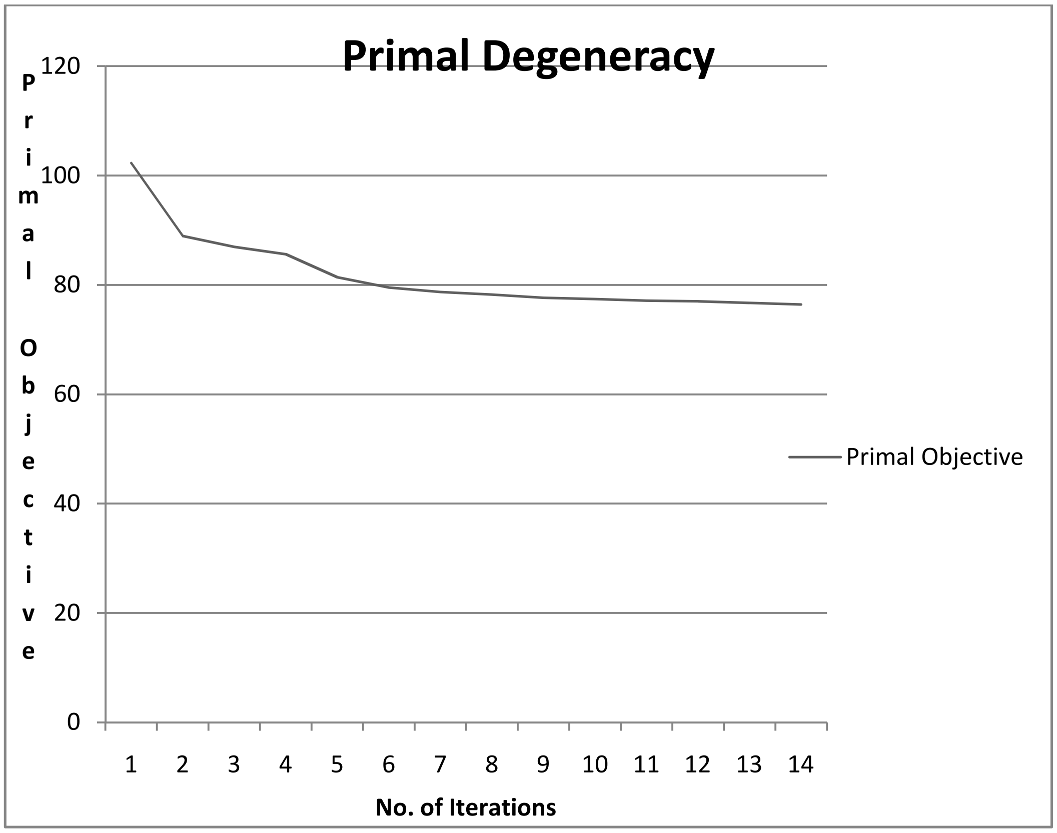
Column Generation : CPLEX CONCERT C#

COLUMN GENERATION FOR CSP. ACTUAL CODE IS AVAILABLE IN CPLEX USER MANUAL. IT IS FURTHER MODIFIED AND SIMPLIED FOR EASY UNDERSTANDING.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ILOG.Concert;
using ILOG.CPLEX;

namespace Cutstock_new_approach
{
    class Parameter
    {
        public static double subproblemobjlimit = 1.0e-6;
        public static void configMasterProb(Cplex cplex)
        {

            try
            {
                // branch and bound

                cplex.SetParam(Cplex.Param.MIP.Strategy.NodeSelect, 1);
                cplex.SetParam(Cplex.Param.MIP.Strategy.Branch, 1);
                //masterproblem.cplex.setParam(IloCplex.Param.Preprocessing.Presolve,
true);
                // display options
                cplex.SetParam(Cplex.Param.MIP.Display, 2);
                cplex.SetParam(Cplex.Param.Tune.Display, 1);
                cplex.SetParam(Cplex.Param.Simplex.Display, 0);
            }
            catch (ILOG.Concert.Exception e) { System.Console.WriteLine("Error for
Masterproblem: " + e); }

        }

        public static void configSubProblem(Cplex cplex)
        {
           try
            {
                // branch and bound
                cplex.SetParam(Cplex.Param.MIP.Strategy.NodeSelect, 1);
                cplex.SetParam(Cplex.Param.MIP.Strategy.Branch, 1);
                cplex.SetParam(Cplex.Param.MIP.Tolerances.MIPGap, 0.1);
                // display options
                cplex.SetParam(Cplex.Param.MIP.Display, 0);
            }
            catch (ILOG.Concert.Exception e) { System.Console.WriteLine("Error for
Subproblem: " + e); }
        }




        }
    }
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ILOG.Concert;
using ILOG.CPLEX;
using System.IO;

namespace Cutstock_new_approach
{

    class Program
    {
        internal static void printfinalresult(Cplex masterprob,
System.Collections.ArrayList cut)
        {
             // Function to display final rsult//
            System.Console.WriteLine();
            System.Console.WriteLine("Best Objective Value:" +
masterprob.GetBestObjValue());
            for (int i = 0; i < cut.Count; i++)
            {

            System.Console.WriteLine("Cut"+i+"="+masterprob.GetValue((INumVar)cut[i]));
            }
        }

        internal static void printmasterproblem(Cplex masterprob,
System.Collections.ArrayList cut, IRange[] length, int iterations, StreamWriter FILE)
        {
            //Function to display final result//
            System.Console.WriteLine();
            System.Console.WriteLine("Masterproblem Objective" + masterprob.ObjValue);
            System.Console.WriteLine("Iteration No:" + iterations);
            FILE = new StreamWriter("C:/Iterations.txt", true);

            FILE.WriteLine(iterations + " " + masterprob.ObjValue);
            FILE.WriteLine("\n");
            FILE.Close();
            System.Console.WriteLine();
            for (int i = 0; i < cut.Count; i++)
            {

System.Console.WriteLine("Cut"+i+"="+masterprob.GetValue((INumVar)cut[i]));

            }
            System.Console.WriteLine();
            for (int j = 0; j < length.Length; j++)
            {
                System.Console.WriteLine("Dual"+j+"="+masterprob.GetDual(length[j]));

            }
        }
        static void Main(string[] args)
        {

            double rollwidth = 115;
            double[] size = { 20, 40, 60, 50, 10,30,45,25,35,35};
            double[] amount = { 51, 35, 23, 10, 20,22,34,35,25,10};
            StreamWriter file = null;
            try
            {
                // Define Master Problem//
                Cplex masterprob = new Cplex();
                IObjective masterprobobj = masterprob.AddMinimize();
                IRange[] length = new IRange[amount.Length];
                for (int i = 0; i < length.Length; i++)
                {
                    length[i] = masterprob.AddRange(amount[i], System.Double.MaxValue);
                }
                System.Collections.ArrayList cut = new System.Collections.ArrayList();
                int width = size.Length;
                for (int j = 0; j < width; j++)
                {
                    cut.Add(masterprob.NumVar(masterprob.Column(masterprobobj,
1).And(masterprob.Column(length[j], (int)(rollwidth / size[j]))), 0.0,
System.Double.MaxValue));

                }

                // Parameter Setting of CPLEX//
                //masterprob.SetParam(Cplex.Param.RootAlgorithm, Cplex.Algorithm.Primal);
                Parameter.configMasterProb(masterprob);
                // Define subproblem//
                Cplex subproblem = new Cplex();
                IObjective subproblemobj = subproblem.AddMinimize();
                INumVar[] cutwidth = subproblem.NumVarArray(width, 0.0,
System.Double.MaxValue, NumVarType.Int);
                //Adding constraint to subproblem//

                subproblem.AddRange(-System.Double.MaxValue, subproblem.ScalProd(size,
cutwidth), rollwidth);
                Parameter.configSubProblem(subproblem);
                // Start Column Generation//
                double[] subprob = new double[width];
                int count = 0;
                string filename;
                for (; ; )
                {

                    // Solve Master Problem and get duals//
                    masterprob.Solve();
                    count = count + 1;
                    printmasterproblem(masterprob, cut, length,count,file);
                    //Name each masterproblem with index//

                    filename="C:/CplexColumnGeneration"+count+".lp";

                    masterprob.ExportModel(filename);
                    double[] subduals = masterprob.GetDuals(length);

                    // Prepare the objective function of the subproblem//
                    subproblemobj.Expr = subproblem.Diff(1, subproblem.ScalProd(cutwidth,
subduals));
                    //Solve subproblem//
                    subproblem.Solve();
                    if (subproblem.ObjValue > -Parameter.subproblemobjlimit)
                        break;
                    // Transfer the value to an array//
                    subprob = subproblem.GetValues(cutwidth);
                    // Add column to masterproblem//
                    Column column = masterprob.Column(masterprobobj, 1);//Add new
objective of masterproblem
                    for (int k = 0; k < subprob.Length; k++)
                    {
                        column = column.And(masterprob.Column(length[k], subprob[k]));
                        //Adding a new variable to arraylist of master problem//
                        cut.Add(masterprob.NumVar(column, 0.0, System.Double.MaxValue));
                    }

                }

                // Convert Array to numvar prior to addition to master problem obj//
                for (int l = 0; l < cut.Count; l++)
                {
                    masterprob.Add(masterprob.Conversion((INumVar)cut[l], 
NumVarType.Int));


                }

                // Solve Master Problem finally//
                masterprob.Solve();
                System.Console.WriteLine("Status:" + masterprob.GetStatus());
                masterprob.ExportModel("C:/CplexColumnGeneration.lp");
                //Display final result//
                printfinalresult(masterprob, cut);
                //Close masterprob solver first//
                masterprob.End();
                subproblem.End();

            }
            catch (ILOG.Concert.Exception EXP)
            {
                System.Console.WriteLine("Error in Concert:" + EXP.Message);
            }
            System.Console.ReadLine();

        }
    }
}
```