Assignment 1

2013

Mahboobeh Abdal Mahmoood Abadi & Mehran Nasseri

KTH

4/10/2013

Table of Contents

[1. Sudoku 3](#_Toc352707740)

[1.1. Model 3](#_Toc352707741)

[1.2. Implementation 3](#_Toc352707742)

[1.3. Experiments 3](#_Toc352707743)

[2. N-Queens 5](#_Toc352707744)

[2.1. Model 5](#_Toc352707745)

[2.2. Implementation 5](#_Toc352707746)

[2.3. Experiments 5](#_Toc352707747)

[2.3.1. Branching heuristic 5](#_Toc352707748)

[2.3.2. Comparison with standard model 6](#_Toc352707749)

## Sudoku

## Model

* Variables and domain

We transferred the rules of the Sudoku into variables and constraints. This helped us to model it as a constraint satisfaction problem. Each cell in the Sudoku grid has a single number. So the variables are the Sudoku’s cell, i.e 9\*9 variables and each variable can take a value from 1 to 9.associated variables are shown in figure 1 where

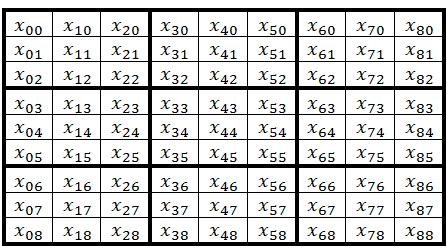


Figure 1

* Constraints
  + Row constraints: each row must be pair-wise different
  + Column constraints: each column must be pair-wise different
  + Block constraint: values in each 3\*3 blocks must be pair-wise different
  + Hints constraints: given hints must be satisfied

## Implementation

Described model implemented using Gecode and attached as Sudoku.ccp. Upon running the application it will solve all provided Sudoku instances with different heuristics and propagator strengths and using DFS exploration. The application will print solution as well as performance information, which includes selected consistency level, number of nodes and depth.

## Experiments

We experimented with different heuristics and propagator strengths on all provided instances of Sudoku. The hint constraints are considered upon their first run under any consistency level. The result of experiment shows that imposing domain consistency on all other 3 constraints and INT\_VAR\_SIZE\_MIN() branching heuristic, leads to lowest search space. Table 1 shows the result of experiment for the Sudoku number 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Min-size branching | | | Max-degree branching | | | Min-degree branching | | | None | | |
| Depth | Node | Fail | Depth | Node | Fail | Depth | Node | Fail | Depth | Node | Fail |
| Default consistency | 11 | 20 | 6 | 8 | 141 | 67 | 8 | 141 | 67 | 8 | 141 | 67 |
| Value consistency | 11 | 20 | 6 | 8 | 141 | 67 | 8 | 141 | 67 | 8 | 141 | 67 |
| Bound consistency | 7 | 9 | 1 | 6 | 15 | 5 | 6 | 15 | 5 | 6 | 15 | 5 |
| Domain consistency | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

Table 1

## N-Queens

## Model

* Variables and domain

The variables are each cell in chess board, so n\*n variables, where n is the number of queens. Each variable can be either 0 or 1. 0 means no queen on that cell and 1 means there is a queen.

* Constraints
  + One queen per row
  + One queen per column
  + At most one queen per diagonal

## Implementation

Described model implemented using Gecode and attached as nqueens.ccp. Upon running the application it will find all solution for a given ‘n’ using DFS exploration (Gist version is also available and should only be uncommented to be active). The application will print all distinct solutions as well as performance information.

## Experiments

## Branching heuristic

Experiment on different branching heuristics proves that the best heuristics for choosing the variable ordering is to choose a variable with maximum.

* The maximum degree heuristic:
  + Chooses the most constrained variable. Such variable is more likely to fail.
  + Value ordering for such variable selection strategy is to select maximum value
    - If value 1 is not feasible the propagation strategies become aware of this soon due to large number of constraints on that variable.
  + This heuristic reduces the search space early on.

## Comparison with standard model

|  |  |  |  |
| --- | --- | --- | --- |
| Standard model | | Cell based model (matrix) | |
| Pros | Cons | Pros | Cons |
| N variable |  |  | N^2 variable |
|  | Domain range = n^2 | Domain range = 2.  {0, 1} |  |
|  | O(n^2) constraints | 6n-4 constraints. So scale better and increase linearly as the number of variables increase |  |
| Small search space size, so faster than cell based. (n^2)^n.\* |  |  | Large search space size. 2^(n^2) |

\* The search space= domain ^ variables. So a model with lower number of variables is preferred.