COMP 440 Homework 3

Tony Chen(xc12) and Adam Wang(sw33)

October 2016

1 Sudoku and constraint satisfaction

- Constraint:
  1. $X_{ij} = x_{ij} \forall \{\text{pre-filled cell } X_{ij}\}$, where x_{ij} is X_{ij}'s pre-filled value
  2. $AllDiff(X_i) \forall \{\text{row } X_i\}$
  3. $AllDiff(X_j) \forall \{\text{column } X_j\}$
  4. $AllDiff(X_k) \forall \{\text{box } X_k\}$

- Forward checking eliminates all the inconsistent value w.r.t. the pre-filled cells from the domains of empty cells.
  For cell X_{74}, 3 will be eliminated from its domain because of X_{42} or X_{95}; 8 will be eliminated from its domain because of X_{44} or X_{86} or X_{78}; 9 will be eliminated from its domain because of X_{54} or X_{76}; 7 will be eliminated from its domain because of X_{84} or X_{77}; 2 will be eliminated from its domain because of X_{85}; 6 will be eliminated from its domain because of X_{94}. So the domain of cell X_{74} becomes $\{1, 4, 5\}$.

- Most-constrained variable heuristic will choose the variable with the smallest domain and thus enable the algorithm to detect failure sooner. This is particularly effective in the case of Sudoku because the constraint for each cell involves 20 other cells and the initial domain size is just 9; also since there are only 81 cells, the time and space complexity to get the cell with the smallest domain can be viewed as constant.
  For the example grid, the cell with the smallest domain are cell X_{18} with domain $\{9\}$, cell X_{64} with domain $\{5\}$, cell X_{96} with domain $\{1\}$, and cell X_{58} with domain $\{5\}$. So any one of them could be chosen to be assigned first.

- Because all other values in the same box as X_{48} cannot have value 7 and there must be a 7 in that box.
  Yes it can.
  After forward checking, cell X_{18} will have domain $\{9\}$, cell X_{48} will have domain $\{4, 5, 7\}$, cell X_{58} will have domain $\{5\}$, and cell X_{98} will have domain $\{4, 9\}$. Then we enforce arc consistency on (X_{48}, X_{58}), which shrinks X_{48}'s domain to $\{4, 7\}$; then we enforce arc consistency on (X_{98}, X_{18}), which shrinks X_{98}'s domain to $\{4\}$; then we enforce arc consistency on (X_{48}, X_{98}), which shrinks

X_{48}'s domain to {7}. Therefore, after forward checking and enforcing arc consistency, the only value can be assigned to cell X_{48} is 7.

2 Constraint satisfaction with non-binary factors

- Define A as $(A_1, ..., A_k)$ with domain S. A has unary factor on itself such that $A_1, ..., A_k$ are constrained by the k-nary factor over $X_1, ..., X_k$. A also has k binary factors $A_i = X_i \forall 1 \leq i \leq k$. Note that the unary factor is not k-nary because it involve one variable with k components of invariant values, instead of k variable.

- The binary factors are $A_1 = X_1$, $A_2 = X_2$, $A_3 = X_3$.
  The initial domain of A: $\{(red, red, red), (red, red, blue), (red, red, green), (red, blue, red), (red, blue, blue), (red, blue, green), (red, green, red), (red, green, blue), (red, green, green), (blue, red, red), (blue, red, blue), (blue, red, green), (blue, blue, red), (blue, blue, blue), (blue, blue, green), (blue, green, red), (blue, green, blue), (blue, green, green), (green, red, red), (green, red, blue), (green, red, green), (green, blue, red), (green, blue, blue), (green, blue, green), (green, green, red), (green, green, blue), (green, green, green)\}$. After enforcing consistent over the unary factor on each possible value from A's original domain, A's domain becomes $\{(red, green, green), (blue, green, green), (green, red, green), (green, blue, green), (green, green, red), (green, green, blue)\}$.

- Variables are X_{ij} where $1 \leq i, j \leq 3$ without X_{22}.
  Domain for each variable is $\{C, A, P, T, I, N, E\}$
  Factors are:
  $w_1 = Cat(X_{11}, X_{12}, X_{13})$
  $w_2 = Cat(X_{11}, X_{21}, X_{31})$
  $w_3 = Cat(X_{31}, X_{32}, X_{33})$
  $w_4 = Cat(X_{13}, X_{23}, X_{33})$
  $w_1 \in D$
  $w_2 \in D$
  $w_3 \in D$
  $w_4 \in D$

- Each auxiliary variable will represent one of the four top-down/left-right stripe.
  Each auxiliary variable's domain is the possible solution on stripe that auxiliary variable stands for.

3 Constrait solving and course scheduling

4 Sudoku and repair algorithms

A local solver using the min-conflicts heuristic would not do very well on Sudoku problems. For the min-conflicts heuristic, the algorithm generates a random initial state and then picks a random variable and tries to change it to a value that can satisfy most constraints. First, for variables that have only 1 unique

value in the solution if we use forward checking and enforce arc consistency, repairing algorithm with min-conflicts heuristic does not neccessarily fix the variable to that unique value. Second, since for each variable the constraits are associated with all the variables in the entire row, column and box, it is very likely that fixing a value to satisfy some constraits in a box will cause violation on constraits on the row or column, so a min-conflicts heuristic does not do much for us as there are not many frequent changes to constrait in this problem. Third, repairing algorithms are not complete, which means it is possible that it may not give us a solution when there exists one, while a constructive approach is guaranteed to give us a solution if there is one given enough amount of time. So I would recommend a constructive approach for solving Sudoku.