décomposition des séries

Introduction:

Dans le concept TOPOROBOT, les visées composant un réseau sont organisées en séries; chaque série correspond généralement à une galerie et est raccordée à ses congénères par une ou deux extrémités. Elle peut aussi se refermer sur elle même.

D'autres séries peuvent partir ou arriver à un point quelconque d'une série donnée.

La méthode de calcul utilisée dans GHTopo est le procédé de TAILLARD modifié, décrit dans ce rapport. Cette méthode suppose que le réseau est organisé en branches et noeuds: une branche relie deux noeuds et il n'y a pas de noeud à l'intérieur d'une branche. De plus, tout point du réseau doit être raccordé aux autres.

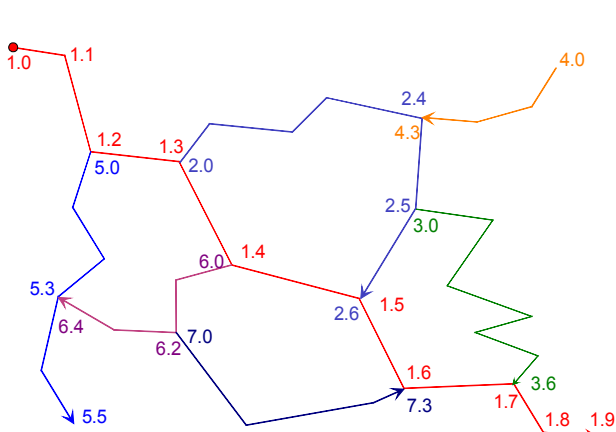


Schéma TOPOROBOT

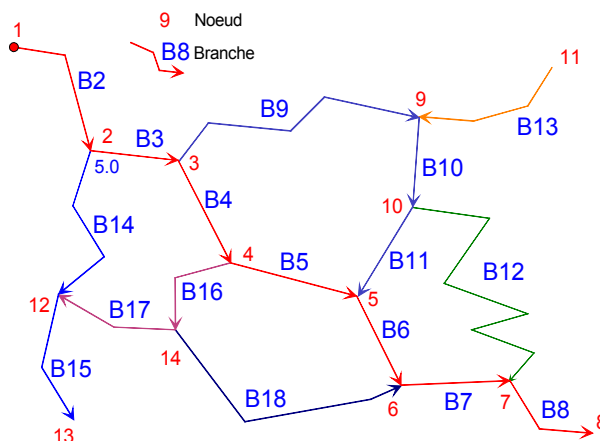


Schéma noeuds/branches

La problématique est donc de transformer un réseau TOPOROBOT en un graphe orienté avec noeuds et branches. Nous allons proposer ici un algorithme simple de décomposition,

Principe et idées de la décomposition:

La décomposition procède d'abord à un recensement des noeuds du réseau. Les entrées, les points fixes et les extrémités des séries sont forcément des noeuds; de plus ils constituent la totalité des noeuds de ce réseau. Par conséquent, il suffit de balayer la liste des entrées et celle des séries pour obtenir la table des noeuds (appelés Jonctions dans le concept TOPOROBOT), qui inclut les entrées et les points fixes.

Une fonction appelée **GetNoeud(No_Serie, No_Station)** permet d'obtenir le numéro de noeud correspondant à un couple série_station. Cette fonction scanne la table des jonctions et retourne le numéro de noeud si le couple série_station correspond à une jonction et -1 sinon.

L'étape suivante est le balayage des séries. Il procède de la façon suivante:

1. La table des branches est vidée et initialisée.

2. L'algorithme suivant est appliqué

Pour(1) chaque série:

Une branche est créée

Elle a pour noeud de départ le résultat de **GetNoeud(Série/Station)** de départ de la série

Pour(2) chaque point de la série:

Une visée est ajoutée à la branche courante

Si (GetNoeud(Série, Point) > -1 **et** si la fin de série n'est pas atteinte) **alors**:

Le noeud d'arrivée de cette branche est GetNoeud(Série, Point)

La branche est clôturée

Une nouvelle branche est créée

Le noeud de départ de cette nouvelle branche est GetNoeud(Série, Point)

Fin Si

Si la fin de série est atteinte **Alors**:

Le noeud d'arrivée de la branche courante est GetNoeud(Série, No_dernier_Point)

La branche est clôturée

Fin Si

Fin Pour(2)

Fin Pour(1)

Enfin les entrées sont ajoutées; chaque entrée correspond à une branche.

Le nœud de départ de cette branche est toujours 0 ou 1 suivant les implémentations; dans GHTopo, cela revient au même.

Le nœud d'arrivée est GetNoeud(No_Serie_Rattachement_Entree, No_Point_Rattachement_Entree)

Le jeu de données résultant est ensuite passé au code de calcul pour traitement.

Ceci se traduit par les extraits de code suivants:

Fonction GetNoeud:

```
// retourne le numéro de noeud pour le point passé par série/station:
function TToporobotStructure.GetNoeud(const Serie: integer; Station: integer):
integer;
var
  i      : integer;
  Jonc: TJonction;
  S      : string;
begin
  Result:=-1;
  for i:=0 to self.FNbJonctions-1 do begin
    Jonc:=self.GetJonction(i);
    S:=Format(FMT_NDSER_PT,[Serie, Station]);
    if S = Jonc.IDJonction then begin
      Result:=i;
      exit;
    end;
  end;
end;
```

Recensement des jonctions, entrées et culs-de-sac

```
procedure TToporobotStructure.RecenserJonctions;
var
  Entree : TEntrance;
  Serie  : TUneSerie;
  Station: TUneVisee;
  Ser, St: integer;
  ListeJnt: TStringList;
  J      : TJonction;
begin
  AfficherMessage('Recensement des jonctions');
  ListeJnt:=TStringList.Create;
  ListeJnt.Clear;
  ListeJnt.Sorted:=True;
  ListeJnt.Duplicates:=dupIgnore;
  // Les coordonnées d'accrochage des entrées de cavités sont forcément des jonctions
  for Ser:=0 to FNbEntrees-1 do begin
    Entree:=GetEntree(Ser);
    ListeJnt.Add(Format(FMT_NDSER_PT,[Entree.eRefSer,Entree.eRefSt]));
  end; //*)
  // Les série/point de départ/arrivée sont forcément des jonctions (noeuds)
  for Ser:=0 to FNbSeries - 1 do begin
    Serie:=GetSerie(Ser);
    ListeJnt.Add(Format(FMT_NDSER_PT,[Serie.SerieDep, Serie.PtDep]));
    ListeJnt.Add(Format(FMT_NDSER_PT,[Serie.SerieArr, Serie.PtArr]));
  end;
  FNbJonctions:=ListeJnt.Count;
  // en interne, l'ID de noeud est un littéral
  for St:=0 to FNbJonctions-1 do begin
    J.IDJonction:=ListeJnt.Strings[St];
```

```

    J.NoNoeud:=St;
    AddJunction(J);
end;
// on libère la table provisoire
ListeJunct.Free;
end;

```

Découpage des séries en branches et noeuds:

```

procedure TToporobotStructure.RecenserBranches;
const
    STR_FMT_STATIONS = '>> %d/%d |%.3d %.3d| %.2f %.2f %.2f | %.2f %.2f %.2f %.2f |
%s';
var
    Entr : TEntrance;
    Serie: TUneSerie;
    Visee: TUneVisee;
    Ser, Vis: integer;
    ll, al, pl: double;
    Br : integer;
    Nd : integer;
    Branche0,
    Branche1: TBranche;
begin
    AfficherMessage('Recensement Branches');
    // initialisation de la première branche
    with Branche0 do begin
        NoSerie:=1;
        NoBranche:=1;
        NomBranche:=Format('Branche %d',[1]);
        NoeudDepart:=0;
        NoeudArrivee:=1;
        DeltaX:=0.01;
        DeltaY:=0.01;
        DeltaZ:=0.01;
    end;
    AddBranche(Branche0);
    Br:=2;
    //*****
    // balayage des séries
    for Ser:=1 to FNbSeries-1 do begin
        Serie:=GetSerie(Ser);
        Nd:=GetNoeud(Serie.SerieDep, Serie.PtDep);
        // début de série = nouvelle branche
        //+++++
        // Numéro de série = ID de la série
        Branche0.NoSerie:=Serie.IndexSerie;
        //+++++
        Branche0.NoBranche:=Br;
        Branche0.NomBranche:=Format('Branche %d',[Br]);
        Branche0.NoeudDepart:=Nd;
        AddBranche(Branche0);
        for Vis:=1 to Serie.NbPoints-1 do begin
            Visee:=GetStation(Ser, Vis);
            Nd:=GetNoeud(Serie.IndexSerie, Vis);
            Visee.NoViseeSer:=Vis;
            //Visee.TypeGalerie := 0; // type de galerie (provisoire)
            AddBrStation(Br, Visee);
            Branche1:=GetBranche(Br);
            Branche0.PointsTopo:=Branche1.PointsTopo;
            Branche0.NbPoints:=Branche1.NbPoints;
            if (Nd>-1) AND (Vis < Serie.NbPoints-1) then begin
                Branche0.NoeudArrivee:=Nd;
                PutBranche(Br, Branche0);
                Br:=Br+1;
                Branche0.NoBranche:=Br;
            end;
        end;
    end;

```

```

    Branche0.NomBranche:=Format('Branche %d',[Br]);
    Branche0.NoeudDepart:=Nd;
    AddBranche(Branche0);
end;
// fin de série = fin de branche: cloturer la branche
if Vis = Serie.NbPoints-1 then begin
    Nd:=GetNoeud(Serie.SerieArr, Serie.PtArr);
    Branche0.NoeudArrivee:=Nd;
    PutBranche(Br, Branche0);
    Br:=Br+1;
end;
end; // for Vis:=1 to Serie.NbPoints-1 do begin
end; // for Ser:=1 to FNbSeries-1 do begin
//*****
// fin du balayage des séries

// ajout des barres fictives (entrées de cavités)
//*****
AfficherMessage('Ajout des barres fictives');
// La première entrée est déjà prise en compte
for Ser:=0 to FNbEntrees-1 do begin
    Entr:=GetEntree(Ser);
    //Branche0.NoSerie:=- (Ser+1);
    Branche0.NoSerie:=Entr.eRefSer;
    Branche0.NoeudDepart:=GetNoeud(1,0);
    Branche0.NoeudArrivee:=GetNoeud(Entr.eRefSer, Entr.eRefSt);
    Branche0.DeltaX :=Entr.eDeltaX;
    Branche0.DeltaY :=Entr.eDeltaY;
    Branche0.DeltaZ :=Entr.eDeltaZ;
    Visee.Code:=0; //1
    Visee.Expe:=0; //1

    l1:=0.0;
    a1:=0.00;
    p1:=0.00;
    // GetBearingInc transforme un vecteur en triplet Long, Az, P
    GetBearingInc(Entr.eDeltaX, Entr.eDeltaY, Entr.eDeltaZ,
        l1, a1, p1,
        400,400);
    Visee.NoVisee := 1;
    Visee.Longueur := l1;
    Visee.Azimut := a1;
    Visee.Pente := p1;
    Visee.LD :=0.00;
    Visee.LG :=Visee.LD;
    Visee.HZ :=Visee.LD;
    Visee.HN :=Visee.LD;
    Visee.Commentaires:='';
    Visee.IDTerrainStation:=Format(FMTSERST,[Entr.eRefSer, Entr.eRefSt]);

    Visee.TypeGalerie := tgENTRANCE;
    Visee.Commentaires:=SafeTruncateString(Entr.eNomEntree, 15);
    AddBranche(Branche0);
    AddBrStation(Br, Visee);

    Visee:=GetBrStation(Br, 1);
    AfficherMessage(Format('%d-%d - %d Type: %d - %.2f, %.2f,'+
        ' %.2f -%.2f, %.2f, %.2f',
        [Br, 1, Visee.NoVisee, Ord(Visee.TypeGalerie),
        Entr.eDeltaX, Entr.eDeltaY, Entr.eDeltaZ,
        Visee.Longueur, Visee.Azimut, Visee.Pente]));
    Br:=Br+1;
end; // for Ser:=0 to FNbEntrees-1 do begin
AfficherMessage('Recensement Branches OK');
end;

```