DEEP NEURAL NETWORKS FOR CONSTRAINT SATISFACTION PROBLEMS: AN INITIAL INVESTIGATION

ANDREA GALASSI, MICHELE LOMBARDI, PAOLA MELLO, MICHELA MILANO

UNIVERSITY OF BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DISI)

SUCCESS OF DEEP NEURAL NETWORKS

Computer Vision

- "Deep residual learning for image recognition", He et al., 2016

Text Classification and NLP

- "Very Deep Convolutional Networks for Text Classification", Conneau et al., 2017

Game Playing

- "Mastering the game of go without human knowledge", Silver et al., 2017
- **"Can Deep Networks Learn to Play by the Rules? A Case Study on Nine Men's Morris", Chesani, Galassi, Lippi, Mello, 2018**

PREVIOUS APPROACHES: NNS AND CSPS

Use of neural networks and Hopfield networks to solve generic CSPs.

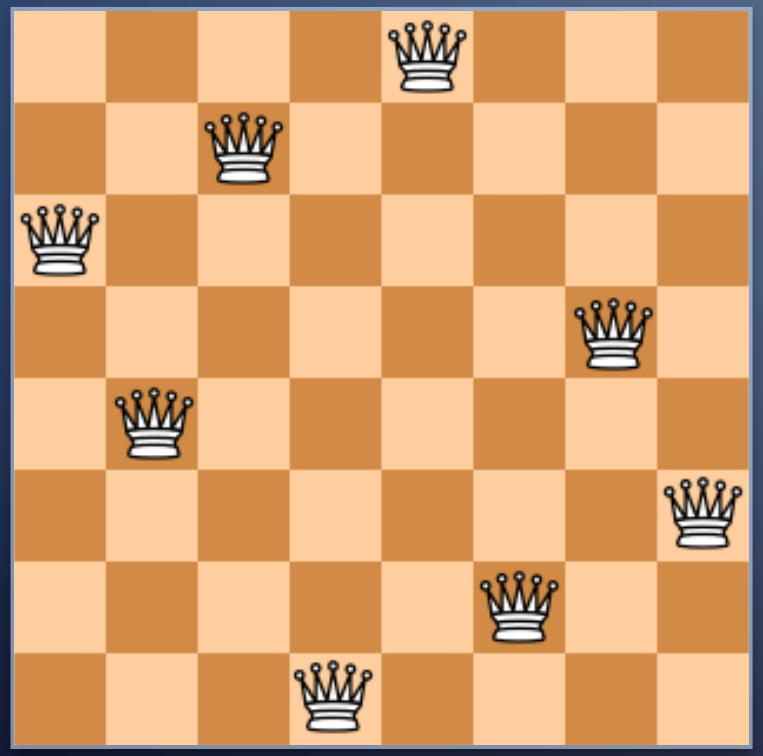
The methods were general, but the networks topologies had to be chosen according to the problem, modeling a constraint as connection between neurons with a specific weight.

It is necessary to have a **complete knowledge** of the problem.

CAN A DNN LEARN TO SOLVE A GENERAL PROBLEM FROM SCRATCH?

- We trained a DNN in a **supervised** fashion to **constructively** solve a problem: given a partial solution, **predict the next feasible variable assignment**
- The setting is as most problem agnostic as possible
- Practical applications:
  - Using a DNNs to solve a problem by itself
  - Create an hybrid system where the network guide a search or its choices are filtered by an expert system

CASE STUDY: N-QUEENS

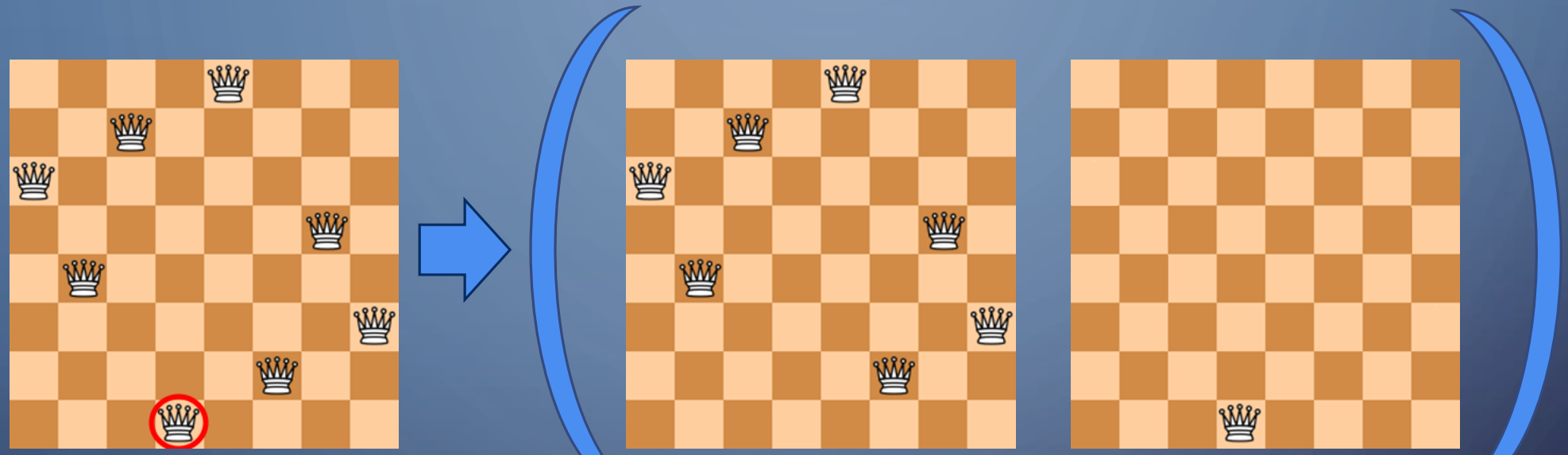


- Place N queens in a NxN chessboard, so that no queen threaten any other
- Usually this problem is represented as N variables that can assume N possible value

DATASET GENERATION: SOLUTIONS (1/2)

- We have taken the 12 asymmetrical solutions of the 8-Queens problem
- We have expanded them into the 92 total solution of the problem
- We have iteratively decomposed the solutions in about 90,000 partial solutions:
  - A single queen is removed from the chessboard of a solution: the incomplete chessboard is a partial solution that will be used as input, while the removed queen will be its target
  - Starting with the 92 complete solution, this procedure is applied to every partial solution, in every possible order

DATASET GENERATION: SOLUTIONS (2/2)



Solution
(complete or partial)

Input State

Target

DATASET GENERATION: TRAINING AND TEST SPLIT

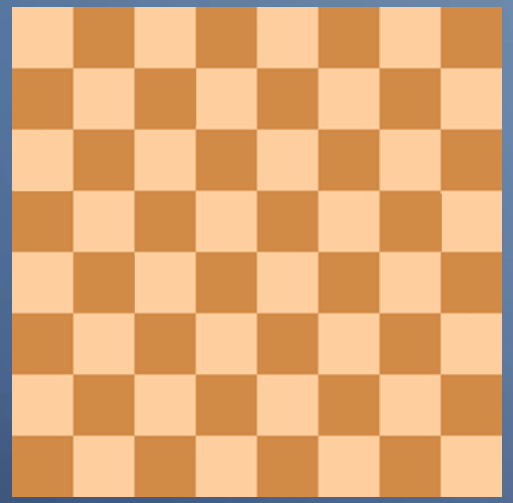
- We have generated three families of dataset, depending on when the split between training and test solutions has been done:
  A. The 12 asymmetrical solutions
  B. The 92 solutions
  C. The over 90,000 partial solutions

# of training solutions / # of test solutions = (about) 1 / 3

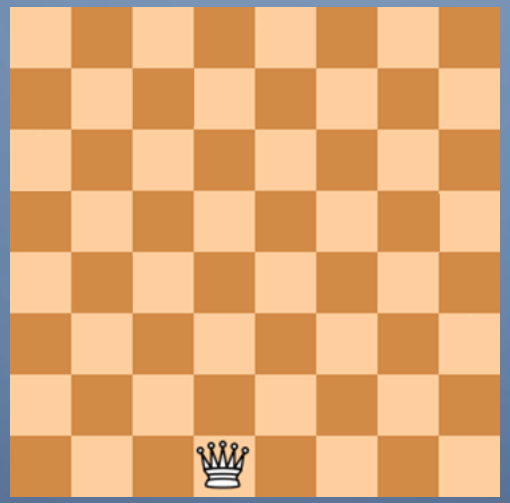
DATASET GENERATION: TRAINING COUPLES (1/3)

- For each partial solutions, there are many possible targets
  - i.e. every position of the chessboard is a possible target for the empty board
- We have generated three families of dataset (orthogonal to the previous three) depending on how we have managed partial solutions with multiple targets:
  - **U**NIQUE: only one target (chosen randomly) is kept. The others are discarded.
  - **M**ULTIPLE: Each possible target is kept, but a different couple is made for each one.
  - **C**OLLAPSED: Each possible target is kept, assigning to each one a uniform probability.

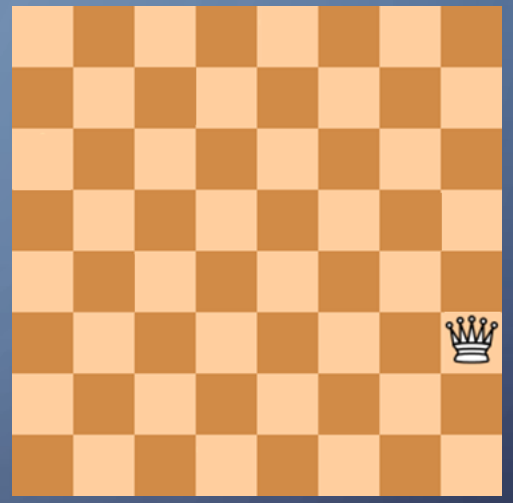
DATASET GENERATION: TRAINING COUPLES (2/3)



Input State

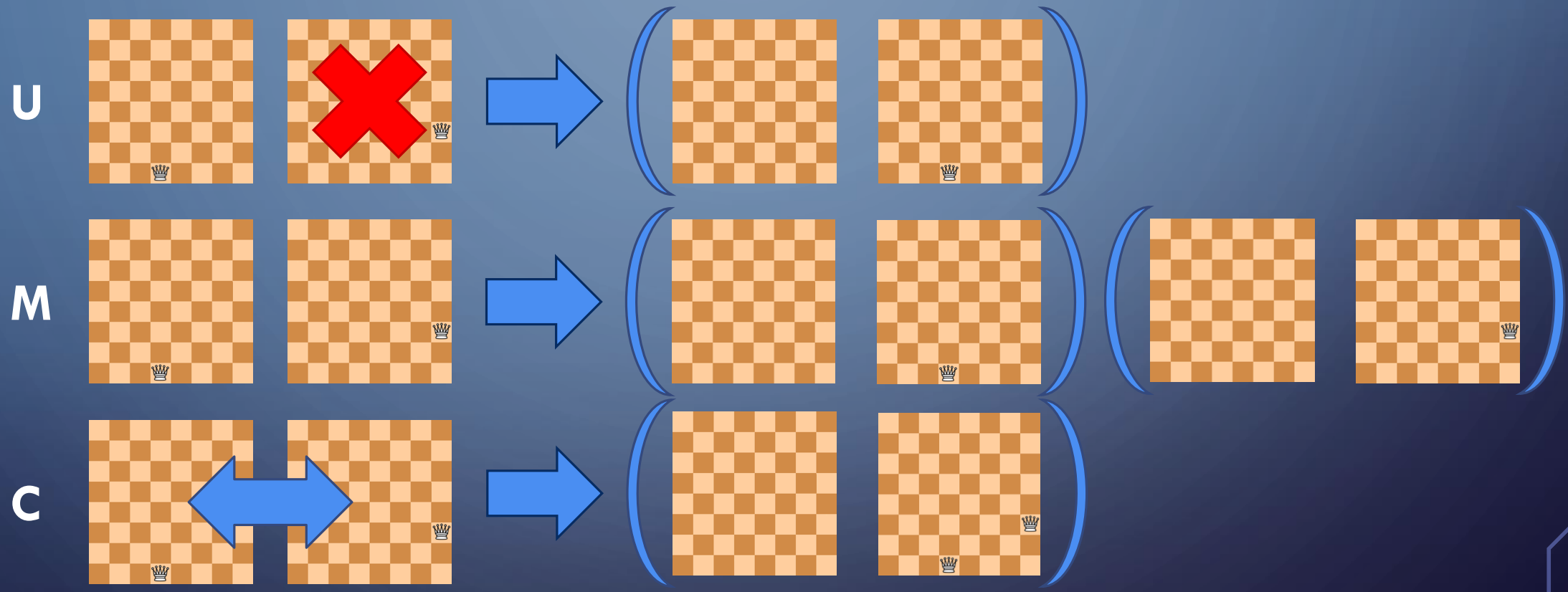


Possible Target 1



Possible Target 2

DATASET GENERATION: TRAINING COUPLES (3/3)

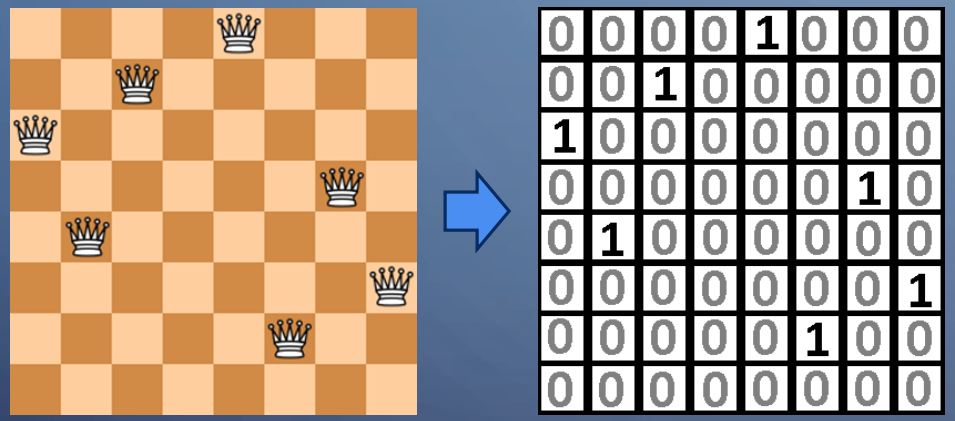


MODEL (1/2)

- **Problem agnostic representation**: arrays of size N^2
  - Each bit is a position on the board.
- Input: a partial assignment of the variables
  - If a bit is set to 1, then its corresponding position is occupied by a queen
- Target: the correct assignment of a variable not yet assigned
- Output: probability distribution over each possible value of each varibale
  - Even the already assigned ones!

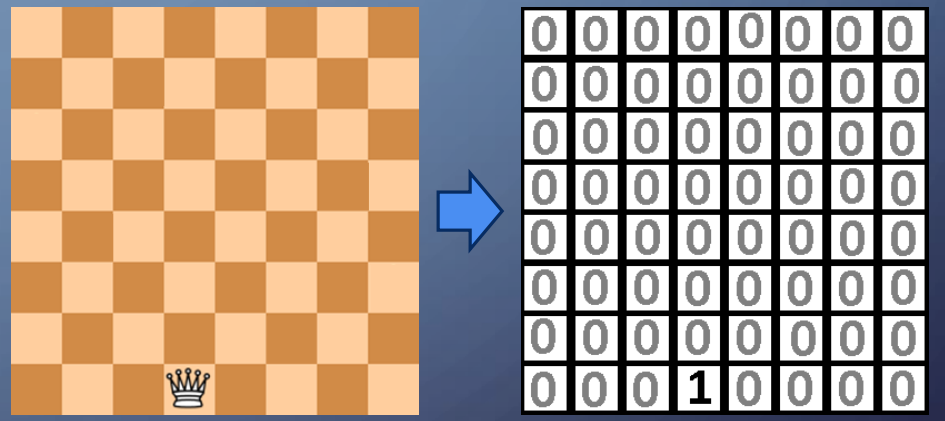
MODEL (2/2)

Input



0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0

Target



0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0

DEEP NETWORKS AND TRAINING

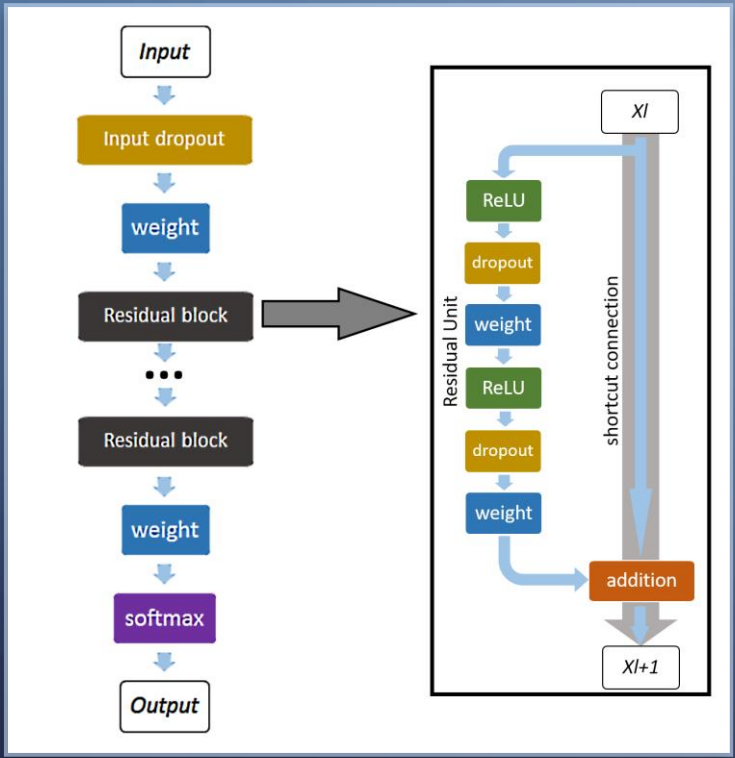


Image from "Can Deep Networks Learn to Play by the Rules? A Case Study on Nine Men's Morris"

- Full-connected model (not convolutional)
- Pre-activated Residual Network architecture
- More than 100 layers, alternating 500 and 200 neurons
  - The last one is a softmax classifier
- Rectifier activation function
- Use of drop out

TEST

We have assested the ability of the networks to solve the problem with two experimental settings:

1) **Stochastic generation of a whole solution**
   Given an empty board, a queen is placed according to the probability scored by the DNNs. For 7 times more, the new board is used as input.
   We measure the ability of the networks to create a complete solution from scratch

2) **Feasible single assigment**
   Given a partial solution from the test set, we evaluate if the network is capable of evaluating as most probable a correct assignment

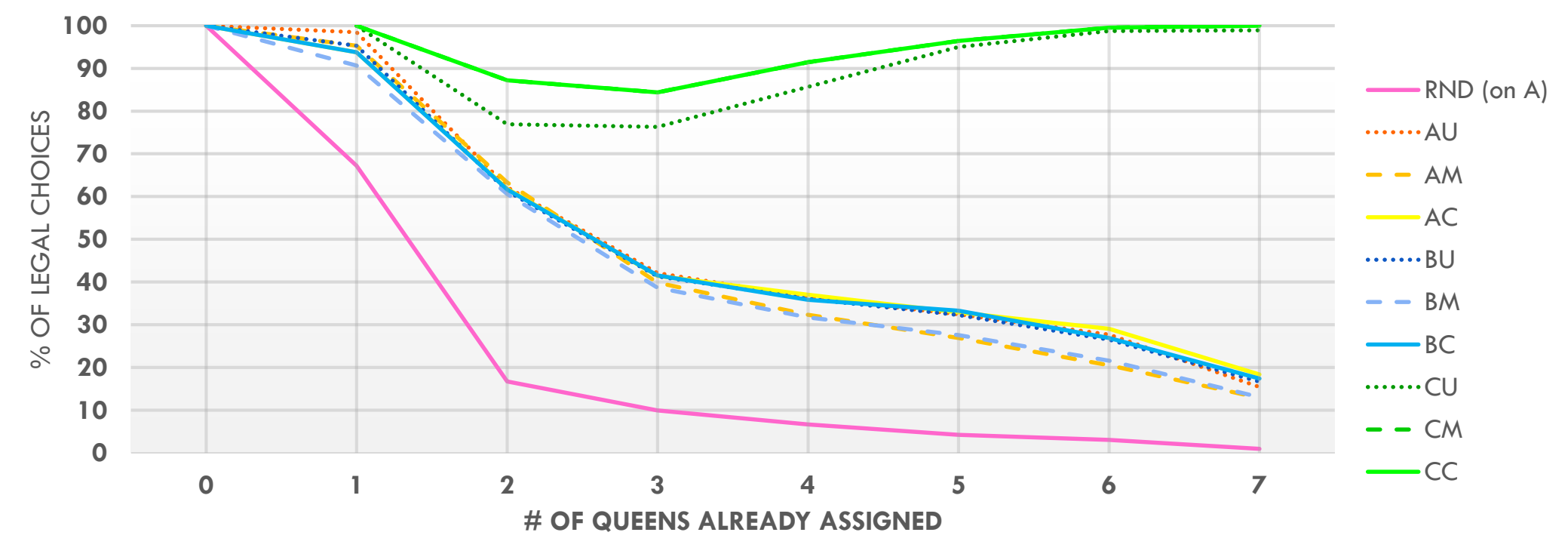
STOCHASTIC GENERATION AND COST

Average correction cost (minimum # of queens that have to be moved) of the solution generated through 8 interaction starting with an empty board.

Training dataset	A	B	C
UNIQUE	0.36	0.51	1.40
COLLAPSED	0.32	0.38	1.99
MULTIPLE	1.49	1.44	1.25

- Using completely **random** choices, the average cost is 4.72

FEASIBLE SINGLE ASSIGNMENT ON TEST SETS



ANOTHER CASE OF STUDY: PLS (1/2)

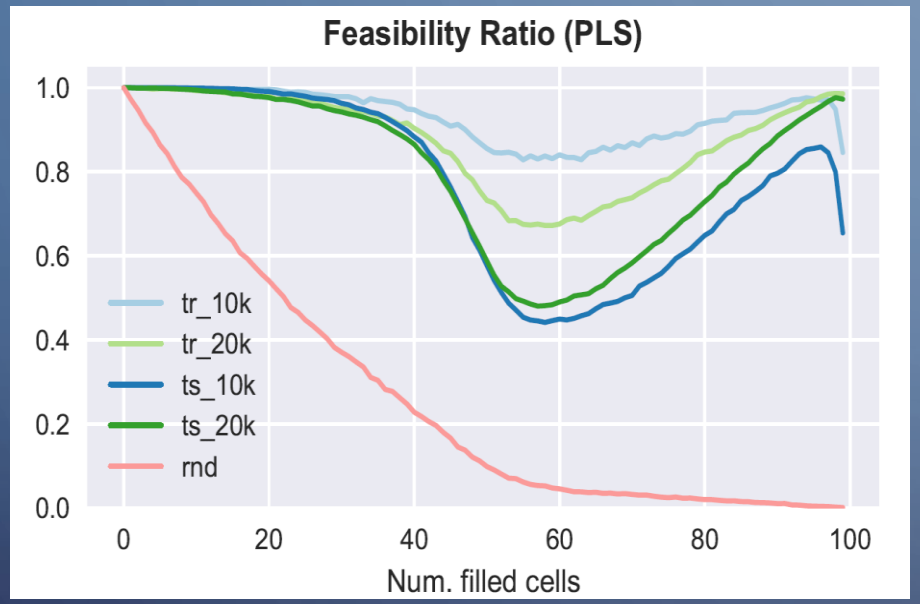
"Model agnostic solution of CSPs via Deep Learning: a preliminary study"

Galassi, Lombardi, Mello, Milano

to be presented at CPAIOR18

- We have extended our research to a bigger problem: the Partial Latin Square problem
- Use of a methodology similar to the A-Unique dataset generation but with very few example w.r.t. the number of possibile partial solutions
- Use of DNNs to guide a simple search to solve 4000 partial solutions, evaluating the number of fails

ANOTHER CASE OF STUDY: PLS (2/2)



Search:

- Peak of fail set at 10^4
- The Random baseline reached the peak 212 times
- The best network reached it only 137 times

CONCLUSION

- Can DNNs learn to solve problems from scratch? YES!

  To some degree…

- The DNNs ability to learn high level patterns could be indeed useful to solve problems without having the need to explicitly formulate them
- The training method highly influence the networks
- But there is still a lot of work to do…

FUTURE STEPS

- Independence of the architecture from the problem size
  - Full-convolutional NN model?
- Transfer learning between different size of the problem
  - Pre-train on data from size M and training with data from size N
- Exploit networks as part of a hybrid system
- Use the networks to construct a full solution in a single step

THANK YOU FOR YOUR ATTENTION