Solving the All Interval Problem

Jean-Francois Puget, Jean-Charles Régin

puget@ilog.fr, regin@ilog.fr

ILOG S.A.

9 rue de Verdun

94253 Gentilly Cedex

Abstract

In this paper we present an ILOG Solver [1] program that solves the all interval problem quite efficiently. Results presented in [2] have shown that finding all solution for medium size problems was quite easy with modern CP tools, but recent results [5] claimed that such tools weren't efficient for finding more than one solution for large instances. We first show a simple program that searches for all solutions, similar to what is presented in [2]. Refinements of this program using arc-consistent constraints are then presented. Finally, we present results on searching for several solutions on very large instances of the problem.

Specification

This problem is described as prob007 in the CSPLIB, at the URL:
http://www-users.cs.york.ac.uk/~tw/csplib/prob/prob007/spec.html
It can be expressed as follow. Find a permutation $(x_1, \ldots, x_n)$ of $\{0,1,\ldots,n-1\}$ such that the list $(\mathrm{abs}(x_2-x_1), \mathrm{abs}(x_3-x_2), \ldots, \mathrm{abs}(x_n-x_{n-1}))$ is a permutation of $\{1,2,\ldots,n-1\}$.

Finding all solutions

The problem is represented with an array of variables x and a second array y for representing the differences. Each array is subject to an all different constraint using bound consistency [3]. Searching for solutions uses a standard first fail strategy (smallest domain first).

```
void prob007(IlcManager m, int n) {
  IlcIntVarArray x(m, n, 0, n-1);
  IlcIntVarArray y(m, n-1, 1, n-1);
  for (int i=0; i<n-1; i++)
    m.add(y[i]==IlcAbs(x[i+1] - x[i]));
  m.add(IlcAllDiff(x, IlcWhenRange));
  m.add(IlcAllDiff(y, IlcWhenRange));
  m.add(IlcGenerate(x, IlcChooseMinSizeInt));
}
```

Results obtained with this program are summarized in the following table. The second column indicates the number of dead ends during search. Running times are given in seconds on a 800MHz PentiumIII laptop running Windows 2000.

n	backtracks	time	#solutions
8	496	0.03	40
9	1826	0.12	120
10	7153	0.51	296
11	30743	2.30	648
12	138990	11.3	1328
13	676359	57.4	3200
14	3457219	316.3	9912

Using arc consistency

It is possible to improve the previous results by using arc-consistency pruning at each node of the search tree. This can be obtained by using arc-consistent implementations of the constraints of the problem. ILOG Solver provides arc-consistent all different constraints [4]. The constraint linking the variables x and y is represented by the set of triples `(i,j,abs(j-i))` and `(j,i,abs(j-i))` satisfying it.

```
void prob007AC(IlcManager m, int n){
  IlcIntVarArray x(m, n, 0, n-1);
  IlcIntVarArray y(m, n-1, 1, n-1);
  IlcIntTupleSet set(m,3);
  int i,j;
  for(i=0;i<n-1;i++){
    for(j=i+1;j<n;j++){
      set.add(IlcIntArray(m,3, i, j, IlcAbs(j-i)));
      set.add(IlcIntArray(m,3, j, i, IlcAbs(j-i)));
    }
  }
  set.close();
  for (i=0; i<n-1; i++){
    IlcIntVarArray vars(m, 3, x[i], x[i+1], y[i]);
    m.add(IlcTableConstraint(vars, set, IlcTrue));
  }
  m.add(IlcAllDiff(x, IlcWhenDomain));
  m.add(IlcAllDiff(y, IlcWhenDomain));
  m.add(IlcGenerate(x, IlcChooseMinSizeInt));
}
```

The results are now

n	backtracks	time
8	270	0.05
9	828	0.18
10	2777	0.66
11	10071	2.50
12	38778	10.3
13	156251	43.3
14	674346	199.6

The number of backtracks has been reduced drastically, and running time growth is slower. For larger instances arc-consistency is better than bound consistency.

Solving large instances

For large instances, the cost of arc-consistency is too high. Using the bound-consistency model presented above, one can solve extremely large instance quickly, using a search ordering similar to the one presented in [1]. The idea is to assign first the largest possible differences. The code of the program is then:

```
void prob007large(IlcManager m, int n){
  IlcIntVarArray x(m, n, 0, n-1);
  IlcIntVarArray y(m, n-1, 1, n-1);
  int i;
  for (i=0; i<n-1; i++){
    m.add(y[i]==IlcAbs(x[i+1] - x[i]));
  }
  m.add(IlcAllDiff(x, IlcWhenRange));
  m.add(IlcAllDiff(y, IlcWhenRange));
  m.add(IlcGenerate(y,IlcChooseMaxMaxInt, IlcIntSelectMax(m)));
  m.add(IlcGenerate(x));
}
```

This program is quite fast at finding one or two solutions, without any backtracking, as shown in the next table. Note that all problems of size smaller than 100 solve in less than .01 second. The last two columns give the running time needed to get the first and the second solutions respectively.

n	backtracks	time(1st)	time(2nd)
100	0	0.01	0.01
200	0	0.04	0.05
500	0	0.27	0.3
1000	0	2.12	2.56
2000	0	11.9	14.9

It was claimed in [5] that ILOG Solver was quite slow at finding a second solution, taking more than one hour to solve the problem of size 18. The above result shows that this is not the case. The two solutions found are the following ones:

```
(0, n-1, 1, n-2, 2, …)
(n-1, 0, n-2, 1, n-3, …)
```

Those two solutions are symmetrical. Indeed, one can be obtained from the other by substituting $n-1-x_i$ for x_i. If we look for different solutions, following the arguments given in [5], we can get rid of the symmetrical solution by adding an extra constraint $x_0<x_1$. The results using this additional constraint are summarized in the table below.

n	backtracks	time(1st)	time(2nd)
100	3	0.01	0.02
200	0	0.04	0.09
500	0	0.03	0.87
1000	3	2.19	6.6
2000	0	12.1	32.8

The running times are greater, but still quite acceptable. In particular for problems of size less than 100, the second solution is found in .02 seconds or less.

Summary

We have shown how to efficiently compute all solutions to the all interval problem, using arc consistent constraints. We have also presented new results on the search for several solutions on very large instances of the same problem.

References

[1] J-F. Puget, M. Leconte. Beyond the Glass Box: Constraints as Objects
in *proceedings of ILPS95*, Portland, Oregon, 1995
[2] H. Simonis, N. Beldiceanu. A note on CPLIB probl007,
http://www-users.cs.york.ac.uk/~tw/csplib/prob/prob007/helmut.pdf
[3] M. Leconte. A Bounds-Based Reduction Scheme for Difference Constraints.
In *proc. of Constraints96*, Key West, Florida, 19 May 1996
[4] J-C. Régin. A filtering algorithm for constraints of difference in CSPs.
In *proceedings of AAAI-94*, pages 362--367, Seattle, Washington, 1994
[5] C. Solnon, Solving permutation problems by ant colony optimization.
In *proceedings ECAI2000*, Berlin, 2000.