TP7 - Les mobiles de Choco

Paul Chaignon - Ulysse Goarant

29 mars 2014

Listing 1 – Mobile.java

```
import static choco.Choco.*;
import choco.Choco;
import choco.cp.model.CPModel;
import choco.cp.solver.CPSolver;
import choco.kernel.model.constraints.Constraint;
import
   choco.kernel.model.variables.integer.IntegerExpressionVariable;
import choco.kernel.model.variables.integer.IntegerVariable;

public class Mobile {
        // longueurs
        private int l1, l2, l3, l4;
        // dernieres valeurs trouvees pour les masses
        private int m1, m2, m3;

        // variables des contraintes sur les longueurs
        private CPModel myModelL;
        private CPSolver mySolverL;
        private boolean coherent;// vrai si les longueurs
           sont coherentes

        // variables des contraintes sur les masses
        private CPModel myModelM;
        private CPSolver mySolverM;

        private IntegerVariable l1IV, l2IV, l3IV, l4IV,
           m1IV, m2IV, m3IV;

        private boolean equilibre;

        // constructeur : _lx : longeur de la branche x,
           m_max : masse maximum
        // disponible
        public Mobile(int _l1, int _l2, int _l3, int _l4,
           int m_max) {
                l1 = _l1;
```

```
        l2 = _l2;
        l3 = _l3;
        l4 = _l4;

        this.l1IV = Choco.makeIntVar("l1", 1, 20);
        this.l2IV = Choco.makeIntVar("l2", 1, 20);
        this.l3IV = Choco.makeIntVar("l3", 1, 20);
        this.l4IV = Choco.makeIntVar("l4", 1, 20);
        this.m1IV = Choco.makeIntVar("m1", 1, m_max);
        this.m2IV = Choco.makeIntVar("m2", 1, m_max);
        this.m3IV = Choco.makeIntVar("m3", 1, m_max);
        mySolverL = new CPSolver();
        myModelL = new CPModel();
        myModelL.addVariable(this.l1IV);
        myModelL.addVariable(this.l2IV);
        myModelL.addVariable(this.l3IV);
        myModelL.addVariable(this.l4IV);
        mySolverM = new CPSolver();
        myModelM = new CPModel();
        myModelM.addVariable(this.m1IV);
        myModelM.addVariable(this.m2IV);
        myModelM.addVariable(this.m3IV);
        coherent = false;
        equilibre = false;
        poseProblemeLongeur();
        poseProblemeMasse();
}

public boolean estEquilibre() {
        return equilibre;
}

// accesseurs
public int getL1() {
        return l1;
}

public int getL2() {
        return l2;
}

public int getL3() {
        return l3;
}

public int getL4() {
        return l4;
}
```

```
public int getM1() {
        return m1;
}

public int getM2() {
        return m2;
}

public int getM3() {
        return m3;
}

// pose le probleme des longeurs (sans le resoudre),
// Les longueurs sont coherentes si le mobile est
   libre
// (remarque : un peu artificiel car faisable en java
// sans contraintes !)
private void poseProblemeLongeur() {
        Constraint cs1 = eq(this.l1IV, l1);
        this.myModelL.addConstraint(cs1);
        Constraint cs2 = eq(this.l2IV, l2);
        this.myModelL.addConstraint(cs2);
        Constraint cs3 = eq(this.l3IV, l3);
        this.myModelL.addConstraint(cs3);
        Constraint cs4 = eq(this.l4IV, l4);
        this.myModelL.addConstraint(cs4);

        IntegerExpressionVariable sum =
           plus(this.l1IV, this.l2IV);
        Constraint constraint1 = gt(sum, l3IV);
        Constraint constraint2 = gt(sum, l4IV);
        this.myModelL.addConstraint(constraint1);
        this.myModelL.addConstraint(constraint2);
}

// verifie la coherence des longueurs
public boolean longueursCoherentes() {
        mySolverL.read(myModelL);
        coherent = mySolverL.solve();
        return coherent;
}

// pose le probleme des masses (sans le resoudre)
private void poseProblemeMasse() {
        IntegerExpressionVariable branch1 =
           mult(this.m1IV, this.l1);
        IntegerExpressionVariable subMobile =
```

```
            plus(this.m2IV, this.m3IV);
        IntegerExpressionVariable branch2 =
            mult(subMobile, this.l2);
        Constraint constraint = eq(branch1, branch2);
        this.myModelM.addConstraint(constraint);

        IntegerExpressionVariable branch3 =
            mult(this.m2IV, this.l3);
        IntegerExpressionVariable branch4 =
            mult(this.m3IV, this.l4);
        Constraint subConstraint = eq(branch3,
            branch4);
        this.myModelM.addConstraint(subConstraint);
}

// resoud le probleme des masses
// la resolution n'est lancee que si l'encombrement
   est coherent
public boolean equilibre() {
        if (!coherent) {
                return false;
        }
        mySolverM.read(myModelM);
        equilibre = mySolverM.solve();
        m1 = mySolverM.getVar(m1IV).getVal();
        m2 = mySolverM.getVar(m2IV).getVal();
        m3 = mySolverM.getVar(m3IV).getVal();
        return equilibre;
}

// cherche une autre solution pour les masses
// la recherche d'une autre solution ne doit etre
   lancee que si le mobile
// est equilibre
public boolean autreSolutionMasse() {
        if(!equilibre) {
                return false;
        }
        boolean solution = mySolverM.nextSolution();
        m1 = mySolverM.getVar(m1IV).getVal();
        m2 = mySolverM.getVar(m2IV).getVal();
        m3 = mySolverM.getVar(m3IV).getVal();
        return solution;
}

// gestion de l'affichage
public String toString() {
        String res = "l1 = " + l1 + "\n l2 = " + l2
```

```
                + "\n l3 = " + l3
                                + "\n l4 = " + l4;
                if (equilibre) {
                        res += "\n m1 = " + m1 + "\n m2 = "
                           + m2 + "\n m3 = " + m3;
                } else {
                        res += "\n masses pas encore
                           trouvees ou impossibles !";
                }
                return res;
        }

        // tests
        public static void main(String[] args) {
                Mobile m = new Mobile(1, 3, 2, 1, 20);
                // Mobile m = new Mobile(1, 1, 2, 3, 20);
                // Mobile m = new Mobile(1, 3, 1, 1, 20);
                // Mobile m = new Mobile(1, 3, 2, 1, 20);
                System.out.println(m);
                if (m.longueursCoherentes()) {
                        System.out.println("Encombrement
                           OK");
                        m.equilibre();
                        System.out.println(m);
                        while (m.autreSolutionMasse()) {
                                System.out.println("OU");
                                System.out.println(m);
                        }
                } else {
                        System.out.println("Encombrement pas
                           coherent !");
                }
        }
}
```