**N H**



**Definition:**

**H** denote a Hidato puzzle

**N** denote a Numbrix puzzle

***C*(N)** set containing all the cells of N

***C*(H)** set containing all the cells of H

**Ci- a cell**

***N*(*ci*)** neighborhood *of a cell ci*  in *C*(H)

**successor** *of cell a ci is a neighboring cell containing a consecutive number*

**Solutions**

**Condition 1:** Every cell of the grid must be filled with a natural number ranging from 1 to *n*;

**Condition 2:** No cell has two numbers;

**Condition 3:** No two cells have the same number;

**Condition 4:** Every cell except the one with number *n* does have a successor;

**Condition 5:** Pre-filled numbers are unchanged.

**Variables**

Let H be a Hidato puzzle with *C*(H) = {*c*1*, c*2*, . . . , cn*}.

Let *UE* denote Unary encoding,specifically the final CNF formula. We use variable *vci,x*, where *ci*  *C*(H)

and *x*  {1*,* 2*, . . . , n*}, to represent the fact that the *i*-th cell contains number *x*.

Consequently, literal ¬*vci,x* indicates the opposite. By this definition we need exactly *n* variables, {*vci,*1*, vci,*2*, . . . , vci,n*}, for each cell *ci*. Therefore, the set of variables for Unary encoding is defined as

*var*(*UE*) = {*vc*1*,*1*, vci,*2*, . . . , vc*1*,n, vc*2*,*1*, vc*2*,*2 *. . . , vc*2*,n, . . . , vcn,n*}.

there is variables.

**Clauses**

consists of five sets of clauses denoted *C*1, *C*2, *C*3, *C*4 and *C*5. Set *C*1 forces the

Unary encoding *is formula UE* = *C*1*C*2*C*3*C*4*C*5 *with*

*C*1 ={*C*1*ci* | *ci*  *C*(H)}*,*

*where C*1*ci* = {*vci,*1*, vci,*2 *. . . , vci,n*}*;*

*C*2 ={*C*2*ci,x*1*,x*2 | *ci*  *C*(H)*, x*1*, x*2 {1*,* 2*, . . . , n*}*, x*1 ≠ *x*2}*,*

*where C*2*ci,x*1*,x*2 = {¬*vci,x*1 *,*¬*vci,x*2}*;*

*C*3 ={*C*3*ci,cj,x* | *ci, cj*  *C*(H)*, ci* 6= *cj , x*  {1*,* 2*, . . . , n*}}*,*

*where C*3*ci,cj,x* = {¬*vci,x,* ¬*vcj,x*}*;*

*C*4 ={*C*4*ci,x* | *ci*  *C*(H)*, x*  {1*,* 2*, . . . , n* − 1}}*,*

*where C*4*ci,x* = {¬*vci,x*} [ {*vcj,x*+1 | *cj*  *N*(*ci*)}*;*

*C*5 ={*C*5*ci,x* | (*ci, x*) *Const*(H)}*,*

*where C*5*ci,x* = {*vci,x*}*.*

*number of clauses:*

|*C*1*UE*| = *n (n variable in each clause )*

|*C*2*UE*| = *(2 variable in each clause )*

|*C*3*UE*| = *(2 variable in each clause )*

|*C*4*UE*| = *n*(*n* − 1) (depend in the garph)

|*C*5*UE*| = |*Const*(H) *(1 variable in each clause )*

**יצירת גרף**

בהינתן n להיות אורך המסלול וq להיות ההסתברות של צלע להיות קיימת בגרף.

נגריל פרמוטציה של 1 עד n כאשר התוצאה שנקבל יהיה הבסיס לגרף שנבנה.

קודקודי הגרף- n קדקודים כאשר כל קודוקוד מייצג את התא במסלול חידאותו.

צלעות הגרף- מסמנים האם תא x y שכנים במפת החידאתו.

בניית צלעות הגרף- בהתחלה נבנה מסלול לפי הפרומטציה שנקבל כך נבטיח לנו פתרון אחד לפחות. אחר כך נגריל צלעות בגרף לפי q שנקבל בקלט..

נשים לב שאין צלעות מתא לעצמו לכן האלכסון הראשי של הגרף יהיה רק אפסים.

**דוגמא**: בהינתן n=5 q=0.2,

[3, 4, 1, 5, 2] הפרמוטציה שהגרלנו

[[0, 0, 0, 1, 1], [0, 0, 0, 0, 1], [0, 1, 0, 1, 1],

[1, 0, 1, 0, 1], [0, 1, 1, 0, 0]]

נשים לב שככל שהp קטן יותר אז הסיכוי לקבל תוצאה זהה לפרמוטציה נמוך.

**המרת הגרף לבעיית החידאתו-**

נשתמש ברדוקציה שהגדרנו קודם ונקבל בעיית CNF

דוגמא: לפי הדוגמא של סעיף קודם נקבל:

C1=(C11 | C12 | C13 | C14 | C15) ^ (C21 | C22 | C23 | C24 | C25) ^ (C31 | C32 | C33 | C34 | C35) ^ (C41 | C42 | C43 | C44 | C45) ^ (C51 | C52 | C53 | C54 | C55)

C2=^ (¬C11 | ¬C12) ^ (¬C11 | ¬C13) ^ (¬C11 | ¬C14) ^ (¬C11 | ¬C15) ^ (¬C12 | ¬C13) ^ (¬C12 | ¬C14) ^ (¬C12 | ¬C15) ^ (¬C13 | ¬C14) ^ (¬C13 | ¬C15) ^ (¬C14 | ¬C15) ^ (¬C21 | ¬C22) ^ (¬C21 | ¬C23) ^ (¬C21 | ¬C24) ^ (¬C21 | ¬C25) ^ (¬C22 | ¬C23) ^ (¬C22 | ¬C24) ^ (¬C22 | ¬C25) ^ (¬C23 | ¬C24) ^ (¬C23 | ¬C25) ^ (¬C24 | ¬C25) ^ (¬C31 | ¬C32) ^ (¬C31 | ¬C33) ^ (¬C31 | ¬C34) ^ (¬C31 | ¬C35) ^ (¬C32 | ¬C33) ^ (¬C32 | ¬C34) ^ (¬C32 | ¬C35) ^ (¬C33 | ¬C34) ^ (¬C33 | ¬C35) ^ (¬C34 | ¬C35) ^ (¬C41 | ¬C42) ^ (¬C41 | ¬C43) ^ (¬C41 | ¬C44) ^ (¬C41 | ¬C45) ^ (¬C42 | ¬C43) ^ (¬C42 | ¬C44) ^ (¬C42 | ¬C45) ^ (¬C43 | ¬C44) ^ (¬C43 | ¬C45) ^ (¬C44 | ¬C45) ^ (¬C51 | ¬C52) ^ (¬C51 | ¬C53) ^ (¬C51 | ¬C54) ^ (¬C51 | ¬C55) ^ (¬C52 | ¬C53) ^ (¬C52 | ¬C54) ^ (¬C52 | ¬C55) ^ (¬C53 | ¬C54) ^ (¬C53 | ¬C55) ^ (¬C54 | ¬C55) ^ (¬C11 | ¬C21) ^ (¬C12 | ¬C22) ^ (¬C13 | ¬C23) ^ (¬C14 | ¬C24) ^ (¬C15 | ¬C25)

C3= ^ (¬C11 | ¬C31) ^ (¬C12 | ¬C32) ^ (¬C13 | ¬C33) ^ (¬C14 | ¬C34) ^ (¬C15 | ¬C35) ^ (¬C11 | ¬C41) ^ (¬C12 | ¬C42) ^ (¬C13 | ¬C43) ^ (¬C14 | ¬C44) ^ (¬C15 | ¬C45) ^ (¬C11 | ¬C51) ^ (¬C12 | ¬C52) ^ (¬C13 | ¬C53) ^ (¬C14 | ¬C54) ^ (¬C15 | ¬C55) ^ (¬C21 | ¬C31) ^ (¬C22 | ¬C32) ^ (¬C23 | ¬C33) ^ (¬C24 | ¬C34) ^ (¬C25 | ¬C35) ^ (¬C21 | ¬C41) ^ (¬C22 | ¬C42) ^ (¬C23 | ¬C43) ^ (¬C24 | ¬C44) ^ (¬C25 | ¬C45) ^ (¬C21 | ¬C51) ^ (¬C22 | ¬C52) ^ (¬C23 | ¬C53) ^ (¬C24 | ¬C54) ^ (¬C25 | ¬C55) ^ (¬C31 | ¬C41) ^ (¬C32 | ¬C42) ^ (¬C33 | ¬C43) ^ (¬C34 | ¬C44) ^ (¬C35 | ¬C45) ^ (¬C31 | ¬C51) ^ (¬C32 | ¬C52) ^ (¬C33 | ¬C53) ^ (¬C34 | ¬C54) ^ (¬C35 | ¬C55) ^ (¬C41 | ¬C51) ^ (¬C42 | ¬C52) ^ (¬C43 | ¬C53) ^ (¬C44 | ¬C54) ^ (¬C45 | ¬C55)

C4= ^ (¬C11 | C42 | C52) ^ (¬C12 | C43 | C53) ^ (¬C13 | C44 | C54) ^ (¬C14 | C45 | C55) ^ (¬C21 | C52) ^ (¬C22 | C53) ^ (¬C23 | C54) ^ (¬C24 | C55) ^ (¬C31 | C22 | C42 | C52) ^ (¬C32 | C23 | C43 | C53) ^ (¬C33 | C24 | C44 | C54) ^ (¬C34 | C25 | C45 | C55) ^ (¬C41 | C12 | C32 | C52) ^ (¬C42 | C13 | C33 | C53) ^ (¬C43 | C14 | C34 | C54) ^ (¬C44 | C15 | C35 | C55) ^ (¬C51 | C22 | C32) ^ (¬C52 | C23 | C33) ^ (¬C53 | C24 | C34) ^ (¬C54 | C25 | C35)

C5=^ (C31) ^ (C25)

**פתירת הcnf –**

נשתמש באלגוריתם DPLL

DPLL-

זוהי מימוש לאלגוריתם Backtracking אלגוריתם כזה מנסה לבנות באופן הדרגתי פתרון לבעיה כלשהי, תוך הסתמכות על כך שהוא יכול לפעמים לזהות "באמצע" הבניה שמשהו התקלקל ואז להתחיל לחזור לאחור ולתקן את עצמו מבלי לבזבז עוד זמן על המשך הבניה המקולקלת עד הסוף. בשל כך, אלגוריתמי Backtracking הם יעילים יותר מאשר "סתם" חיפוש ממצה שעובר על כל ההשמות האפשריות.

**ניתוח פתרון האלגוריתם**

ניתוח 1- האם יש קשר בין q ההסתברות להגריל צלע בגרף לבין התוצאה שנקבל במקרה שלנו האם התוצאה שנקבל היא הפרמוטציה שהגרלנו בהתחלה שהוא הבסיס לגרף שבנינו בהתחלה

ניתוח 2- כמו אחד רק בלי C5 כלומר האילוץ שתא הראשון והאחרון יהיו חייבים להיות בתוצאה הסופית

N=5

%1

with probability 0.9 we get the initial path 25.4%

with probability 0.7 we get the initial path 50.9%

with probability 0.6 we get the initial path 66.4%

with probability 0.4 we get the initial path 86.8%

with probability 0.2 we get the initial path 97.2%

with probability 0.1 we get the initial path 98.1%

with probability 0.015 we get the initial path 99.8%

with probability 0.01 we get the initial path 99.7%

%2

with probability 0.9 we get the initial path 0.8999999999999999%

with probability 0.7 we get the initial path 2.9000000000000004%

with probability 0.6 we get the initial path 5.4%

with probability 0.4 we get the initial path 17.5%

with probability 0.2 we get the initial path 39.6%

with probability 0.1 we get the initial path 60.3%

with probability 0.015 we get the initial path 80.4%

with probability 0.01 we get the initial path 81.5%

n=8

%1

with probability 0.9 we get the initial path 0.2%

with probability 0.7 we get the initial path 1.0%

with probability 0.6 we get the initial path 5.3%

with probability 0.4 we get the initial path 28.1%

with probability 0.2 we get the initial path 85.5%

with probability 0.1 we get the initial path 97.3%

with probability 0.015 we get the initial path 99.4%

with probability 0.01 we get the initial path 99.5%

%2

with probability 0.9 we get the initial path 0.0%

with probability 0.7 we get the initial path 0.3%

with probability 0.6 we get the initial path 0.1%

with probability 0.4 we get the initial path 1.2%

with probability 0.2 we get the initial path 16.400000000000002%

with probability 0.1 we get the initial path 50.6%

with probability 0.015 we get the initial path 83.5%

with probability 0.01 we get the initial path 86.4%