Catholic University of Louvain

Project 5

LINGI2365 - Constraint Programming

Auteurs :
Vanwelde Romain (3143-10-00)
Crochelet Martin (2236-10-00)

Groupe 7

Superviseurs :
Pr. Yves Deville
François Aubry

8 mai 2014

Table des matières

1 Explain the provided model (how are the constraints enforced, what do the global constraints used do, what do the decision variables represent, . . .)

First of all, the model creates three range variables :
— Customers from 1 to n.
— Depots from n+1 to n+K
— CustomersAndDepots from 1 to n+K
As we can see, we create one depot per vehicle (K vehicles) because we use the giant tour representation.

Decision variables

We can now take a look to decisions variables :

```
var<CP>{int} previous[CustomersAndDepots](cp,CustomersAndDepots);
```

Here, previous will contain for each customer/depot the id of the previous customer/depot.

```
var<CP>{int} routeOf[i in CustomersAndDepots](cp,1..K);
```

routeOf will contain for each customer/depot the id of the vehicle which will stop there.

```
var<CP>{int} service_start[i in CustomersAndDepots](cp,Horizon);
```

service_start will contain for each customer/depot the time when it will begin to be serviced.

```
var<CP>{int} departure[i in CustomersAndDepots](cp,Horizon);
```

departure will contain for each customer/depot the time when the vehicle will leave the actual customer/depot.

```
var<CP>{int} totDist = minCircuit(previous,distance);
```

totDist will contain the minimum distance linking all customer/depot.

Constraints

```
forall(i in Customers) cp.post(routeOf[i] == routeOf[previous[i]]);
```

This constraint forces all the customers to be served by the same vehicle that the previous customer/depot. Thus, if we visualize it on the giant tour representation, it means that if we take a depot, this will have a specific vehicle, and that all the following customers (until the following depot) will have the same vehicle which will serve them.

```
forall(i in Depots) cp.post(routeOf[i] == i - Customers.getUp());
```

This constraint will assign to all depots a different vehicle.

```
forall(i in CustomersAndDepots){
        cp.post(service_start[i] == max(tw_start[i],departure[previous[i]] +
            distance[previous[i],i]));
        cp.post(service_start[i] <= tw_end[i]);
        cp.post(service_start[i] >= tw_start[i]);
}
```

This constraint express the fact that all customers/depot will begin to be served whether at the beginning of the time window if the vehicle were on place sooner, or when the vehicle arrives (time when it leaves the previous customer plus the displacement time (which is equal to the distance)).
Then it expresses the fact that the service must begins before it ends (which is logical).
Last, it express the fact that for each customer/depot, the service will not be performed earlier than the beginning of his own starting window.

```
forall(i in Customers){
        cp.post(departure[i] ==  service_start[i] + service_duration[i]);
}
```

This constraint express the fact that the vehicle leave a customer the duration of the service after having beginning the task.

```
forall(i in Depots){
        cp.post(departure[i] == 0);
}
```

This constraint express the fact that the vehicle will immediately leave the depot at time t0.

Global constraint

```
cp.post(multiknapsack(routeOf, demand, all(k in 1..K) Q));
```

2 Explain how you adapt the model to minimize the number of routes

A first adaptation could be create by maintaining the previous model, and adding a decision variable.

```
var<CP>{int} nOfRoutes(cp,1..K);
```

This variable will keep track the number of different routes (This is what we will try to minimize).

If less vehicles than available are used, it will be represented on the giant tour representation as a depot directly following another depot. (The vehicle directly leaves the depot, and directly arrives there again).

```
cp.post(nOfRoutes == (sum(i in Depots) previous[i] <= Customers.getUp()) );
```

This constraint assign to OfRoutes the number of depot which follows another depot. Thus, it represents the number of used vehicles that we want to minimize.

Since we want to first minimize the number of route, and the total distance, we minimize or porblem with the following evaluation function.

```
minimize<cp> (nOfRoutes * 10000) + totDist
```

3 Explain each of your optimization procedures in detail (what, why and how)

4 Indicate and analyze your test results