SUDOKU SOLVER

COMPSCI 271 Project

Liangjian Chen
Yu Guo
Shengnan Wang
Shutao Xiao

Outline

- Introduction
- Method
  - Backtracking without constraint
  - Optimization
    - Constraint propagation
    - Heuristic
      - Minimum-remaining-values (MRV)
      - Least-constraining-value (LCV)
  - Dancing links
- Current results
- Future work

Introduction

Sudoku: Fill in all the empty squares such that no digit appears twice in any row, column, or 3 × 3 subbox.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Initial State

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Goal State

Problem definition

Variables: Using the variable names *A1* through *A9* for the top row (left to right), down to *I1* through *I9* for the bottom row.

Domains: The empty squares have the domain {1, 2, 3, 4, 5, 6, 7, 8, 9} and the pre-filled squares have a domain consisting of a single value.

Constraints:

Alldiff (*A1, A2, A3, A4, A5, A6, A7, A8, A9*)

Alldiff (*B1, B2, B3, B4, B5, B6, B7, B8, B9*)

…

Alldiff (*A1, B1, C1, D1, E1, F1, G1, H1, I1*)

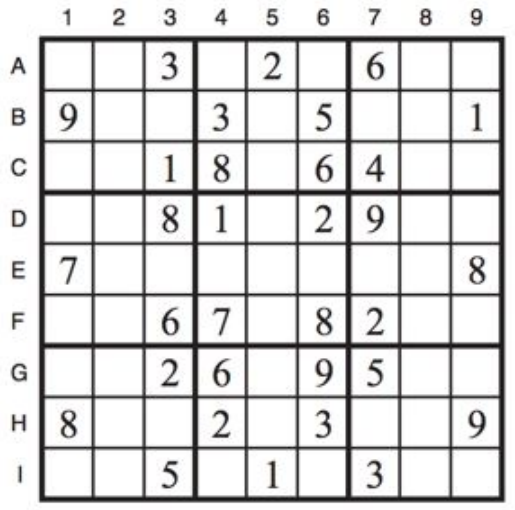
Alldiff (*A2, B2, C2, D2, E2, F2, G2, H2, I2*)

…

Alldiff (*A1, A2, A3, B1, B2, B3, C1, C2, C3*)

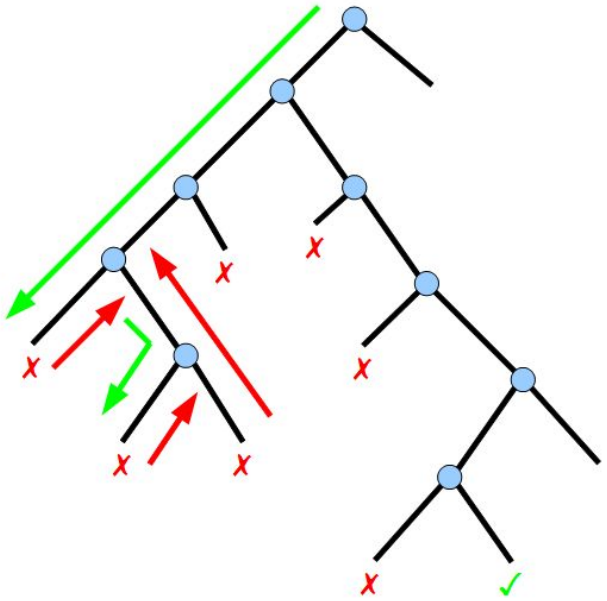
Alldiff (*A4, A5, A6, B4, B5, B6, C4, C5, C6*)

…



Method: CSP

- Backtracking:
  - Backtracking search is used for a depth-first search that chooses values for one variable at a time and backtracks when a variable has no legal values left to assign.



Method: CSP

- Constraint Propagation:
  - If a square has only one possible value, then eliminate that value from the other 20 squares in the same row, column, or subbox.
  - If a unit has only one possible place for a value, then put the value there.

Ex: A1, A2, A3, A4, A5, A6, A7, A8, A9

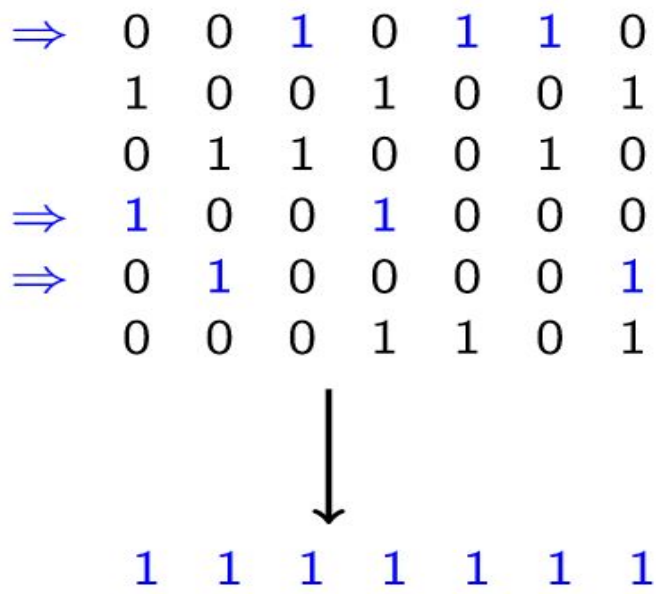
1. A1: 7
2. if it none of A3 ~ A9 has a 3  A2 = 3

Method: CSP

- Heuristic:
  - MRV: Choosing the variable with the fewest “legal” values. It picks a variable that is most likely to cause a failure soon, thereby pruning the search tree.
  - LCV: Once a variable has been selected, the algorithm must decide on the order in which to examine its values. This heuristic is trying to leave the maximum flexibility for subsequent variable assignments.

Algorithm X with Dance-Link(DLX)

- Exact Cover Problem(ECP): what is it ?
  - Two ways
  - Constraint Satisfied Problem
    - n variables $X_1, X_2, \ldots, X_n$
    - m constraints
      - A set B contains some variables above
      - Exact one variable in B is $True$
  - Matrix Cover
    - Pick some rows
    - All columns is covered
    - Each column is covered by one row

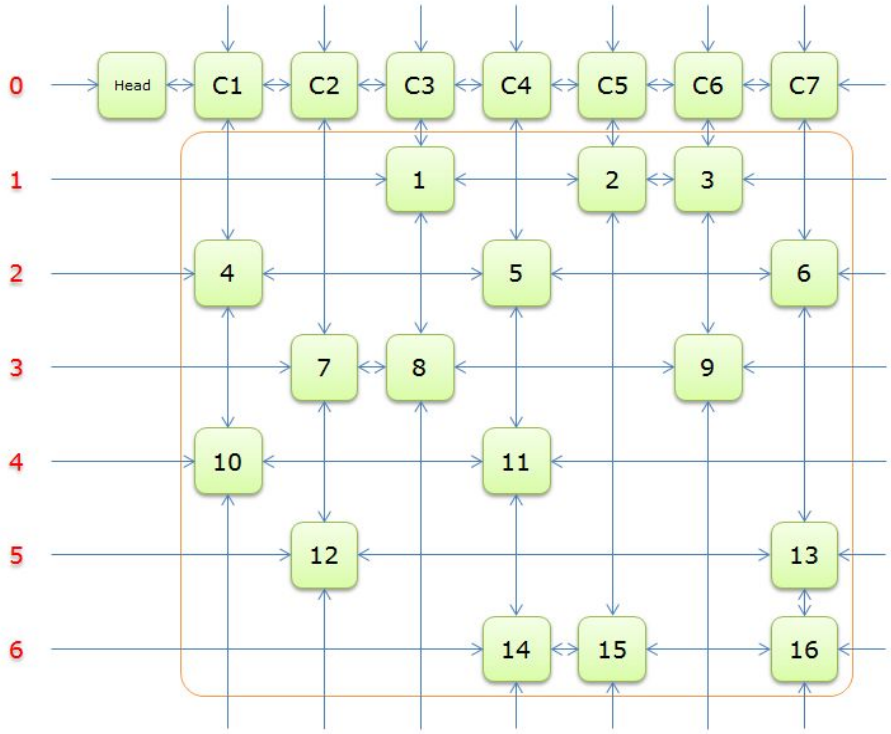


Convert Sudoku to ECP

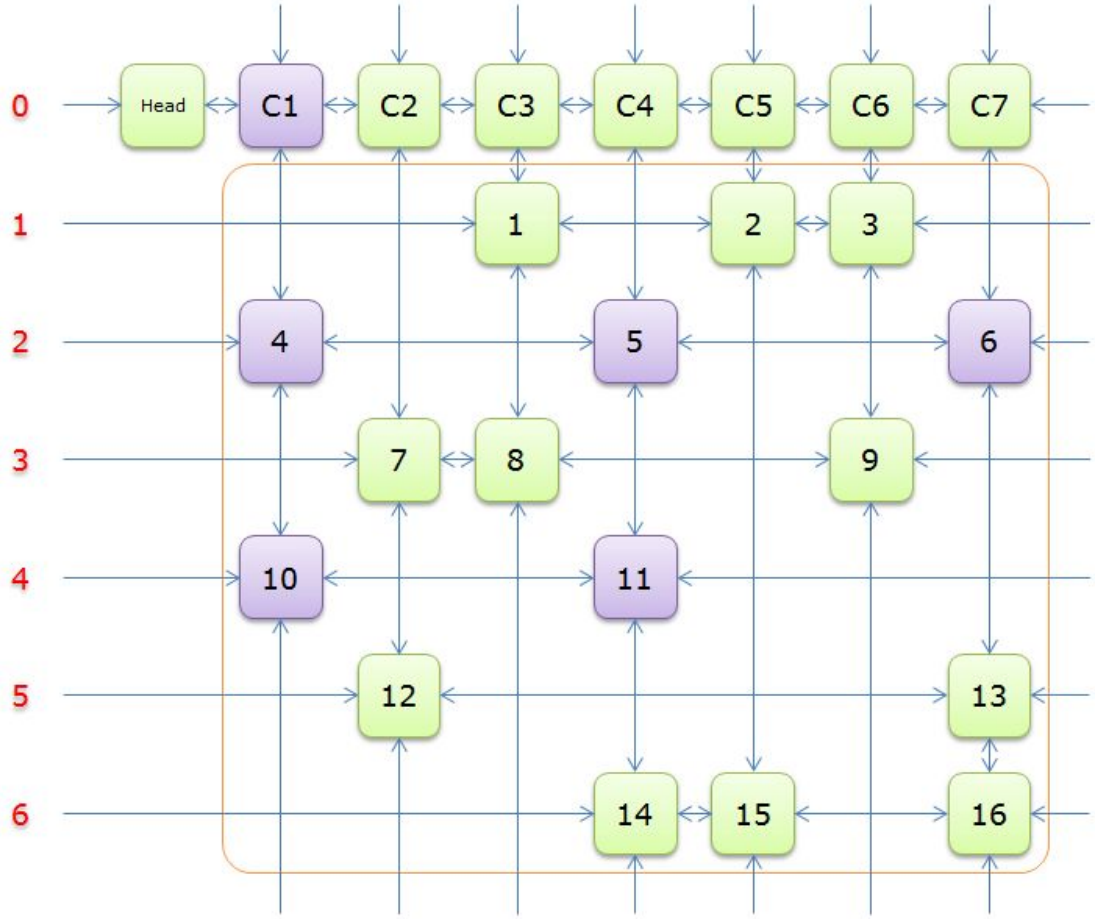
- CSP problem
- Variable
  - Binary variable $A_{i,j,k}$ means i^{th} row j^{th} column is K
- Constrain
  - Row : each number appear in each Row once
    - Any fixed i, k, constraint $B = \{A_{i,j,k} | \, j = 0,1, \dots, 8\}$
  - Column: each number appear in each Column once
    - Any fixed j, k, constraint $B = \{A_{i,j,k} | \, i = 0,1, \dots, 8\}$
  - 3 * 3 grid: each number appear in each 3 * 3 once
    - Any fixed $k, 3 * 3 \; grid$ constraint $B = \{A_{i,j,k} | \, i, j \; is \; in \; that \; 3 * 3 \; grid\}$
  - One number per cell
    - Any fixed i, j, a constraint $B = \{A_{i,j,k} | \, k = 1,2, \dots, 9\}$

DLX- What is it?

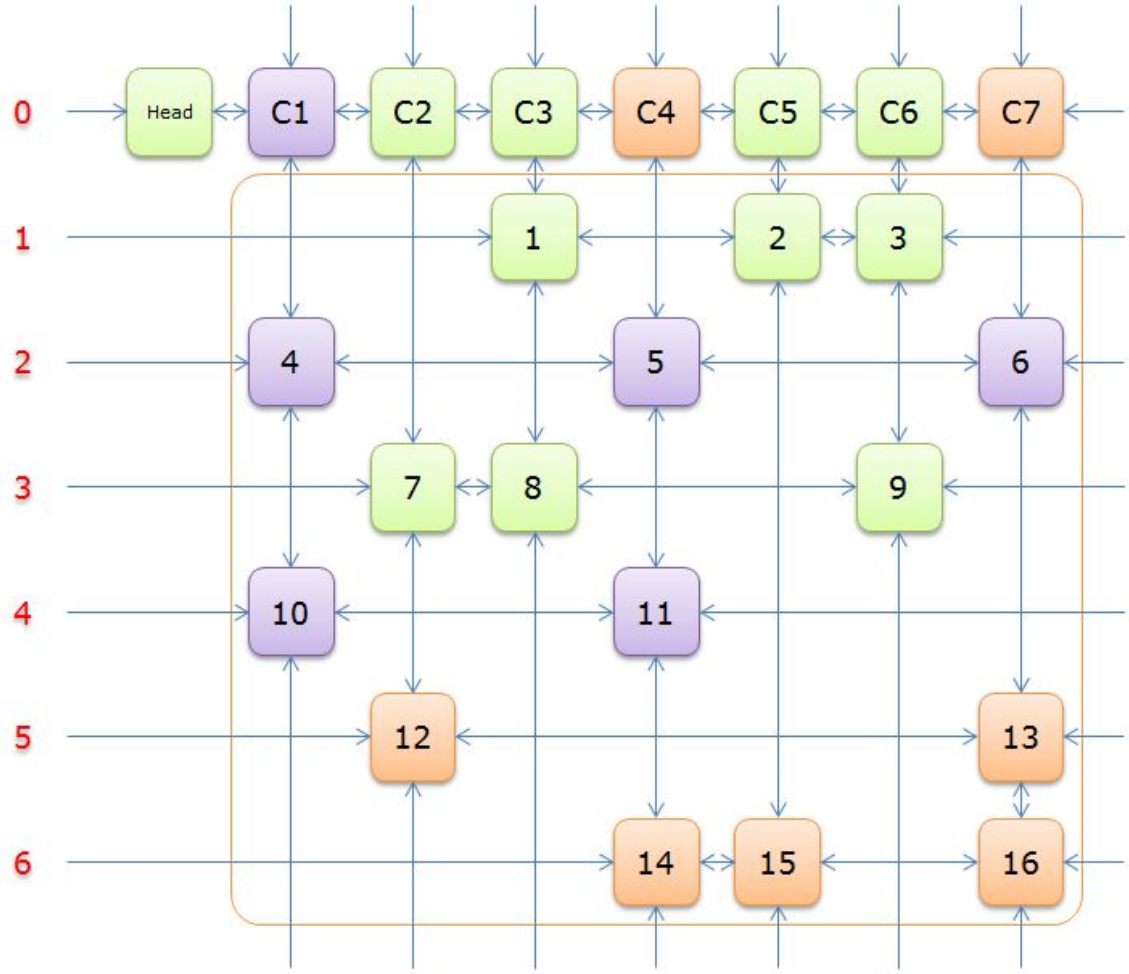
- Donald.E.Knuth in 2000
- A represent of Matrix
  - Bidirectional Cross link list
- Main nodes
  - Only keep 1
- Accessory nodes
  - head
    - check empty
  - Column list
    - How many nodes



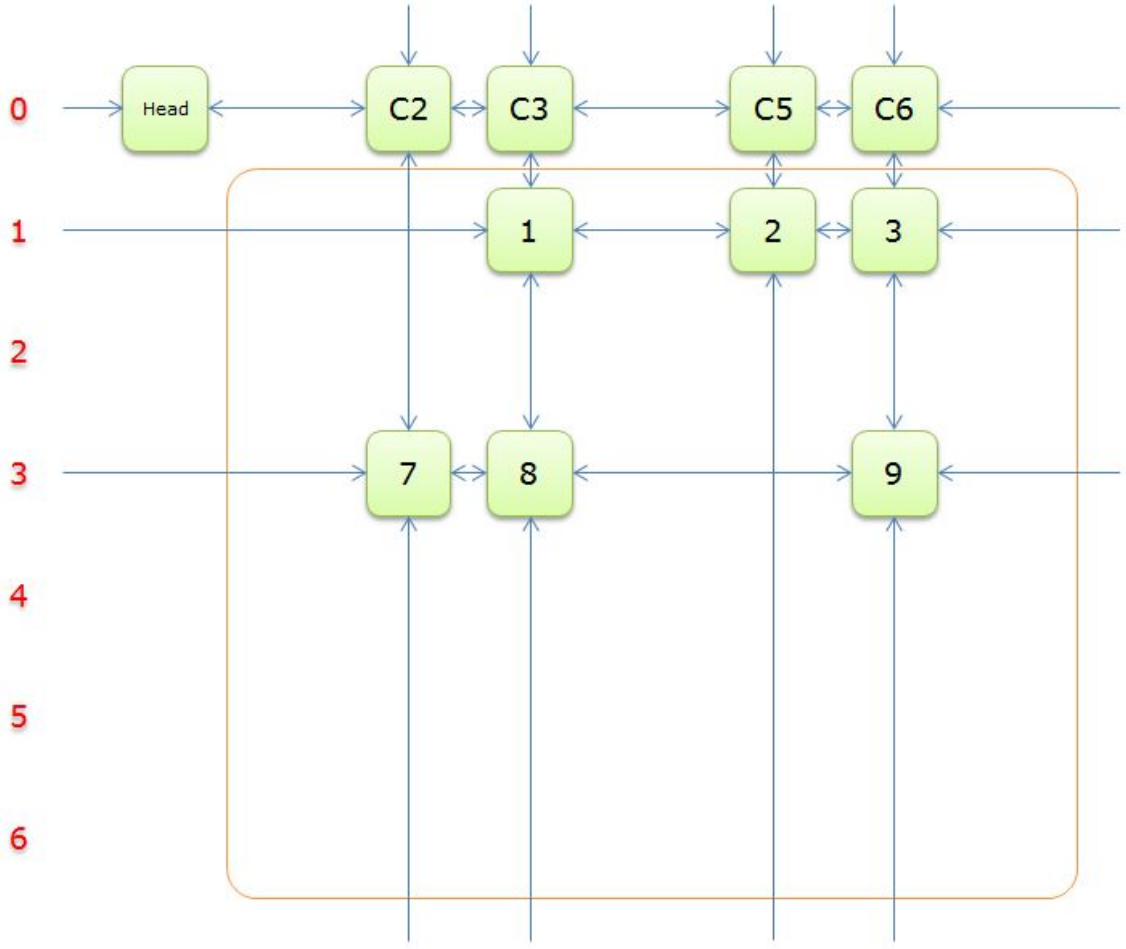
DLX -- How it works



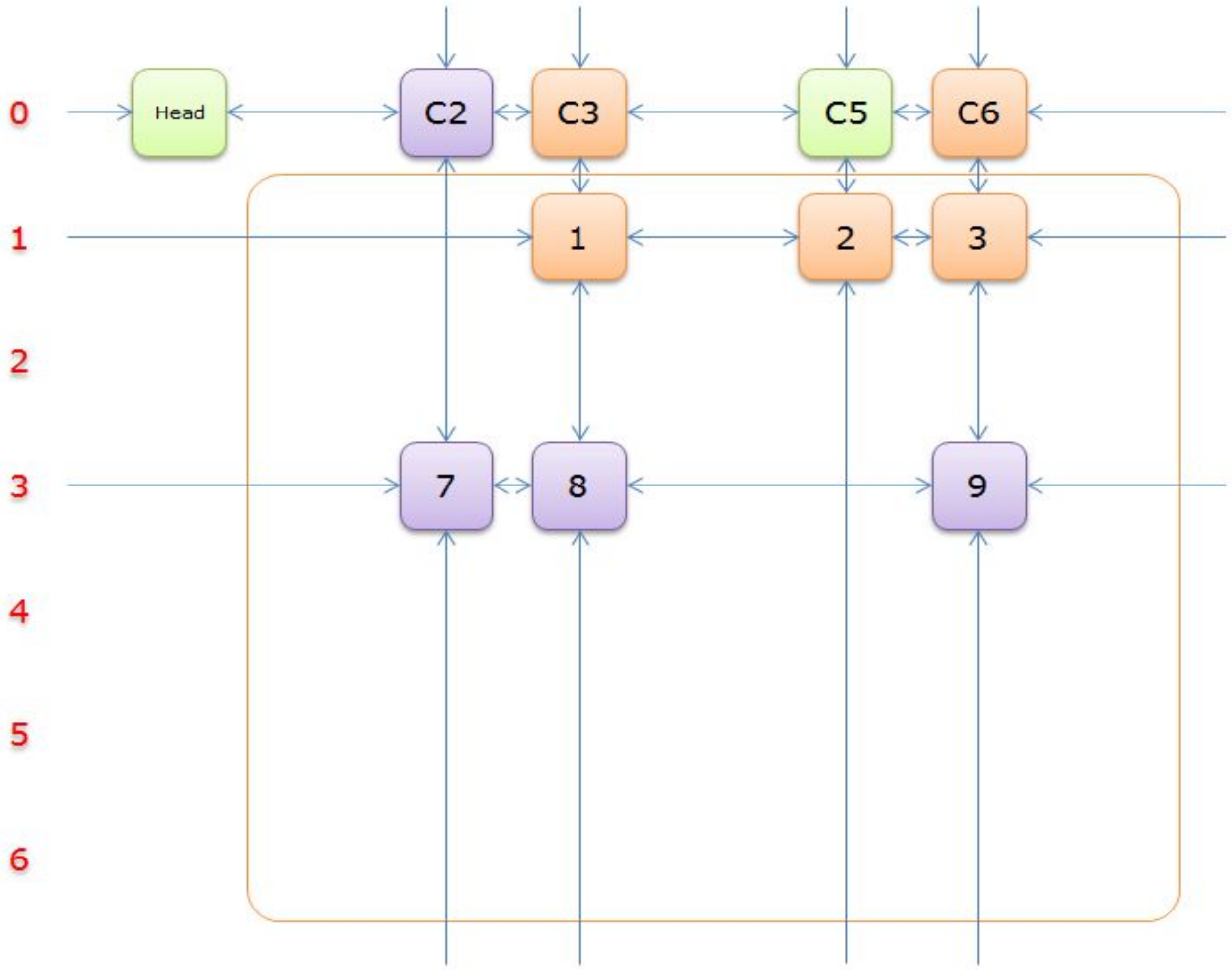
DLX -- How it works



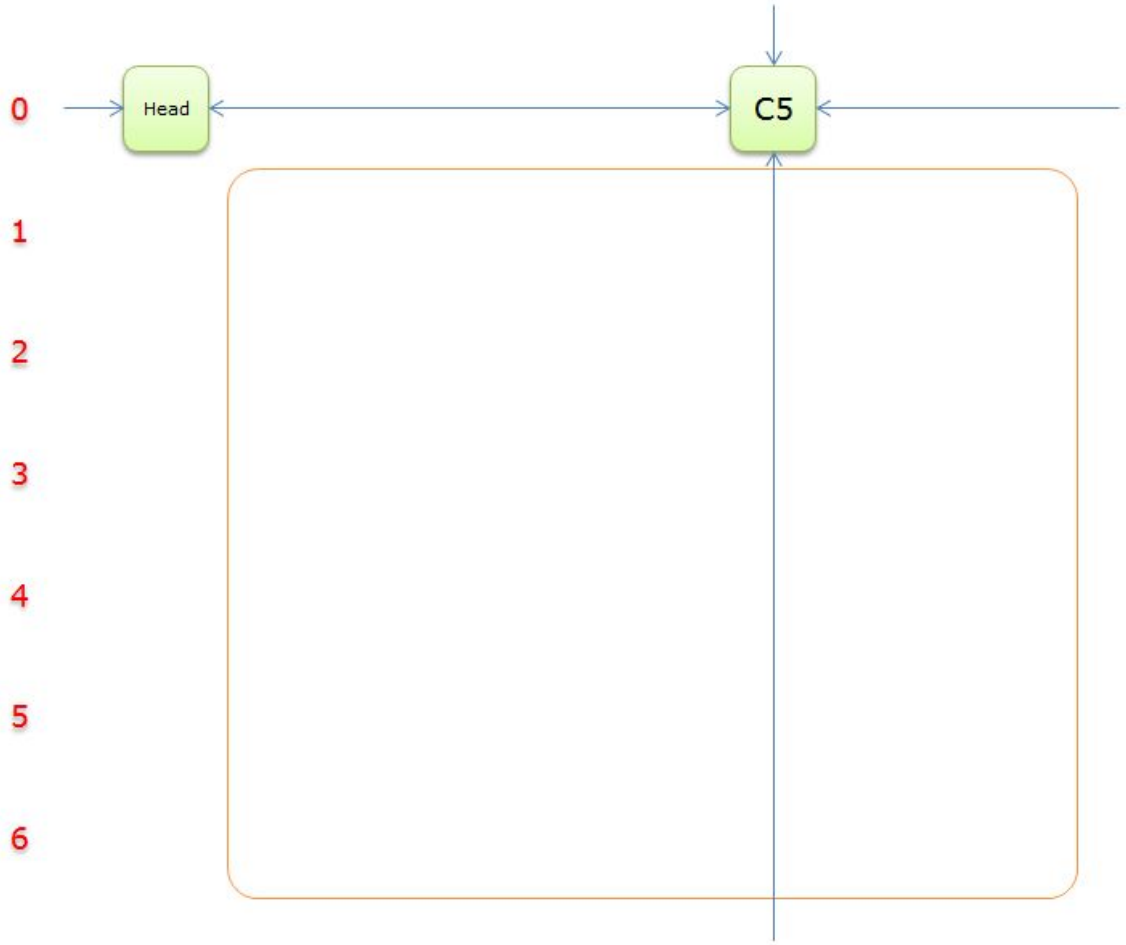
DLX -- How it works



DLX -- How it works



DLX -- How it works



DLX--why it works

- Extremely efficient
  - Delete
    - a->l->r = a->r
    - a->r->l = a->l
  - Insert
    - a->l->r = a->r->l = a
- DLX is suitable to back-tracking

Comparation

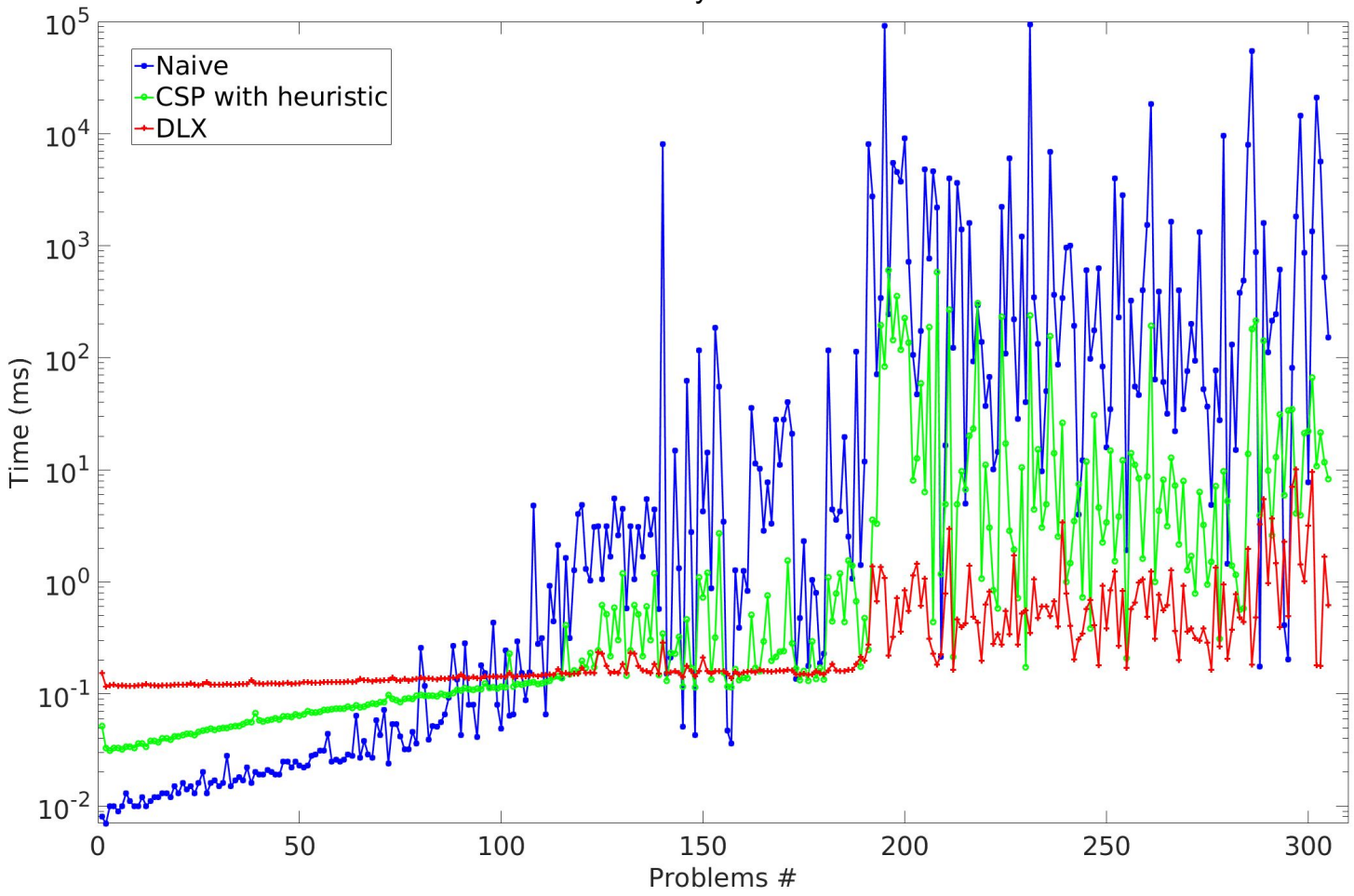
- Text case: 50 difficult Sudoku problems online.
- Test in 2013 Macbook pro 15’

Algorithm	Algorithm X with DLX	CSP with heuristic	Naive DFS
Running Time	0.04s	2.54s	> 10s

CSP with heuristic using MRV, LCV, Forward checking.

More Results

OS: Ubuntu 14.04
CPU: Intel® Core™2 Quad CPU Q9400 @ 2.66GHz × 4
Memory: 8G



Future work

- Add more constraints:
  - Arc-consistency
  - Path-consistency
  - K-consistency
- Run more tests with various inputs in different difficulty levels
- Finish the report

ຂໍຂອບໃຈທ່ານ
សូមអរគុណអ្នក
ಧನ್ಯವಾದ
ви благодариме
شكرا
təşəkkür
köszönöm
gràcies
спасибо
grazas
eskerrik asko
nirringrazzjak
gratias agimus tibi
谢谢
kiitos
di ou mèsi
dank je
dankon
danke
gracias
paldies
dank u
cảm ơn bạn
dankie
tack
aitäh
thank you
þakka þér
ありがとう
mochchakkeram
teşekkür ederim
σας ευχαριστώ
ngiyabonga
grazie
merci
děkuji
благодаря
תודה
આભાર
kop khun krap
dziękuję
дзякуй
go raibh maith agat
감사합니다
ačiū
asante
hvala
хвала
धन्यवाद
sukriya
terima kasih
ďakujem
takk
mulţumesc
obrigado
buíochas a ghabháil leat
salamat
diolch i chi
দ্বারা
তোমাকে ধন্যবাদ
falemnderit
շնորհակալություն