Código del programa y la metodología empleada para la selección aleatoria de los elementos requeridos por el algoritmo de Schöning.

```
def schoning_algo(formula,lines):

   (k, n) = check_getn(formula)
   count = 0
   if not k:
      return (False,n)
   values = assign(formula)
   newFile.write("Original Instance: "+ str(values) +"\n\n")
   for index in xrange(3 * n):
      count += 1
      satisfied, failed = check_solution(formula, values)
      newFile.write("**********************************************
iteration no."+str(count)+"\n")
      if satisfied:

newFile.write("**********************************************"+"\n")

newFile.write("**********************************************"+"\n")
         newFile.write("SATISFACTORY PROPOSAL: "+instance+"\n")

newFile.write("**********************************************"+"\n")

newFile.write("**********************************************"+"\n")
         return (values, count)
      else:
         selection = random.choice(failed)
         false_clause = formula.split("a")[selection]
         newFile.write("Satisfactory clauses: "+ str(91-len(false_clause)) +"\n")
         newFile.write("Unsatisfactory clauses: "+ str(len(false_clause)) +"\n")

         var_to_change = random.choice(num_variables(false_clause))
```

```
            values[var_to_change] = False if values[var_to_change]  else True
            newFile.write("-Clausule to change: [" + str(lines[selection]) + "]\n")
            newFile.write("-Variable to change: " +str(var_to_change) + " --> " +
str(values[var_to_change]) + "\n" )
            newFile.write("New Instance: "+ str(values) +"\n\n")

    newFile.write("**********************************************"+"\n")
    newFile.write("**********************************************"+"\n")
    newFile.write("NO SOLUTION FOR THIS INSTANCE\n")
    newFile.write("**********************************************"+"\n")
    newFile.write("**********************************************"+"\n")
    return
```

El método “schoning_algo” recibe como parámetros el desglose de variables con las cláusulas de formato CNF. Este método aplica el algoritmo de Schoning directamente, iterando 3n veces la proposición inicial del método “assign” y comprobando su satisfacción con cada una de las cláusulas del problema K-SAT.

Si encuentra la proposición asignada como válida (ósea que se satisfaga todas las cláusulas) regresa la combinación obtenida como indicador de que se ha encontrado la instancia que resuelve el problema.

De lo contrario, selecciona aleatoriamente una de las variables de las cláusulas NO satisfechas y altera su valor binario comprobando la satisfacción de las cláusulas con la nueva iteración. Si al terminar las 3n iteraciones no se ha encontrado una proposición que satisfaga correctamente las cláusulas se considera el problema como imposible de resolver.

```
def assign(formula):
    variables = list(set(num_variables(formula)))
    values = {}
    for var in variables:
        values[var] = random.choice([False,True])
    return values
```

Este método se utiliza para crear la proposición inicial basados en la cantidad de cláusulas y variables, el cual he llamado “formula”. Usando la librería de random genera una combinación aleatoria entre valores de True y False que asigna a cada una de las 20 variables como primera instancia.