

Assignment 1 – Constraint Programming with Gecode

Stefano Benedettini Andrea Roli

Artificial Intelligence LM 2011

Contents

1	Gecode Hints and Guidelines	2
2	Constraint Satisfaction Problems	2
2.1	Combinatorial Mathematics and Number Theory	3
2.1.1	Map-coloring	3
2.1.2	Latin Square	3
2.1.3	N -queens	3
2.1.4	All-interval Series (adapted from CSPLib)	3
2.1.5	Magic Square (adapted from CSPLib)	4
2.1.6	Magic Sequence (adapted from CSPLib)	4
2.1.7	Partial Latin Square	4
2.1.8	n -fractions (adapted from CSPLib)	4
2.2	Working with Sets	5
2.2.1	Steiner Triplets (adapted from CSPLib)	5
2.2.2	Schur's Lemma (adapted from CSPLib)	5
2.2.3	Number Partitioning (from CSPLib)	5
2.3	Logic Puzzles	6
2.3.1	Cryptoarithmic	6
2.3.2	Sudoku	6
2.3.3	Futoshiki	6
2.3.4	Minesweeper	7
2.3.5	Nonogram (Crucipixel in Italian Magazines)	7
3	Constraint Optimization Problems	7
3.1	Low Autocorrelation Binary Sequence (adapted from CSPLib)	7
3.2	Fixed Length Error Correcting Codes (adapted from CSPLib)	8
3.3	Maximum Clique problem	8
3.4	Chromatic Number	8
3.5	Golomb Rulers (adapted from CSPLib)	8
3.6	Prime Queens (from CSPLib)	9

3.7	Rack Configuration Problem (from CSPLib)	9
3.8	Domination Number	9

Topics

- Constraint programming with **Gecode**
- Constraint satisfaction problems (CSP)
- Constraint optimization problems (COP)
- Search methods for CSPs and COPs

1 Gecode Hints and Guidelines

- You should organize your solver code for an exercise on a directory or a single file. Don't put solutions of your exercises on the same file because it gets messy pretty soon and compile times are longer.
- Remember to include the right headers.
- Remember to link the correct libraries. A general rule is to link `gecode<some-header>` library whenever you use `<some-header>`.
- Some exercise also require some C++ programming effort. For example, you might need to use containers or files. Remember to look their interfaces up on <http://www.cplusplus.com/reference/> or in any other documentation you might have.
- Remember to check companion material provided with this file.

2 Constraint Satisfaction Problems

In this section you will be asked to model constraint satisfaction problems. The focus will be mainly on the modeling part. That means that the majority of the problems faced can be easily solved by a naïve labeling strategy, if not by propagation alone (for example see 2.3.2).

You are required to use **Gecode** for your models and are strongly encouraged to keep on-line documentation at hand in order to find efficient, ready-made constraint definitions and utilities that will save you a lot of time and effort (for example, the **Matrix** class).

And now, on with exercises!

2.1 Combinatorial Mathematics and Number Theory

2.1.1 Map-coloring

1. Model and solve the Australia map-coloring instance on Russel and Norvig's book. Hard-code constraints in your model.
2. Come up with a more general model for solving a map-coloring problem. Users should be able to specify neighborhood relations in a text file. Encode and solve again the Australia map-coloring instance. A fairly simple format might be:

```
France Germany
France Italy
France Spain
Germany Italy
```

that is, a graph given as a list of pairs: each country is a label for each vertex and each pair represents an (undirected) edge.

2.1.2 Latin Square

A Latin Square is a $n \times n$ table filled with n different symbols such that no symbol occurs twice in each row and column. Use integers instead of symbols without restraint.

1. Write a general model for a Latin Square of size n , where n is an input.
2. Do you think that solutions to a Latin Square have interesting algebraic properties? What if we use a solution as “multiplication table” for a binary operator?

2.1.3 N -queens

The famous n -queens problem.

1. Write a model for every n .
2. Think about symmetries in this problem and add some constraints to remove some of them.

2.1.4 All-interval Series (adapted from CSPLib)

An interesting problem that has application in music, is the All-interval Series. Mathematically you are to find a vector $s = (s_1, \dots, s_n)$ where $s_i \in \mathbb{Z}_n$ such that:

- s is a permutation of $\mathbb{Z}_n = \{0, 1, \dots, n-1\}$;
- the interval vector $v = (|s_2 - s_1|, |s_3 - s_2|, \dots, |s_n - s_{n-1}|)$ is a permutation of $\mathbb{Z}_n \setminus \{0\} = \{1, 2, \dots, n-1\}$

Write a CSP formulation for this problem.

2.1.5 Magic Square (adapted from CSPLib)

An order n magic square is a $n \times n$ matrix containing the integers from 1 to n^2 , with each row, column, main and secondary diagonals equal the same sum. This sum M is called *magic constant* and equal to:

$$M = \frac{n(n^2 + 1)}{2}$$

It is strongly advised to use this information in your model.

Write a CSP formulation for this problem.

2.1.6 Magic Sequence (adapted from CSPLib)

A magic sequence of length n is a sequence of integers $x_0, x_1, \dots, x_{n-1}$ where $x \in [0, n-1]$, such that $\forall i \in [0, n-1]$, the number i occurs exactly x_i times in the sequence. For instance, $\langle 6, 2, 1, 0, 0, 0, 1, 0, 0, 0 \rangle$ is a magic sequence since 0 occurs 6 times in it, 1 occurs twice, etc. . .

1. Write a CSP model for this problem parametrized in n .
2. First try to solve it through labeling. Can you find a better search strategy, knowing how a solution looks like?
3. Sometimes adding redundant constraints helps propagation. Magic sequences have this property:

$$\sum_{i=0}^{n-1} i x_i = n$$

Exploit it to write a faster solver.

2.1.7 Partial Latin Square

Partial Latin Square problem or Quasigroup Completion problem is a *much more* challenging problem than plain Latin Square which shows, as you know, also a phase transition.

Try and solve some of the instances. Mind that they have all been generated on the phase transition so they will be hard! (Maybe too hard!)

2.1.8 n -fractions (adapted from CSPLib)

The original fractions puzzle is specified as follows. Find 9 *distinct non-zero* digits that satisfy:

$$\frac{A}{BC} + \frac{D}{EF} + \frac{G}{HI} = 1$$

where BC is shorthand for $10B + C$, EF for $10E + F$ and HI for $10H + I$ (like in cryptarithmic puzzles).

A simple generalisation (n -fractions) is as follows. Find $3n$ non-zero digits satisfying:

$$\sum_{i=1}^n \frac{x_i}{10y_i + z_i} = 1$$

and the number of occurrences of each digit in $1, \dots, 9$ is between 1 and $\lceil \frac{n}{3} \rceil$.

2.2 Working with Sets

These problems also involve set variables.

2.2.1 Steiner Triplets (adapted from CSPLib)

The ternary Steiner problem of order n consists of finding a set of $\frac{n \cdot (n-1)}{6}$ triples of distinct integer elements in $\{1, \dots, n\}$ such that any two triples have at most one common element.

2.2.2 Schur's Lemma (adapted from CSPLib)

The problem requires to put n balls labelled $\{1, \dots, n\}$ into k boxes so that for any triple of balls (x, y, z) with $x + y = z$, not all are in the same box.

1. Write a formulation using integer variables and solve for $k = 3$. (Hint: reduce it to an assignment problem, i.e., use binary variables.)
2. Write a formulation using set variables.

2.2.3 Number Partitioning (from CSPLib)

This problem consists in finding a partition of numbers $1, \dots, n$ into two sets A and B such that:

- A and B have the same cardinality
- $\sum_{x \in A} x = \sum_{x \in B} x$
- $\sum_{x \in A} x^2 = \sum_{x \in B} x^2$

Here is an example for $n = 8$: $A = \{1, 4, 6, 7\}$, $B = \{2, 3, 5, 8\}$.

There is no solution for $n < 8$. Also, from $n \geq 8$, there is no solution if n is not a multiple of 4.

2.3 Logic Puzzles

2.3.1 Cryptarithmic

Cryptarithmic puzzles consist of mathematical equations among unknown numbers, whose digits are represented by letters. One prominent example is `SEND + MORE = MONEY`, but there are lots of them on the Internet.

1. Write models for some of the following puzzles:
 - (a) `TWO + TWO = FOUR`
 - (b) `LOTS + SOLD + TO + OLD = FOOLS`
 - (c) `CROSS + ROADS = DANGER`
 - (d) `LINDON × B = JOHNSON`
2. Write a generic cryptarithmic solver that reads a puzzle instance from a file. Suppose that an instance consists of only 3 words and that the only operation used is addition. For example, a file containing:

`SEND MORE MONEY`

encodes the usual `SEND + MORE = MONEY` instance.

2.3.2 Sudoku

For those of you who have been away from Earth these last few years, Sudoku is a number puzzle where a 9×9 grid, further subdivided into $9 \ 3 \times 3$ square regions, is to be filled with natural numbers from 1 to 9 such that in no row, column or square region a number is repeated. Here is the Wikipedia page on Sudoku.

- Write a general CSP formulation for this problem. Suppose that your solver is fed with a partially filled instance, like those you find on magazines.
- Write a solver that can interface with I/O and read an instance from a file. File format is up to you.

2.3.3 Futoshiki

Futoshiki is a Sudoku-like number placement puzzle. A $n \times n$ grid has to be filled with integers such that each row and column contain all the numbers in $1, 2, \dots, n$. In addition, some cells are constrained to contain a number that is strictly greater than a neighboring cell content.

Write a general CSP formulation for this problem. The difficulty here is in instance format representation. Basically, you have to provide, beside grid-size n , all the hints, that is the initially placed numbers (you can freely recycle the code you wrote for Sudoku), and the constraints between cells (a pair of grid cell coordinates will do).

2.3.4 Minesweeper

Minesweeper is the pencil-and-paper version of the famous video game, with the very same rules. The objective is to find all the mines in a rectangular grid knowing:

- the number of mines;
- that for every cell containing a number, the 8 neighboring cells contain exactly that many mines;
- that a numbered cell never contains a mine.

Write a general formulation for this problem that reads an instance from a file.

2.3.5 Nonogram (Crucipixel in Italian Magazines)

Program a Nonogram solver that takes a specification file. Besides grid size (rectangular in general), a Nonogram instance is defined by its row and column hints.

Hint: extensional constraints might come in handy.

3 Constraint Optimization Problems

3.1 Low Autocorrelation Binary Sequence (adapted from CSPLib)

These problems have many practical applications in communications and electrical engineering. The objective is to construct a binary sequence S (S is a zero based indexed array) of length $n \in \mathbb{N}_0$ that minimizes the autocorrelations between bits. Each bit in the sequence takes the value $+1$ or -1 . There are two possible variants:

Non-periodic boundary conditions: the k -th autocorrelation $C_k = \sum_{i=0}^{n-k-1} S_i S_{i+k}$

Periodic boundary conditions: the k -th autocorrelation, $C_k = \sum_{i=0}^{n-1} S_i S_{(i+k) \bmod n}$

The aim is to minimize the sum of the squares of these autocorrelations. That is, to minimize $E = \sum_{k=1}^{n-1} C_k^2$.

Choose a boundary condition and model this problem as a COP with n as an input parameter.

You may find optimal sequences for $n \leq 60$ and for $n \geq 61$.

3.2 Fixed Length Error Correcting Codes (adapted from CSPLib)

A fixed length error correcting code $\mathcal{C}$ of length n over an alphabet $\mathbb{F}$ is a set of strings from $\mathbb{F}^n$. Given two strings from $\mathbb{F}^n$ we can define the distance between them. The most commonly used distance is the Hamming distance, defined as the number of positions where the strings differ. Using this we define the minimum distance of $\mathcal{C}$ as the minimum of the distances between distinct pairs of strings from $\mathcal{C}$.

Note that the CSP version of this problem (minimum Hamming distance is fixed) generates all the possible solutions, that is, all possible Hamming codes whose minimum distance is at least a certain number.

Implement a CSP and a COP model of this problem and solve it.

3.3 Maximum Clique problem

In graph theory, a k -clique $\mathcal{C}$ of an undirected graph G is a subset of the vertices of G which has cardinality k and such that for every two vertices there exists an edge connecting the two in G . $\mathcal{C}$ therefore induces a complete graph in G (it has $\binom{k}{2}$ edges).

1. Model this problem as a COP.
2. Successfully solve it on some graphs.

3.4 Chromatic Number

The chromatic number $\chi(G)$ of a graph G is the minimum number of color needed to color each vertex in G such that no two adjacent vertices have the same color.

1. Model and solve this problem as a COP.
2. What is the relationship between this problem and the Maximum Clique problem?

3.5 Golomb Rulers (adapted from CSPLib)

A Golomb ruler may be defined as a set of m integers $0 = a_1 < a_2 < \dots < a_m$ such that the $\frac{m \cdot (m-1)}{2}$ differences $a_j - a_i, 1 \leq i < j \leq m$ are distinct. Such a ruler is said to contain m marks and is of length a_m . The objective is to find optimal (minimum length) or near optimal rulers.

1. Model this problem as a COP and solve it.
2. This problem is particularly difficult for $m > 8$. Try to remove the following symmetry and see if it helps:

$$a_2 - a_1 < a_m - a_{m-1}$$

that is to say that the first difference is less than the last.

3.6 Prime Queens (from CSPLib)

This problem, posed first by G.L. Honaker, is to put a queen and the n^2 integer numbers $1, \dots, n^2$, on a $n \times n$ chessboard so that:

- no two numbers are on the same cell;
- any number $i + 1$ is reachable by a knight move from the cell containing i ;
- the number of “free” primes (i.e., primes not attacked by the queen) is minimal.

Remember that 1 *is not* prime, and that the queen does not attack its own cell.

3.7 Rack Configuration Problem (from CSPLib)

The rack configuration problem consists of plugging a set of electronic cards into racks with electronic connectors. Each card plugged into a rack uses a connector. In order to plug a card into a rack, the rack must be of a rack model.

Each card is characterised by the power it requires. Each rack model is characterised by the maximal power it can supply, its number of connectors, and its price. The problem is to decide how many of the available racks are actually needed, and which rack is of which rack model such that:

- every card is plugged into one rack;
- the total power demand and the number of connectors required by the cards does not exceed that available for a rack;
- the total price is minimised.

Hint: for this problem you might consider built-in bin packing constraint useful.

3.8 Domination Number

The domination number of a $n \times n$ chess board is the minimum number of queens to place in order to attack or occupy every square.

1. Model this problem for any given n as a COP.
2. Demonstrate that the domination number for a 8×8 chessboard is 5.
3. Can you find domination number for other chess pieces? Try refactor your solver in order to be able to easily switch between pieces. Find the domination number for kings, bishops and rooks on a chessboard of your choice.