Model de soluţie agnostică a CSP-urilor prin învăţare aprofundată: un studiu preliminar

1. Introducere

În această lucrare, cercetăm ideea că DNN-urile ar putea fi capabile să înveţe cum să rezolve probleme combinatorii, fără informaţii explicite despre constrângerile problemei.

În principiu, o astfel de reţea ar putea fi utilizată pentru a ghida un proces de căutare, de ex. să ţină cont de constrângerile (soft) care sunt implicite în soluţii trecute. Utilizarea DNN poate, de asemenea, să asigure o accelerare la rezolvarea mai multor instanţe ale aceleiaşi probleme.

In urma acestui studiu, avem rezultate intrigante cu privire la Partial Latin Square and N-Queen,două probleme clasice de satisfacţie a constrângerilor (CSP). Cel mai interesant rezultat se reflecta in discrepanţa impresionantă între precizia DNN (scăzută) şi capacitatea sa (foarte mare) de a genera sarcini fezabile: acest lucru sugerează că reţeaua învaţă într-adevăr ceva despre structura problemei, chiar dacă a fost instruit pentru a „imita" secvenţe specifice de construcţie a soluţiei.

Alte studii de caz in care s-au folosit DNN pentru a rezolva CSP-urile:

1. In *"Guarded Discrete Stochastic networks"*, se pot rezolva CSP-urile generice într-un mod nesupravegheat:
   -se bazează pe o reţea Hopfield pentru a găsi o alocare constantă a variabilelor şi pe o reţea „de pază" pentru a forţa alocarea tuturor variabilelor.
   -metoda GENET poate construi reţele neuronale capabile să rezolve CSP-uri binare şi a fost extinsă ulterior în EGENET,pentru a susţine CSP-uri non-binare;

2. In articolul *"Scheduling meeting solved by neural network and min-conflict heuristic"* ,un CSP este reformulat pentru prima dată ca o problemă de optimizare quadratică:
   -apoi, o euristica este folosita pentru a ghida evoluţia unei reţele de Hopfield

dintr-o stare iniţială reprezentând o soluţie inefabilă până la o stare finală fezabilă;
-toate aceste metode se bazează pe o cunoaştere deplină a constrângerilor de problemă pentru crearea atât a structurii, cât şi a greutăţilor reţelelor;

2. Abordarea generala si pregatirea

2.1 Abordare generala

- folosim vectori de biţi simpli atât pentru input-ul reţelei cât şi pentru output; pentru o variabilă cu n valori rezervăm n biţi;
- ridicarea bitului i corespunde atribuirii valorii domeniului i; dacă nu este ridicat niciun bit, variabila este neasignată;
- ca un dezavantaj major, metoda este în prezent restrânsă la probleme cu o dimensiune prestabilită;

In acest studiu de caz s-au folosit 2 strategii: strategii diferite, denumite deconstrucţii aleatorii şi sistematice:

Alg. 1 RandomDeconstruction(s)

Randomly choose a variable index i
$s' = s$ (copy the partial solution)
$s'_i = \perp$ (undo one assignment)
Insert (s', s_i) in T
RandomDeconstruction(s')

Alg. 2 SystematicDeconstruction(s)

for all variable indices i **do**
$s' = s$ (copy the partial solution)
$s'_i = \perp$ (undo one assignment)
Insert (s', s_i) in T
SystematicDeconstruction(s')

-reţeaua obţine un scor normalizat pentru fiecare bit din vectorul de ieşire, care poate fi interpretat ca o distribuţie de probabilitate
-bitul cu cel mai mare scor corespunde următoarei alocări sugerate;

Aceste alegeri au trei consecinţe importante:

1. reţeaua este agnostică la structura problemei;
2. reţeaua este pregătită tehnic pentru a imita secvenţe specifice de construcţie a soluţiilor alese în mod arbitrar;
3. presupunând că DNN-ul este folosit pentru a ghida un proces de

căutare, este uşor de luat în considerare propagarea nerespectând scorurile pentru perechile de valori variabile care au fost tăiate. Ca efect advers, renunţăm la posibile avantaje de performanţă care ar putea veni prin includerea informaţiilor despre structura problemei.

2.2 Pregatirea-probleme de referinta

In acest studiu de caz s-au analizat 2 probleme clasice CSP: N-Queens si Partial Latin Square. S-a optat pentru prima problema, o dimensiune n=8 (vector de biti cu n=64), iar pentru cea de-a doua, un n=10 (vector de biti cu n=1.000).

Pentru problema cu 8 regine, am folosit 1/4 din cele 12 soluţii nesimetrice pentru semănarea setului de antrenament, iar cele rămase pentru setul de testare. Atât formarea, cât şi setul de testare sunt obţinute apoi prin generarea tuturor echivalenţilor simetrici şi apoi prin aplicarea deconstrucţiei sistematice la soluţiile rezultate.

Pentru PLS, am utilizat o metodă de generare aleatorie nepărtinitoare pentru a obţine două seturi de date „brute", respectiv conţinând 10.000 şi 20.000 de soluţii.

Fiecare set de date brute este împărţit într-un set de antrenament şi unul de test, care conţine, respectiv, 1/4 şi 3/4 din soluţii. Exemplele reale au fost apoi obţinute prin deconstrucţie aleatorie.

2.3 Pregatirea-retele si date de antrenament

S-au utilizat reţelele neuronale reziduale preactive. S-au instruit reţelele într-un mod supravegheat, folosind 10% din exemple (alese la întâmplare) ca set de validare.În cele din urmă, ne-am stabilit să folosim optimizatorul Adam, cu parametrii $\beta 1 = 0.9$ şi $\beta 2 = 0.99$.

Rata iniţială de învăţare $\alpha 0$, a fost trezită progresiv prin epoci cu degradare proporţională cu epoca de formare t, rezultând o rată de învăţare $\alpha = \alpha 0 /(1 + k \times t)$ cu $k = 10\text{-}3$. Pregătirea a fost oprită după ce nu a fost îmbunătăţită acurateţea de validare pentru epocile e.

In problema N-Queens cu n=8, am folosit un strat iniţial de 200 de neuroni, peste 100 de blocuri reziduale, fiecare compus din două straturi de 500 şi 200 de neuroni şi, în final, un strat de ieşire de 64 de neuroni, pentru un total de peste 200 de straturi. S-a facut o optimizarea lotului, folosind o rată de învăţare iniţială de $\alpha 0 = 0,1$ şi o toleranta de e = 200 epoci, iar dropout a fost aplicat pe fiecare intrare şi neuron ascuns cu probabilitatea p = 0.1.

Pentru problema PLS, am folosit o reţea mai mică din cauza vectorilor de intrare / ieşire mai mari. Am folosit un strat iniţial de 200 de neuroni, apoi 10 blocuri reziduale, fiecare compus din două straturi de 300 şi 200 de neuroni şi un strat final de ieşire de 1000 de neuroni, pentru un total de 22 de straturi. S-a facut o optimizarea a mini-lotului, folosind amestecarea în fiecare epocă, folosind o rată de învăţare iniţială de $\alpha 0 = 0,03$ şi o toleranta de e = 50 epoci, iar abandonarea a fost aplicată neuronului ascuns cu probabilitatea p = 0.1. Mărimea mini lotului a fost stabilită la 50.000 pentru antrenamentul pe setul de date 10k şi la 100.000 pentru setul de date 20k.

3. Experiment

Interese: Un DDN ghiceste atribuirea „corectă” în conformitate cu metoda de deconstrucţie folosită? DNN-urile învaţă să genereze sarcini fezabile, indiferent dacă acestea sunt „corecte” conform seturilor de date?

Presupunând că reţelele învaţă de fapt ceva despre constrângerile problemei, este logic să verificăm dacă anumite tipuri de constrângeri sunt învăţate mai bine decât altele. Utilizarea DNN-urilor pentru a ghida un proces de căutare a arborelor reale duce la reducerea numărului de eşecuri?

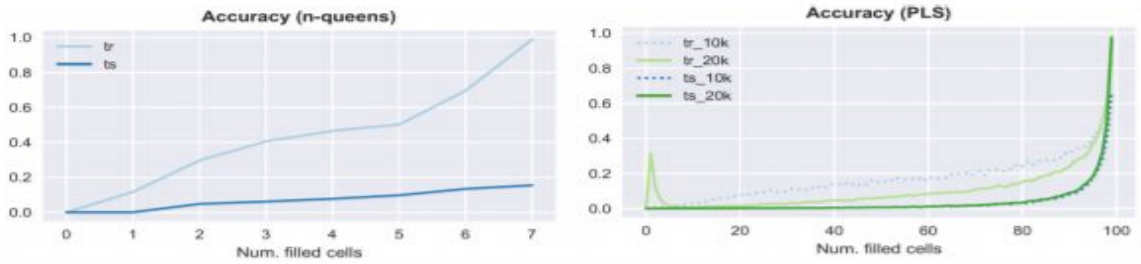

Fig. 1. Accuracy on the training and test sets

3.1 Precizia retelei

1. Precizia este un pic mai mare decât la ghicirea aleatorie, dar totuşi diferenţa este destul de mică, în special pentru PLS; acest lucru sugerează că reţelele nu se descurcă foarte bine la sarcina pentru care sunt instruiţi.

2. În al doilea rând, precizia setului de testare este considerabil mai mică decât a setului de antrenament, dar în acest caz există şi un motiv structural. Tăierea în ultima fază a generarii setului de date introduce un grad de ambiguitate în formarea noastră: deci, pentru aceeaşi atribuţie parţială, setul de antrenament şi setul de teste pot raporta diferite sarcini „corecte”, care nu pot fi prezise în mod corect.

3. În al treilea rând, precizia tinde să crească cu numărul de celule umplute. A avea multe celule umplute înseamnă a avea foarte puţine completări fezabile şi, prin urmare, este mai puţin probabil ca aceeaşi instanţă să apară atât în setul de test cât şi în cel de antrenament. În această situaţie, este intuitiv mai uşor pentru reţea să eticheteze o sarcină specifică drept „corectă”.

A treia observaţie lasă o întrebare deschisă: în timp ce numărul mic de completări fezabile poate explica de ce creşte exactitatea, nu reuşeşte să explice amploarea creşterii. Rezultatul ar fi mult mai uşor de explicat presupunând că DNN a învăţat cumva ceva despre constrângerile problemei.

3.2 Raportul de fezabilitate

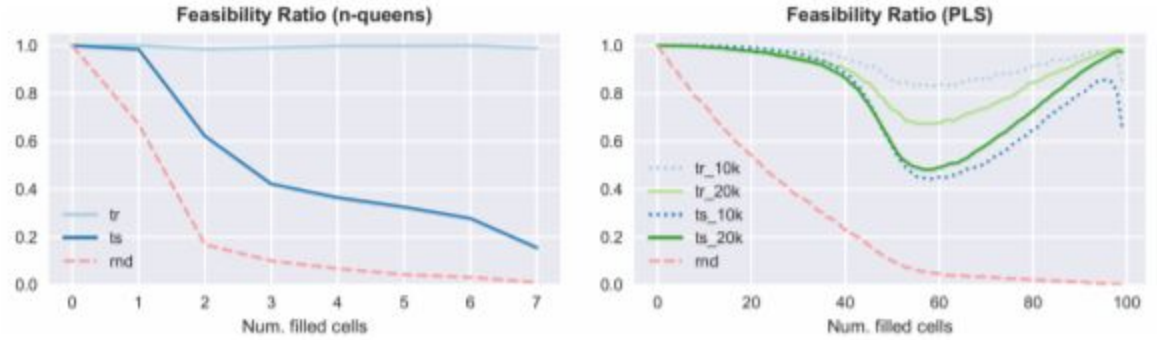

Fig. 2. Feasibility ratios on the training and test sets

1. Există o diferenţă izbitoare între valorile de precizie din figura 1 şi raporturile de fezabilitate.

2. O astfel de discrepanţă se poate datora faptului că cu cât o soluţie parţială este goală, cu atât sunt mai multe sarcini fezabile care pot fi găsite chiar şi prin ghicire. Cu toate acestea, raportul de fezabilitate raportat este, de asemenea, semnificativ mai mare decât valoarea de bază aleatorie. Acest lucru este greu de explicat, cu excepţia cazului în care presupunem că DNN-urile au învăţat cumva semantica constrângerilor problemei.

3. Raporturile de fezabilitate pentru reţelele PLS au o înclinare între 50 şi 60 de variabile prealocate şi apoi tind să crească din nou. Acesta este exact comportamentul la care se poate aştepta mulţumirea propagării constrângerilor: atunci când multe variabile sunt legate, multe valori sunt tăiate şi numărul de sarcini disponibile este redus. Cu toate acestea, DNN-urile din această etapă nu se bazează deloc pe propagare. Chiar şi precizia mai mare din figura 1 nu este suficientă pentru a justifica cât de mult tendinţele de fezabilitate tind să crească pentru soluţii aproape complete. Presupunând că DNN-ul a aflat constrângerile problemei poate explica creşterea.

3.3 Preferinta constrangerilor

În continuare, am conceput un experiment pentru a investiga dacă unele constrângeri sunt gestionate mai bine decât altele. Având în vedere o soluţie parţială, folosim DNN pentru a obţine o distribuţie a probabilităţii pe toată sarcina posibilă, una dintre ele fiind aleasă şi executată la întâmplare. Procesul se repetă de câte ori există variabile şi se bazează pe reprezentarea noastră pe un nivel scăzut, pe vectorul de biţi, al soluţiei parţiale. În consecinţă, la sfârşitul procesului pot exista variabile care au fost „alocate de mai multe ori", şi, de asemenea, variabile neasignate. Am utilizat această abordare pentru a genera 10.000 de soluţii pentru fiecare DNN, iar pentru comparaţie am făcut acelaşi lucru folosind o distribuţie uniformă.

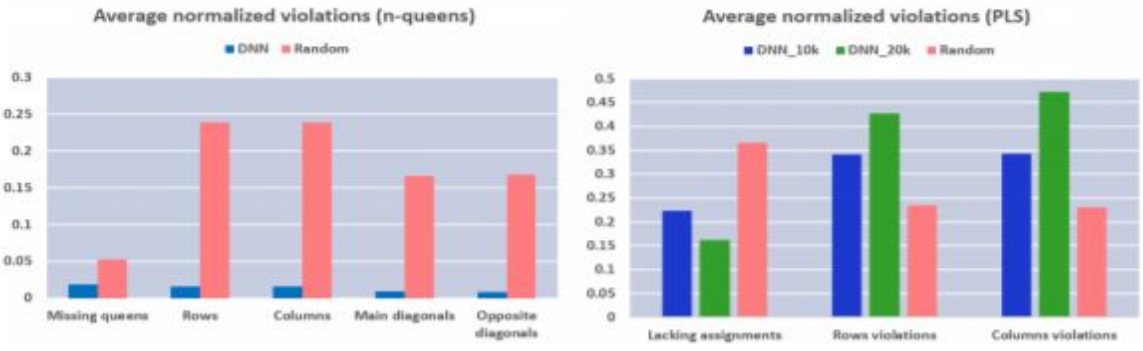

Fig. 3. Average violations on the two problems

3.4 Căutare în arborele de ghidare

Se utilizează un model CP clasic pentru PLS (o variabilă de domeniu finit pe celulă) şi folosim propagatorul GAC AllDifferent pentru constrângerea rândului şi coloanei. Se compară rezultatele celor două DNN-uri cu cele ale euristicilor care aleg uniform la întâmplare atât variabila cât şi valoarea ce trebuie atribuită;

S-a optat pentru o implementare simplă (dar ineficientă) bazându-ne pe

API-ul Google sau tool-uri din Python: din acest motiv ne concentrăm evaluarea pe numărul de eşecuri;

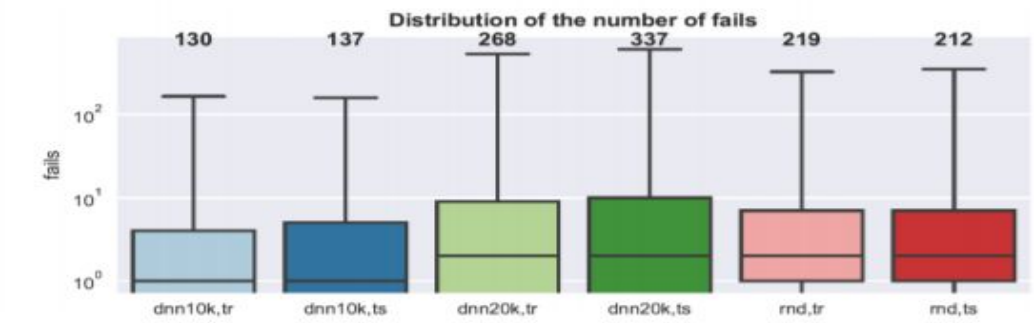

Fig. 4. Distribution of the number of fails on the train and test sets. At the top of each box we report the number of times the fail cap was reached.

4. Concluzii

Am efectuat o investigaţie prelimiară pentru a înţelege dacă DNN-urile pot învăţa cum să rezolve problemele combinatorii.

Am adoptat o configuraţie generală, total agnostică cu structura problemei şi am instruit reţelele pe secvenţe de construcţie ale soluţiei alese în mod arbitrar.

Deşi are o precizie scăzută în timpul antrenamentului, un DNN poate deveni capabil să genereze atribuţii de valoare variabilă consistente la nivel global.

Reţelele nu imită doar secvenţele de atribuire în setul de antrenament, dar este compatibil cu ipoteza că DNN-urile au aflat ceva despre structura problemei.

Reţelele nu par să favorizeze nicio restricţie abstractă în special, ceea ce sugerează că ceea ce învaţă nu se potriveşte cu înţelegerea noastră obişnuită a CSP-urilor.

Când sunt utilizate pentru ghidarea unui proces de căutare, DNN-urile noastre au furnizat rezultate mixte, subliniind că realizarea îmbunătăţirii

performanţei poate necesita o analiza mai aprofundată.

Această linie de cercetare se află încă într-un stadiu incipient, iar metoda din acest studiu este în prezent limitată la probleme de dimensiuni fixe.

Bibliografie:

https://www.researchgate.net/profile/Michela_Milano/publication/325622479_Model_Agnostic_Solution_of_CSPs_via_Deep_Learning_A_Preliminary_Study/links/5c0fcc1e92851c39ebe5395d/Model-Agnostic-Solution-of-CSPs-via-Deep-Learning-A-Preliminary-Study.pdf