THE COMPLEXITIES OF PUZZLES, CROSS SUM AND THEIR ANOTHER SOLUTION PROBLEMS(ASP)

パズルとカックロ、そして、その別解問題の計算量

by

SETA Takahiro

瀬田 剛広

A Senior Thesis

卒業論文

Submitted to

the Department of Information Science

the Faculty of Science, the University of Tokyo

on February 5, 2002

in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Science

Thesis Supervisor: IMAI Hiroshi　今井 浩
Professor of Information Science

ABSTRACT

It is expected that the fun of popular puzzles like crossword has relation with the complexity and many of those puzzles are NP-complete. In this thesis, first, to support the conjecture, the NP-completeness of CROSS SUM was proved by showing poly-time reductions from PARTITION INTO CIRCUITS of max 3 neighbor directed graph and from 3SAT or ONE-IN-THREE 3SAT. Second, the NP-completeness of ANOTHER HAMILTONIAN CIRCUIT restricted to planar and max 3-degree graph was proved by showing a poly-time parsimonious reduction, which seems to be related to puzzles most in ANOTHER SOLUTION PROBLEMS (ASP), or problems to decide if there is some solution other than a given solution, which are important for designing puzzle problems especially. By the known results, the NP-completeness of ASP-SLITHER LINK was also proved. The NP-completeness of ASP of CROSS SUM is also proved by showing the NP-completeness of ONE-IN-THREE 3SAT and pointing that the reductions from ONE-IN-THREE 3SAT to CROSS SUM are parsimosious. Third, a classification of puzzles from the viewpoint of automatical problem design or of design support is suggested.

論文要旨

クロスワードなどに代表される、多くの人に親しまれているパズルの面白さは、その計算量と関係があり、多くのパズルは NP 完全であると予想されている。そこで、それを裏付けるべく、カックロ (クロスサム) を例にとり、隣接点数 3 以下の PARTITION INTO CIRCUITS および 3SAT, ONE-IN-THREE 3SAT からの多項式時間還元を示すことにより、その NP 完全性を示した。また、パズルの作成において重要となる別解問題、つまり、ひとつ解が与えられたとき他に解があるかどうか判定する問題、の中でもパズルとの関係が深いと思われる、次数 3 以下の平面グラフに制限した HAMILTONIAN CIRCUIT の別解問題について、3SAT からの多項式時間 1 対 1 還元 (poly-time parsimonious reduction) を示すことにより、その NP 完全性を示した。知られた結果により、スリザーリンクの別解問題の NP 完全性も示されたことになる。カックロの別解問題については、ONE-IN-THREE 3SAT の別解問題の NP 完全性を証明し、還元が 1 対 1 還元になっている事を指摘することにより、その NP 完全性が証明された。さらに、パズルの自動作成、作成支援に焦点をあてた分類法を提案した。

Acknowledgements

I would like to thank Mr. Yato, the author of "On the NP-completeness of Slither Link Puzzle", for motivating me to study such a interesting problem.

I would also thank my supervisor Prof. Imai, Mr. Ito, Mr. Sasaki and other members in Imai laboratory for their helpful advices.

Contents

Chapter 1

Introduction

1.1 Backgrounds

There are a variety of puzzles, CROSS WORD puzzle and other CROSS WORD type puzzles, JIGSAW puzzles and other combination type puzzles, NONOGRAM and other picture drawing puzzles, LOGIC puzzles and so on. There are many people who enjoy puzzles. Children, the elder or even people who hesitate mathematics can enjoy puzzles and do enjoy puzzles. There is also a long history on puzzles. I hear the ancient China has puzzles. Anyway puzzle are interesting.[15, 14]

From the viewpoint of complexity, it is expected that there are some relations between the intersst of puzzles and the complexity of puzzles. In other word, it is expected that many puzzles are NP-complete or have higher complexity. In fact, many puzzles are proved to have NP-completeness or higher complexity as shown in Table.1.1, and they are also listed in books [7].

puzzle name	complexity	reference
CROSS WORD CONSTRUCTION	NP-complete	[7]
NONOGRAM	NP-complete	[12]
SLITHER LINK	NP-complete	[13]
CRYPTARITHMS	NP-complete	[6]
CLICKOMANIA (SAMEGAME)	NP-complete	[2]
15-PUZZLE	NP-complete	[10]
SOKOBAN	PSPACE-complete	[3, 4]

Table 1.1: This table shows the complexities of various puzzles.

There are also many games whose complexities are studied as shown in Table.1.2.

game name	complexity	reference
CHESS	EXPTIME-complete	[5]
GO with ko	EXPTIME-complete	[5]
GO without ko	PSPACE-hard	[5]
Othello	PSPACE-complete	[5]

Table 1.2: This table shows the complexities of various games.

David Eppstein says

> Many of the games and puzzles people play are interesting because of their difficulty: it requires cleverness to solve them. Often this difficulty can be shown mathematically, in the form of computational intractibility results: every NP-complete problem is in some sense a puzzle, and conversely many puzzles are NP-complete. Two-player games often have higher complexities such as being PSPACE-complete.

in his home page.[5]

So, in this thesis, we will see the complexity of a pencil puzzle CROSS SUM in order to support the idea that there are some relations between the interest of puzzles and the complexity of puzzles.

1.2 Construction

First, in Chapter.2, we will prove the NP-completeness of CROSS SUM and consider where the complexity come from.

Second, in Chapter.3, we will consider a problem group called Another Solution Problem (ASP), see the importance of it on making puzzle problems. and prove NP-completeness of some ASPs.

Last, in Chapter.4,we will consider automatical designing of puzzle problem and support of designing puzzle problem.

Chapter 2

Complexities of Puzzles

As mentioned in Chapter 1, it is expected that many puzzles are NP-complete or have higher complexity. So, in this chapter, we will consider the complexity of CROSS SUM as an example.

2.1 The NP-completeness of CROSS SUM

CROSS SUM is a pencil puzzle famous in all over the world and known as KAKKURO in Japanese. It's a puzzle looks like CROSS WORD, but not like CROSS WORD, it doesn't need any knowledge or any dictionaries, people just can add numbers can enjoy this puzzle. So, in some sence, CROSS SUM is easy to consider with computers. And, on CROSS SUM, like most of puzzles, the correctness of an answer is easy to confirm. It's an evidence that CROSS SUM is in NP.

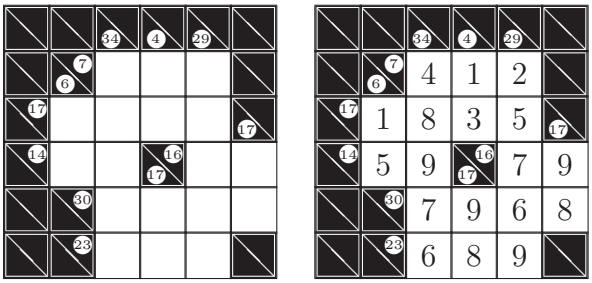


Figure 2.1: This is an example of CROSS SUM and the answer frow WebNikoli [15].

2.1.1 Definitions

To consider CROSS SUM, we define CROSS SUM as follows. (See WebNikoli[15] for original rules.)

Definition CROSS SUM

Input A $W \times H$ square board which satisfies following properties

- The board is made of white boxes and black boxes.
- Numbers called sum are given one for each vertical or horizontal line of white boxes. (And the numbers are put in the black boxes on the beginning of the white line.)
- White boxes are connected, not separated by black boxes. (Not an impotant property: If not connected it can be separated to two puzzles.)
- Black boxes can be located next to each other not like CROSS WORD.
- Any line of white boxes has at least two white boxes.

Question Does the given instance have any settings of numbers in white boxes which satisfies following properties?

- Numbers in white boxes are from 1 to 9.
- The sum of numbers in one line is same to the given sum.
- Any number appears at most once in one white line.

The settings are called answers or solutions.

and PARTITION INTO CIRCUITS

Definition PARTITION INTO CIRCUITS

Input A graph.

Question Are there any set of vertex disjoint circuits which covers all vertices?

And, we also define a word *neighbor*, which is used with directed graphs, as follows

Definition Neighbor

Neighbor of a vertex v means the set of vertices next to v. i.e.

$$Neighbor(v) = \{w | (v, w) \in E \vee (w, v) \in E\}$$

If we allow both undirected edges and directed edges, the number of neighbors will be the number of edges connecting to the vertex.

2.1.2 Preliminaries

What we must need for considering the complexity of CROSS SUM is:

- 3SAT is NP-complete.[7]

Some more knowledge may be helpful:

- ONE-IN-THREE 3SAT is NP-complete [11, 7]. We will define ONE-IN-THREE 3SAT and prove this theorem in Chapter.3.
- PARTITION INTO CIRCUITS is NP-complete [7]. We will also prove this theorem in Chapter.3.

And reading some proofs on the NP-completeness of Hamiltonian Circuit [8, 9] may also be helpful.

2.1.3 Proof of the NP-completeness

The NP-completeness of restricted PARTITION INTO CIRCUITD

First, we prove the NP-completeness of PARTITION INTO CIRCUITS restricted to max 3 neighbor planar directed graph, showing a poly-time reduction from 3SAT as it is done in [8]. We can also prove it by showing another poly-time reduction from a problem called ONE-IN-THREE 3SAT. (It's obvious that PARTITION INTO CIRCUITS is in NP. We can easily confirm the correctness of a solution, by seeing the circuits pass all the vertices.)

To show the reduction, we need to make some gadgets for OR, XOR, 3OR and so on.

First, we start by seeing some useful small parts. Two are shown in Figure.2.2. The first one makes edges which must be selected. This is very useful when we need to avoid undesired possible local settings of some gadget. And, this type of parts is also used in previous works [8, 9]. The second one works like XOR or NOT, since just one of the two outer edges is selected.

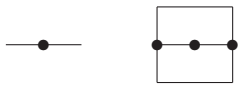


Figure 2.2: These figures show some useful small parts to make gadgets for reduction. The first one can be used to make an edge which must be selected. The second one is a kind of XOR, just one of the two outer edges is selected.

Next, we will make gadgets for OR and XOR. But, a previous work says PARTITION INTO CIRCUITS (PARTITION INTO HAMILTONIAN SUBGRAPHS) is solvable in P if the graph is undirected [7]. So, we start from making OR and XOR for directed edges. The constructions are shown in Figure.2.3 and Figure.2.4.

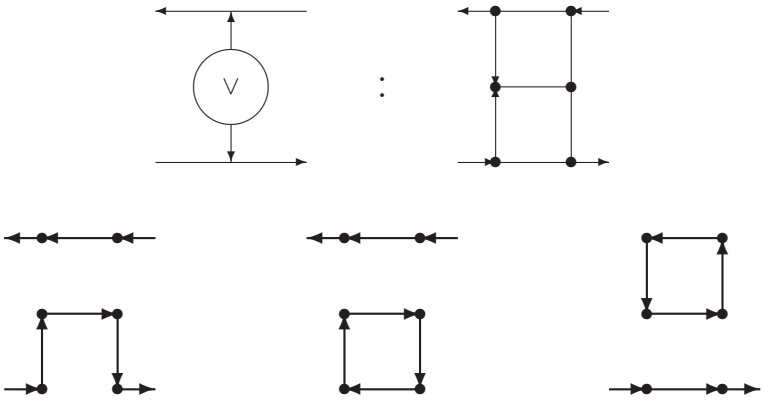


Figure 2.3: These figures show how to make the gadget for directed-OR, and the possible local settings of the gadget.

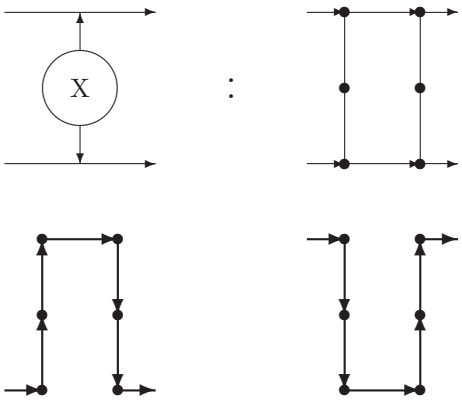


Figure 2.4: These figures show how to make the gadget for directed-XOR, and the possible local settings of the gadget.

If we need to change the directions of some edges of OR or XOR, we can do it by making a loop as shown in Figure.2.5. But, we don't use these non-planar gadgets since we will get planar ones soon.

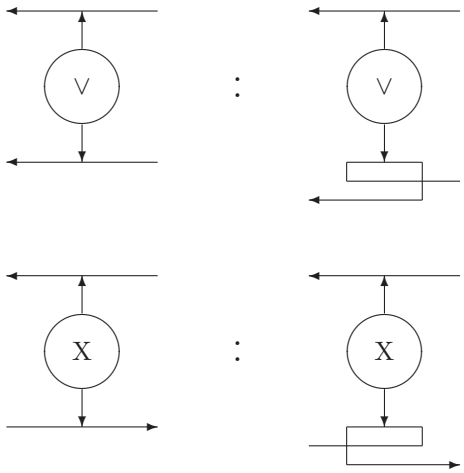


Figure 2.5: These figures show how to chande the directions of edges in the gadgets.

Now, we will start to make gadgets for OR and XOR for undirected edges. Once undirected one is made, then we can make directed one by adding directions to edges. But, it is a little difficult to make undirected gadgets available in all cases. So, we will see the gadgets only for special cases first (Figure.2.6,Figure.2.7). Note that we are considering PARTITION INTO CIRCUITS, and we need not to connect the start of a edge and the end of the edge in the first possible setting of the OR gadget.

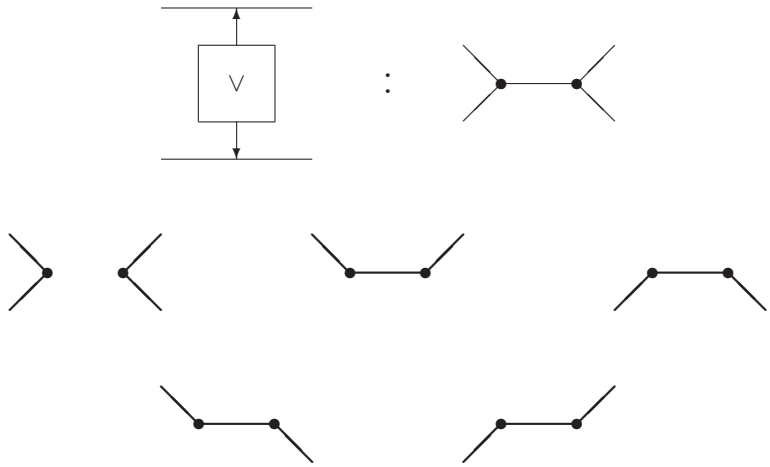


Figure 2.6: These figures show how to make the gadget for undirected-OR only available in special cases, and the possible local settings of the gadget. The 2 settings shown in the bottom are not desirable.

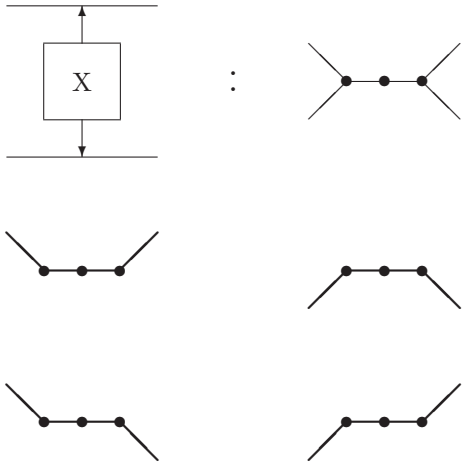


Figure 2.7: These figures show how to make the gadget for undirected-XOR only available in special cases, and the possible local settings of the gadget. The 2 settings shown in the bottom are not desirable.

Now, we can make the gadgets for OR and XOR for undirected edges. The constructions are shown in Figure.2.8, Figure.2.9. Note that these gadgets are made of planar graph, and we can make any planar directed OR and XOR by adding directions to the edges out from these gadget.

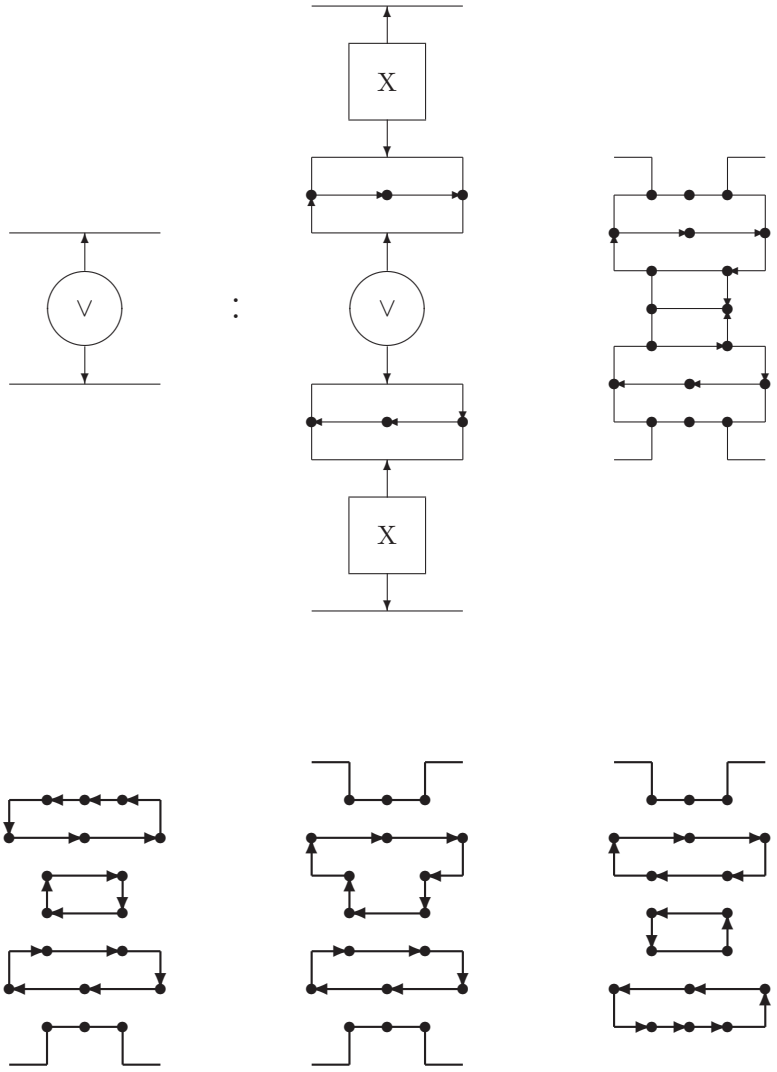


Figure 2.8: These figures show how to make the gadget for undirected-OR available in all cases, and the possible local settings of the gadget. The figure in the middle of the top shows the idea to make this gadget and the right shows the full construction of it. Make sure that this case is the "special case" of the previous undirected-XOR and all of the local settings are shown.

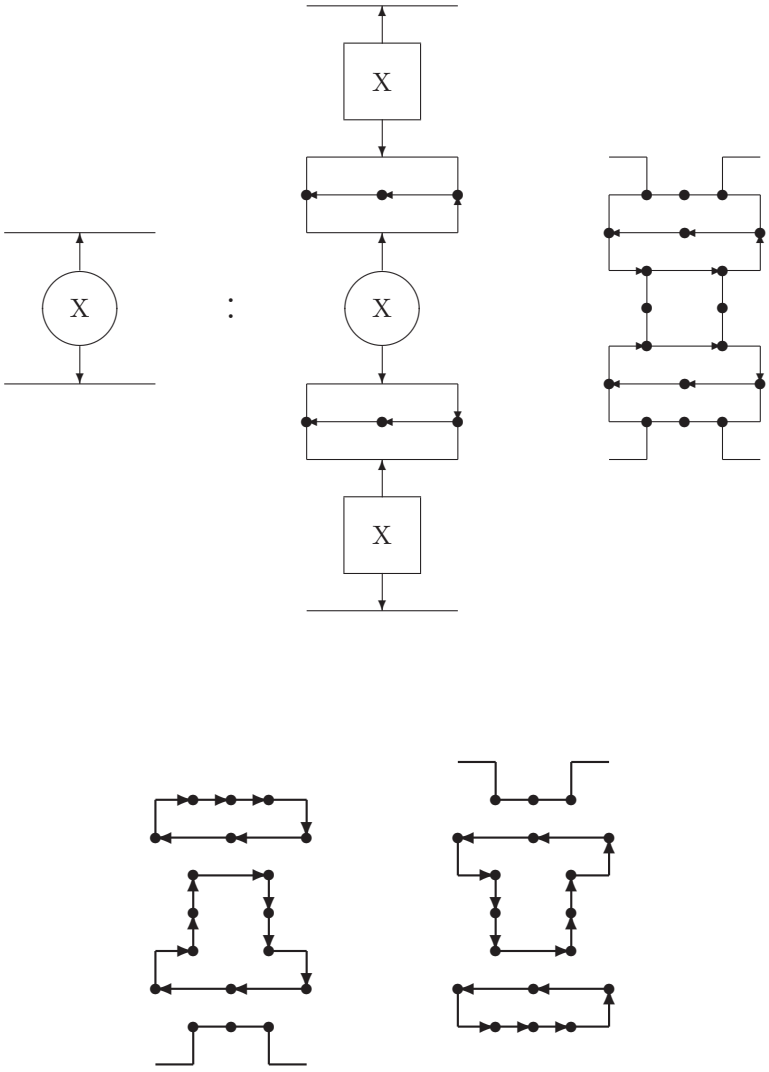


Figure 2.9: These figures show how to make the gadget for undirected-XOR available in all cases, and the possible local settings of the gadget. The figure in the middle of the top shows the idea to make this gadget, and the right shows the full construction of it. Make sure that this case is the "special case" of the previous undirected-XOR and all of the local settings are shown.

Then, we will make the gadgets for 3OR and Select (One-in-three). The constructions are shown in Figure.2.10, Figure.2.11.

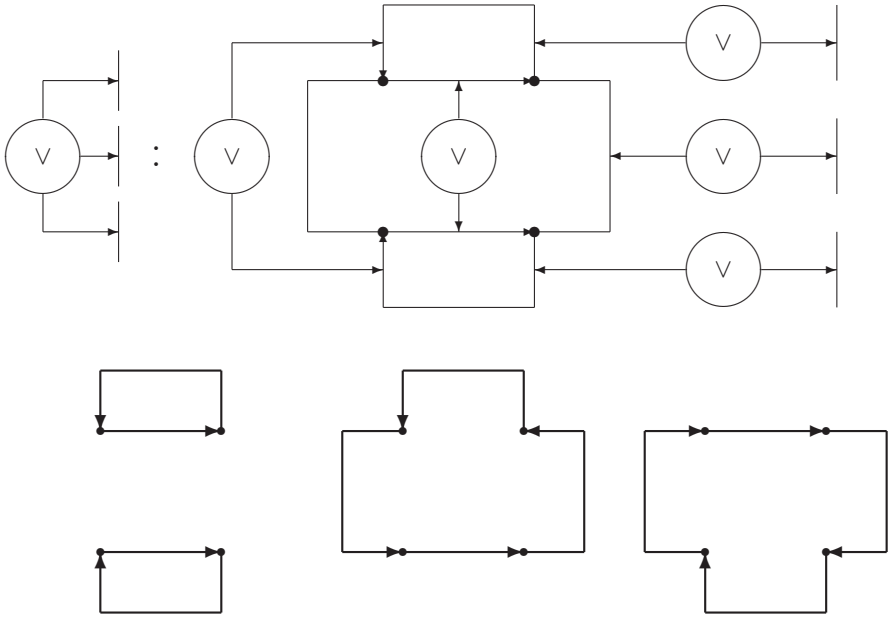


Figure 2.10: These figures show how to make the gadget for 3OR, and the possible settings of the middle part of it.

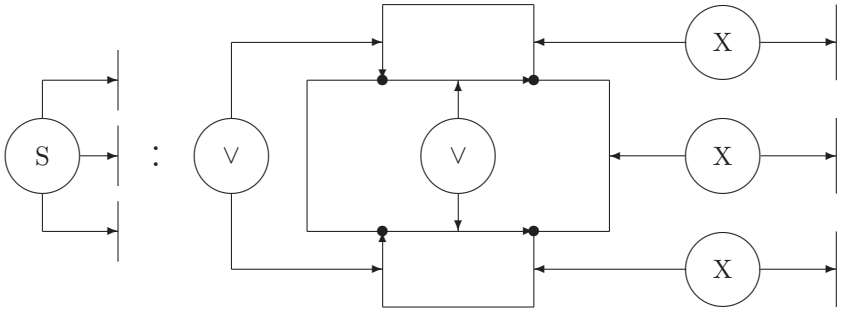


Figure 2.11: These figures show how to make the gadget for Select (One-in-three).

Last, we will make the gadgets for cross of ORs and XORs as sown in Figure.2.12. With these gadgets, we can cross 3OR and Select gadgets, too.

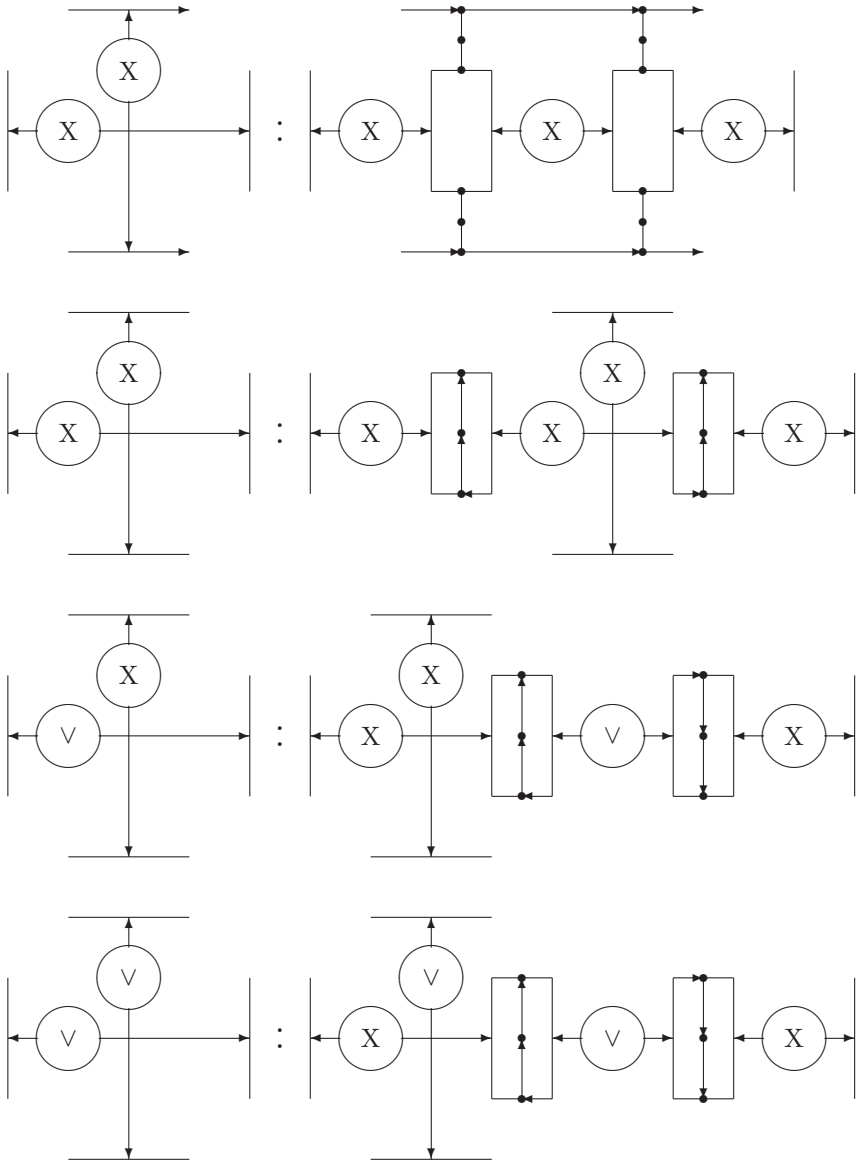


Figure 2.12: These figures show how to cross XORs and ORs in planar graph.

Now, we can prove the NP-completeness of PARTITION INTO CIRCUITS restricted to max 3 neighbor planar directed graph by showing poly-time reductions from 3SAT or ONE-IN-THREE 3SAT. We can do the reduction using 3OR gadgets or Select gadgets as shown in Figure.2.13. And, we can put the reduced graph onto a grid easily. (Since the gadgets are constructed on grids and the reduced graph is on a grid, and we can see a method putting graph on grid in [1].)

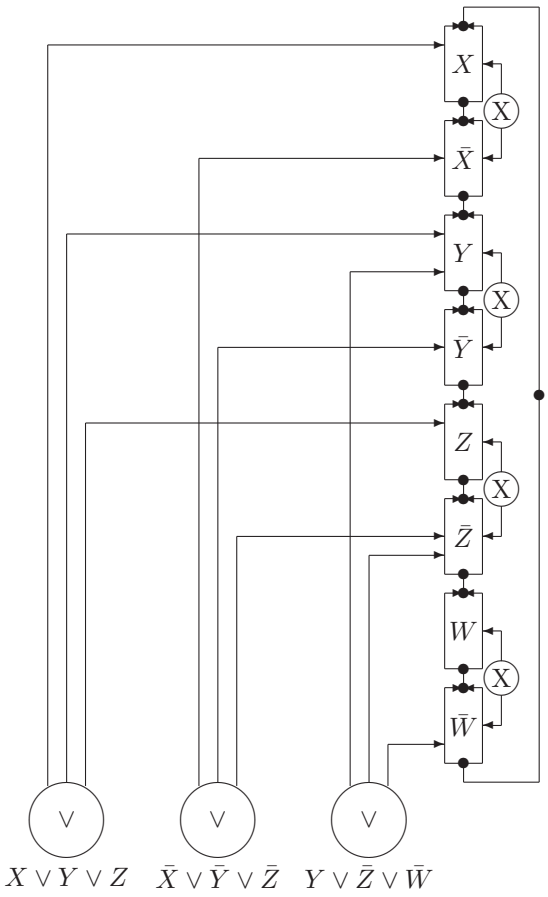


Figure 2.13: This figure shows the reduction from 3SAT to PARTITION INTO CIRCUITS restricted to max 3 neighbor directed graph. By changing 3OR to Select, this figure will be the reduction from ONE-IN-THREE 3SAT.

The NP-completeness of CROSS SUM

In the previous section, we proved the NP-completeness of PARTITION INTO CIRCUITS restricted to max 3 neighbor directed graph. So, in this section, we prove the NP-completeness of CROSS SUM by showing a poly-time reduction from the restricted PARTITION INTO CIRCUITS on grids. (It's obvious that CROSS SUM is in NP. We can confirm the correectness of a solution by comparing sums of numbers in each line to the given sums, and seeing that any two numbers in one line are different.) To prove the NP completeness, we need some gadgets named $\dot{\mathrm{T}}$ for vertices with 3 neighbors, $\dot{\mathrm{I}}$ and $\dot{\mathrm{L}}$ for vertices with 2 neighbors, I and L for grid points in middle of edge, $\rightarrow$ for directed edges, $\leftrightarrow$ for undirected edge, and if the original graph is not planar, X for cross of two edges. (Figure.2.14,2.15)

The all gadgets are shown in the following pages.

In those gadgets some patterns are often used.

- $6 = 1 + 2 + 3$. This is used for select (1 in 3) or vertex with 3 neigbors.
- $3 = 1 + 2$, $4 = 1 + 3$. The box at cross of them must be set to 1.
- These patterns of number k can be replaced to number $10 - k$, like $24 = 9+8+7$, $17 = 9 + 8$, $16 = 9 + 7$.

The relation between CROSS SUM gadgets to PARTITION INTO CIRCUITS is as follows.

- For the T, I, L, X gadgets, if a white box in the line which joints the gadget to another gadget is set to 1 or 3, it means that the corresponding edge is selected, and not selected if set to 2.
- For the $\rightarrow$ and $\leftrightarrow$ gadgets, the edge is selected if white box is set to 9 or 7, and not selected if set to 8.
- For the vartical way, directions of cycles are from 1 to 3 or from 7 to 9 of the jointing white boxes.
- For the horizontal way, the directions are opposite to the vartical way, from 3 to 1 or from 9 to 7.

All sums of those jointing line are set to $10 = 1 + 9$.

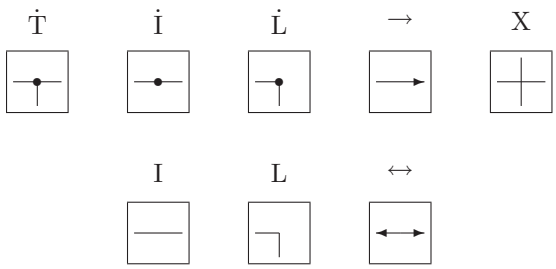


Figure 2.14: Thess figures show the gadgets required.

Once these gadgets are constructed, we can do the reduction as shown in Figure.2.15.

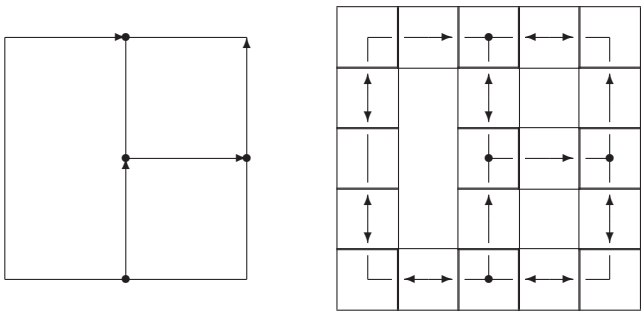


Figure 2.15: This figure shows the reduction to CROSS SUM from restricted PARTITION INTO CIRCUITS.

We can construct the gadgets as following figures show. The sums of lines which connect two gadgets are all 10.

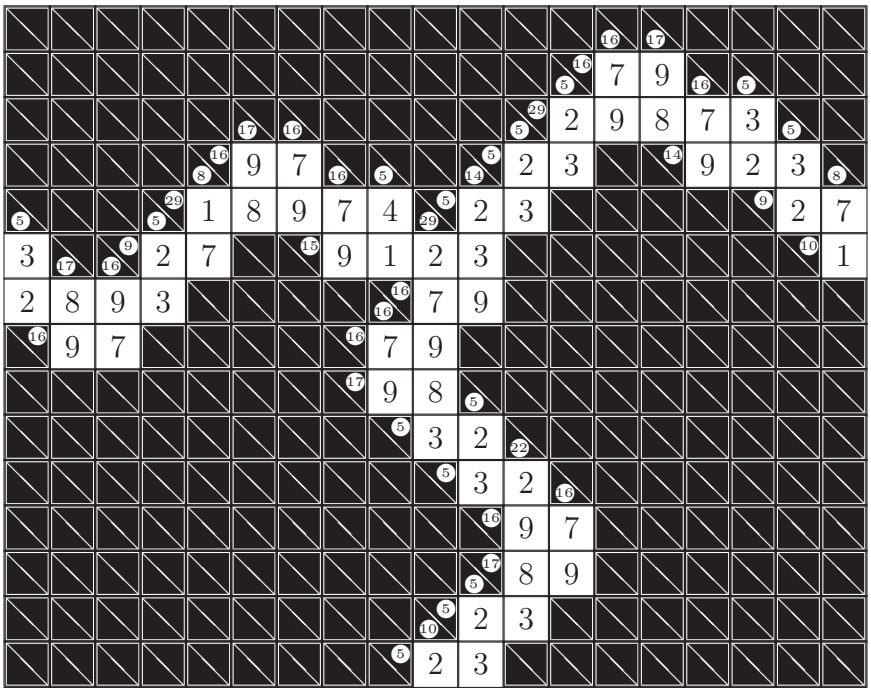


Figure 2.16: This figure shows how to make the gadget $\dot{\mathrm{T}}$ for vertices with three neighbors (or edges) with one possible setting. The gadget itself has no numbers in white boxes. This setting means that the edges at the left and at the right are selected, the edge at the bottom is not selected, and the cycle goes from left to right (from 3 to 1). The size is 19×19, but some rows on the top or on the bottom are omitted, since they are made of black boxes only. Two numbers in black boxes on the right or bottom edges are not determined by this gadget, but by the neighbor gadget.

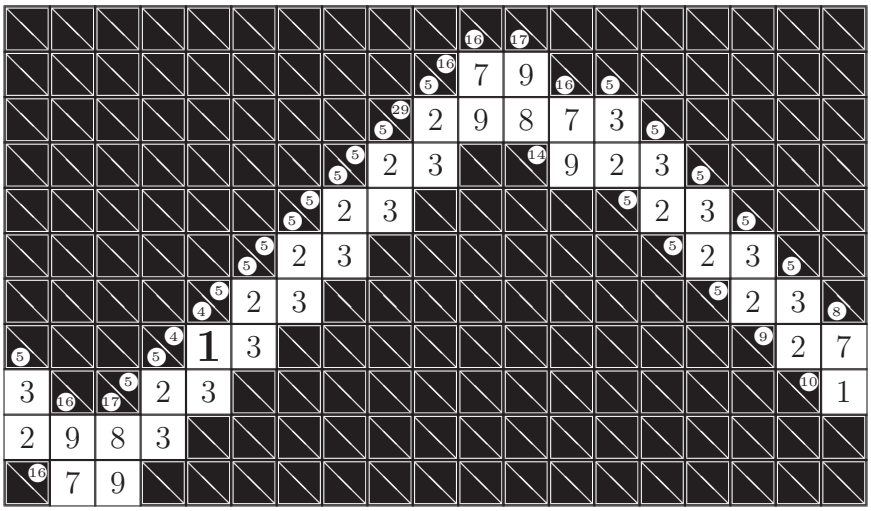


Figure 2.17: This figure shows how to make the gadget İ for vertices with two neighbors (or edges) with one possible setting. The gadget itself has no numbers in white boxes. This setting means that the both edges connecting this vertex are selected, and the cycle goes from left to right (from 3 to 1). The size is 19×19, but some rows on the top or columns on the right are omitted, since they are made of black boxes only. Two numbers in a black box on the right or bottom edges are not determined by this gadget, but by the neighbor gadget. If the number 1 written in **large bold font** is changed to 2, and so are the sums of the lines including the box to 5, then this gadget becomes the gadget I, not a vertex but just a edge.

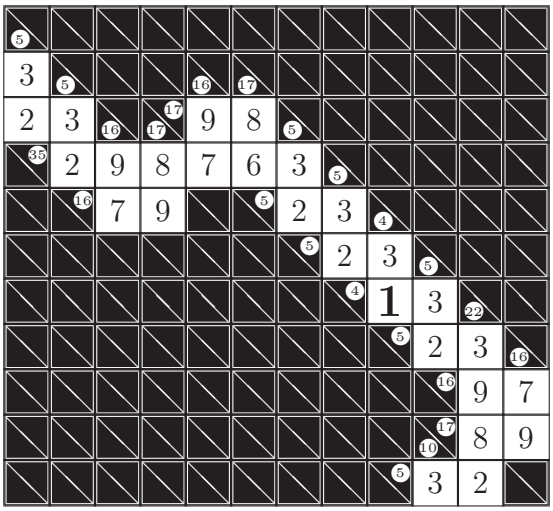


Figure 2.18: This figure shows how to make the gadget $\dot{\mathrm{L}}$ for vertices with two neighbors (or edges) with one possible setting. The gadget itself has no numbers in white boxes. This setting means that the both edges connecting this vertex are selected, and the cycle goes from left to bottom (from 3 of horizontal position to 3 of vartical position). The size is 19×19, but some rows on the top are omitted, since they are made of black boxes only. A number in a black box on the bottom edge are not determined by this gadget, but by the neighbor gadget. If the number 1 written in **large bold font** is changed to 2, and so are the sums of the lines including the box to 5, then this gadget becomes the gadget L, not a vertex but just a edge.

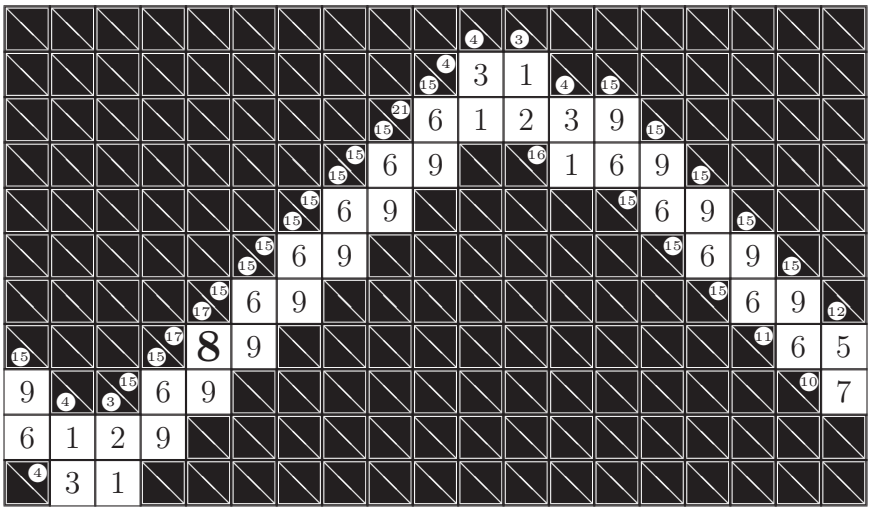


Figure 2.19: This figure shows how to make one of the gadget $\rightarrow$ for directed edges with one possible setting. The gadget itself has no numbers in white boxes. This setting means that this vertex are selected, and the cycle goes from left to right (from 9 to 7). The size is 19×19, but some rows on the top or on the bottom are omitted, since they are made of black boxes only. A number in a black box on the right edge are not determined by this gadget, but by the neighbor gadget. If the number 8 written in **large bold font** is changed to 6, and so are the sums of the lines including the box to 15, then this gadget becomes a $\leftrightarrow$ gadget for undirected edges. When the arrow corresponding this gadget becomes vertical direction, then the right side on this gadget becomes the tail side of the arrow (starting points of edge). In other words the direction of gadgets for a horizontal edge is the reverse of those for a vertical edge.

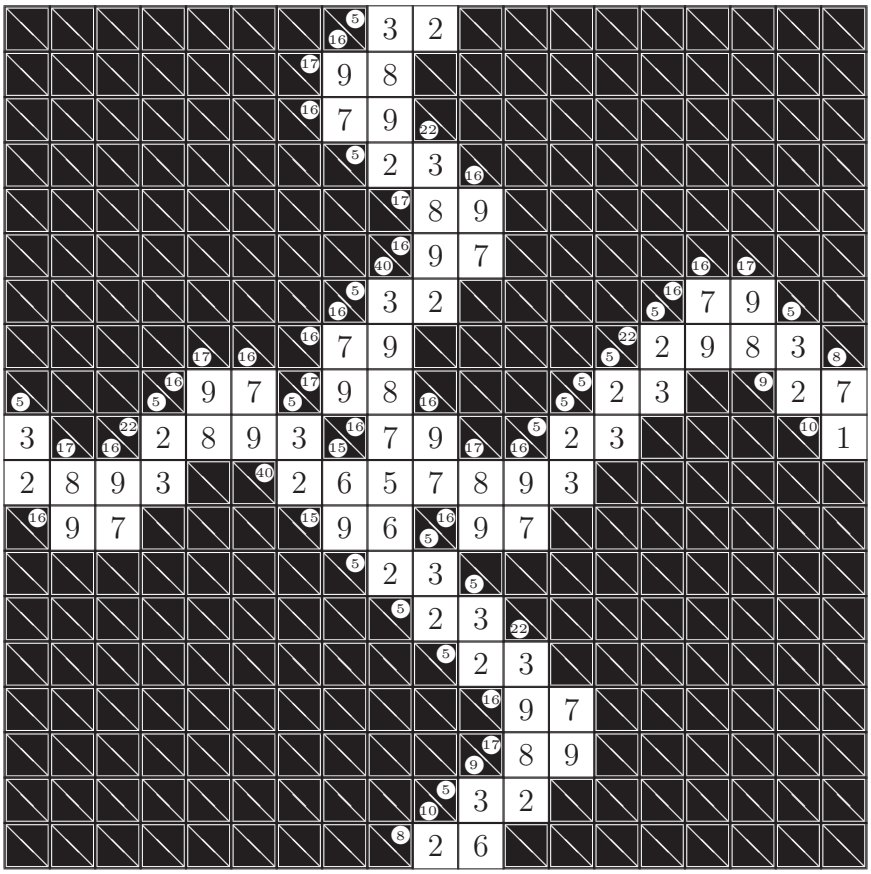


Figure 2.20: This figure shows how to make the gadget X for edge cross with one possible setting. The gadget itself has no numbers in white boxes. The size is 19×19. This setting means that the horizontal edge is selected and the vartical one is not selected, and the cycle on the horizontal goes from left to right (from 3 to 1). Two numbers in black boxes on the right or bottom edges are not determined by this gadget, but by the neighbor gadget.

Now, we got the all gadgets and we can construct the reduction as it was shown in Figure.2.15.

2.2 Extension of CROSS SUM: Where does the complexity come from?

In the previous section, we proved the NP-completeness of CROSS SUM the puzzle which looks like CROSS WORD.

A previous work says CROSS WORD CONSTRUCTION is NP-complete.(See Chapter.1,[7]) And it's also NP-complete even if the board is made with white boxes only.

In the computers, all the data are binary, 0 or 1.

Then, how about CROSS SUM? We know we can't make CROSS SUM problem if white box only. But how about the case we set some limitations on the length of one white line? Is CROSS SUM restricted to binary number also NP-complete? Is the 9 numbers are critical? How about 8 numbers?

So, the question of this section is why CROSS SUM is so difficult. We will consider this problem by making some variants or extensions of CROSS SUM.

2.2.1 Definitions

To extend CROSS SUM, we redefine CROSS SUM as follows.

Definition (N, l, L)-CROSS SUM

Instance A $W \times H$ square board which satisfies following properties

- The board is made of white boxes and black boxes.
- Numbers called sum are given one for each line of white boxes.
- White boxes are connected not separated by black boxes.
- Black boxes can be located next to each other not like CROSS WORD.
- Any line of white boxes has at least $\boldsymbol{l}$ white boxes.
- Any line of white boxes has at most $\boldsymbol{L}$ white boxes.

Problem Does the given instance have any settings of numbers in white boxes which satisfies following properties?

- Numbers in white boxes are from 1 to $\boldsymbol{N}$.
- The sum of numbers in one line is same to given sum.
- Any number appears at most once in one white line.

Definition Hinted CROSS SUM

Instance A $W \times H$ square board which satisfies following properties

- The board is made of white boxes and black boxes.
- Numbers called sum are given one for each line of white boxes.
- White boxes are connected not separated by black boxes.
- Some white boxes have a number in it. The numbers are from 1 to 9. We call them "hint".

- Black boxes can be located next to each other not like CROSS WORD.
- Any line of white boxes has at least 2 white boxes.

Problem Does the given instance have any settings of numbers in white boxes which satisfies following properties?

- Numbers in white boxes are from 1 to 9.
- The setting must not contradict the hints.
- The sum of numbers in one line is same to given sum.
- Any number appears at most once in one white line.

Definition Hinted (N, l, L)-CROSS SUM

Defined in the same way as above.

Note

- We can assume $1 \leq l \leq L \leq N$.
- The original version of CROSS SUM becomes $(9, 2, 9)$-CROSS SUM in this description.
- If $l_1 \leq l_2$ and $L_1 \geq L_2$, (N, l_1, L_1)-CROSS SUM includes (N, l_2, L_2)-CROSS SUM.
- If $N_1 > N_2$, (N_1, l, L)-CROSS SUM and (N_2, l, L)-CROSS SUM are different at all, (N_1, l, L)-CROSS SUM doesn't include (N_2, l, L)-CROSS SUM. But, if the given instance are same and (N_2, l, L)-CROSS SUM has a solution, then (N_1, l, L)-CROSS SUM also has the same solution.
- Hinted CROSS SUM includes CROSS SUM. CROSS SUM is hinted CROSS SUM with no hint.

2.2.2 Conplexities of extensions of CROSS SUM

In this section, we will see the complexities of some CROSS SUM variants.

Problems in P

- $(N, l, 2)$-CROSS SUM is solvable in linear time, especially binary $(2, l, L)$-CROSS SUM. Since we can set numbers in white boxes from the left most of white boxes.

- (N, k, k)-CROSS SUM is solvable in $O(1)$ time. Since we can prove the problems must heve the size $(k+1) \times (k+1)$ with $k \times k$ white boxes by constructing the puzzle from the left most vertical line.
- (N, l, L)-CROSS SUM with limited width W is solvable in linear time of the height and so the size, using dynamic programming method.

proof

We assume the height of the given instance is h.

First, we devide the given problem into $2N \times W$ blocks. These blocks are constant size, and we can list all the possible settings in constant time for each block, and the number of settings are O(1). (Possible setting of one box is from 1 to N, so the total variation of settings will be $O(N^{2NW}) = O(1)$)

Second, we put the blocks from the bottom, memorizing the possible settings of jointing white lines. We can check if each settings of the blocks can be jointed, in $O(1) \times O(1) = O(1)$ time. So, we can solve the whole problem in $O(h/2N) = O(h)$ time.

Problems of NP-complete

- Hinted CROSS SUM is NP-complete. (Since hinted CROSS SUM includes the original CROSS SUM.)
- $(9, 2, 6)$-CROSS SUM is NP-complete. (We didn't use white line of length 7 or more in the proof of the NP-completeness of CROSS SUM.)
- $(N, 2, 5)$-CROSS SUM $(7 \leq N < \infty)$ is NP-complete. (We can make gadgets as shown in 2.14 in the same way. $\dot{\mathrm{T}}$ gadget and $\dot{\mathrm{L}}$ gadget need some big changes and are shown in Figure.2.21. Note we don't need to make the gadget X.)
- $(N, 1, 3)$-CROSS SUM $(7 \leq N < \infty)$ is NP-complete. (We can make gadgets as shown in 2.14 in the same way. Note we don't need to make the gadget X.)

Problems not known to be P or NP-complete

- (N, l, L)-CROSS SUM $(3 \leq N \leq 6)$ is not known to be P or NP-complete. Hints seem not to change the situation.

Seeing these complexities of some CROSS SUM variants, we will find some trends.

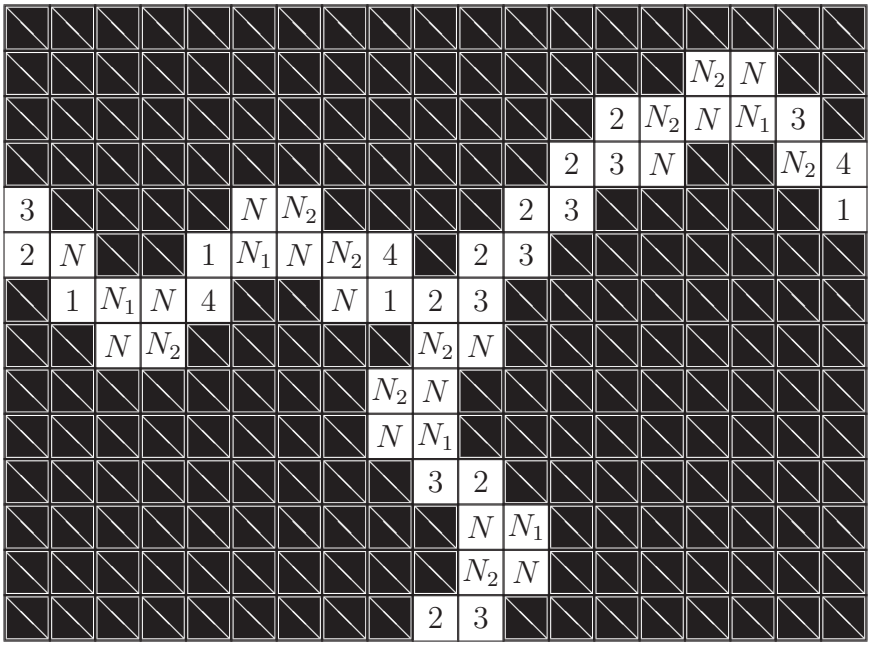
- l and L have some relations with the complexity, the difference $L - l$ seems to play a important role for the variety of puzzles.
- N seems to have some relations with the complexity, CROSS SUM becomes P with binary.
- Not only the size of CROSS SUM determines the time required, the width and the height are slso important.
- Hints seems to be not so impotant from the viewpoint of complexity.

But, we couldn't find a significant trend. We might need some more future works. For example,

- Are there any poly-time algorithms solving $(3, l, L)$-CROSS SUM? $(2, l, L)$-CROSS SUM is easily solved, but it is because of L. So, N is related to the complexity indirectly. Is N itself important or just L is what is important?

2.3 Summary of Chapter.2

- PARTITION INTO CIRCUITS is proved to keep the NP-completeness even if it is restricted to planar and max 3 neighbor.
- CROSS SUM is proved to be NP-complete.
- Some variants of CROSS SUM are suggested.
- Some variants of CROSS SUM are NP-complete, others in P.



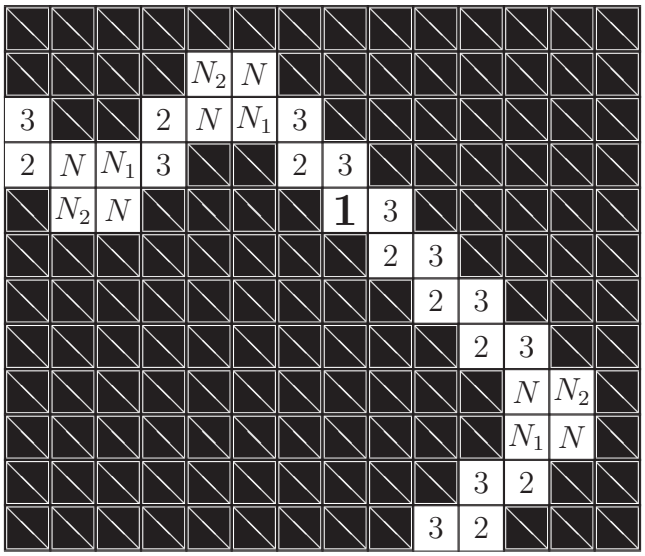


Figure 2.21: These figures show the $\dot{\mathrm{T}}$ gadget and the $\dot{\mathrm{L}}$ gadget of $(N, 2, 5)$-CROSS SUM by one of the possible settings. The sums of lines are omitted. N_1 means $N-1$, N_2 means $N-2$. The $\dot{\mathrm{L}}$ gadget will be L if the number 1 written in **large bold font** is changed to 2.

Chapter 3

Another Solution Problem (ASP)

There are some cases we want to know if there are any solutions other than the given solution. Especially in puzzle problem designing, we need to decide if the solution is unique. Puzzle problems mustn't be ambiguous.

UEDA and NAGAO suggested these problem group and named them ASP (Another Solution Problem) [12], and they also say

> ASP occur frequentry in puzzles. When designing a puzzle, we often determine one solution first and make a puzzle consistent with this solution. Then it is important to check whether the obtained puzzle has no other solution besides the desired one.

So, in this chapter we will consider these problems, ASP.

3.1 Definitions

SAT (SATISFIABILITY)

Input A set of logical clauses of form $L_1 \vee \cdots \vee L_n$.

Question Is there an assignment of variables witch satisfies all the clausess?

3SAT

Input A set of logical clauses of form $L_1 \vee L_2 \vee L_3$.

Question Is there an assignment of variables witch satisfies all the clausess?

HC (HAMILTONIAN CIRCUIT)

Input A graph.

Question Are there any hamiltonian circuit (i.e. a circuit which passes all the vertices) on the graph?

SLITHER LINK

Input A gridded board and numbers in some cells.

Question Are there any circuit which satisfies the following properties on the board?

- The circuit consists of edges of the cells.
- The circuit doesn't cross, or is planar.
- The number in each cell is equal to the numbers of the used edges of the cell.

JUST-ONE SAT

Input Same as SAT.

Question Is there a assignment which satisfies just one literal each clause.

ONE-IN-THREE 3SAT

Input Same as 3SAT.

Question Is there a assignment which satisfies just one literal each clause.

ASP of problem Π

Input A instance of Π and one of its solution.

Question Is there a solution other than the given solution?

Parsimonious reduction

We define a reduction is parsimonious iff there is some one-to-one and onto function from the solutions of the original problem to those of the reduced probrem and the function is computable in polynomial time. Usually just the existance of one-to-one and onto function is necessary, but here the polynomial-time computability is also required. If we have a parsimonious reduction from a problem Π_1 to another problem Π_2 and ASP Π_1 is NP-complete, then we will get the NP-completeness of ASP Π_2. If we have a parsimonious reduction from a problem Π_1 to another problem Π_2 and $\#\Pi_1$ is #P-complete, then we will get the #P-completeness of $\#\Pi_2$.

3.2 The NP-completeness of some ASPs

In this section, we prove the NP-completeness of ASP of some NP-complete problems and CROSS SUM. It is worth considering, because even if a problem is NP-complete, the ASP of the problem can be easier complexity. For example, the answer of ASP

Vertex Coloring and the answer of ASP Hamilton Circuit on cubic (i.e. all vertices connect to 3 edges, or degree 3) graph are always YES [9]. And, there also are problems proved to be NP-complete like ASP 3DM and ASP NONOGRAM [12].

3.2.1 ASP 3SAT

proof Reduction from 3SAT

It's obvious that ASP 3SAT is in P. So, we prove the NP-completeness by showing a reduction from 3SAT. Let the original 3SAT problem be made of N variables $V_1, \ldots, V_N$ and M clauses $L_{1,1} \vee L_{1,2} \vee L_{1,3}, \ldots, L_{M,1} \vee L_{M,2} \vee L_{M,3}$.

Then, we can do the reduction by

1. Prepare new variables $A, T_1, T_2, T_3, D_1, \ldots, D_M$.
2. Devide each clause $L_{j,1} \vee L_{j,2} \vee L_{j,3}$ to 2 clauses $L_{j,1} \vee L_{j,2} \vee D_j, \bar{D}_j \vee L_{j,3} \vee A$
3. Make 7 new clauses $T_1 \vee T_2 \vee T_3$, $T_1 \vee T_2 \vee \bar{T}_3$, $T_1 \vee \bar{T}_2 \vee T_3$, $T_1 \vee \bar{T}_2 \vee \bar{T}_3$, $\bar{T}_1 \vee T_2 \vee T_3$, $\bar{T}_1 \vee T_2 \vee \bar{T}_3$, $\bar{T}_1 \vee \bar{T}_2 \vee T_3$
4. For each variables $V_1, \ldots, V_N, D_1, \ldots, D_M$, make new clauses $V_i \vee \bar{T}_1 \vee \bar{A}$ or $D_j \vee \bar{T}_1 \vee \bar{A}$.
5. Give the solution of all T, as the input solution for ASP.

Because of the 7 clauses made in 3, T_1, T_2, T_3 are all 1, and If A is 1, all variables should be 1 by the clauses made in 4. So, if there is another solution, A must be set to 0, and clauses made in 4 are satisfied. So, the clauses made in 2 become all clauses to satisfy. And, they are equivalant to the original 3SAT problem.

3.2.2 ASP HC

We prove the NP-completeness of HC restricted to max 3 degree planar graph.

proof Reduction from ASP 3SAT

It's obvious that restricted ASP HC is in NP. And a parsimonious reduction from ASP 3SAT is shown in following figures. They are local changes on the method shown in [8].

Figure 3.1: This figure shows how to make an edge must be selected.

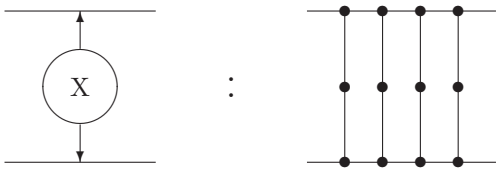


Figure 3.2: This figure shows how to make the gadget for XOR.

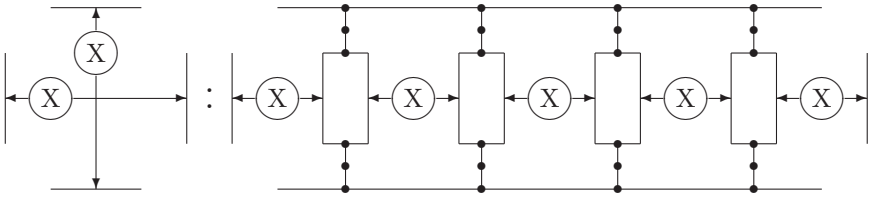


Figure 3.3: This figure shows how to cross XOR gadgets in the planar condition.

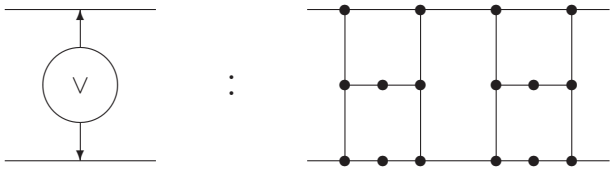


Figure 3.4: This figure shows how to make the gadget for OR.

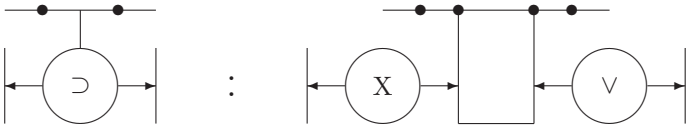


Figure 3.5: This figure shows how to make the gadget for Imply. This gadget is an additional gadget to [8].

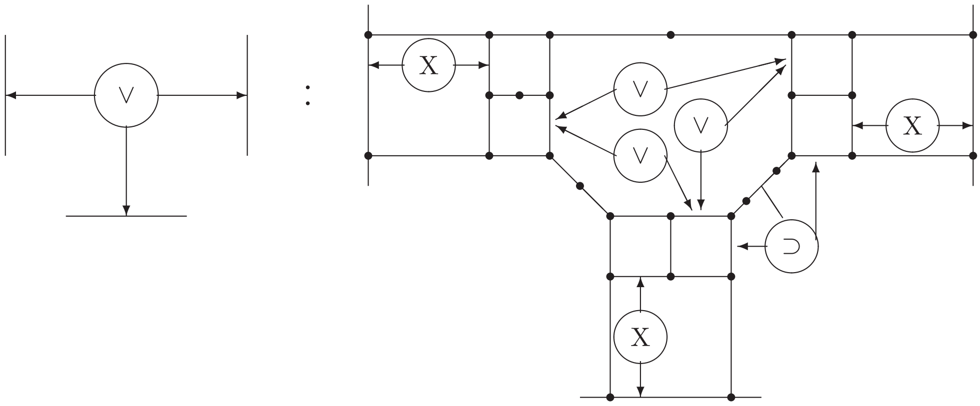


Figure 3.6: This figure shows how to make the gadget for 3OR.

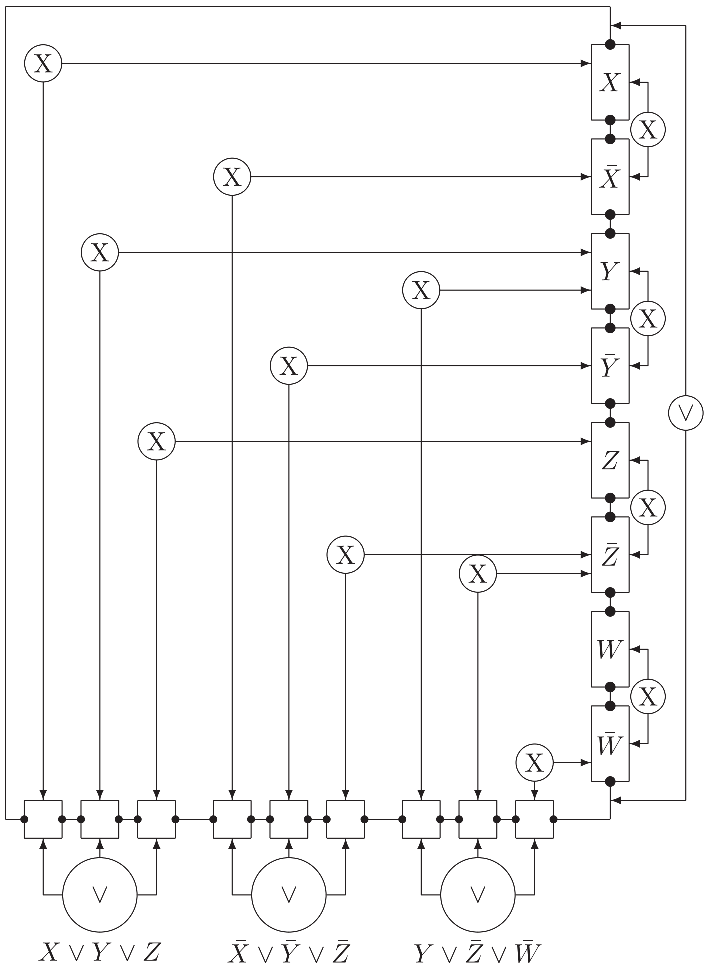


Figure 3.7: This figure shows the reduction from $X \vee Y \vee Z, \bar{X} \vee \bar{Y} \vee \bar{Z}, \bar{Y} \vee \bar{Z} \vee W$.

3.2.3 ASP SLITHER LINK

Reduction from restricted HC

See [13].

3.2.4 ASP ONE-IN-THREE 3SAT

It's obvious that ASP ONE-IN-THREE 3SAT is in P. So we will construct reductions as follows

Reduction from 3SAT to JUST-ONE SAT

First, let the original 3SAT problem be made of N variables $V_i, 1 \leq i \leq N$ and M clauses $L_{j,1} \vee L_{j,2} \vee L_{j,3}, 1 \leq j \leq M$.

Second, for all $1 \leq i < j < k \leq n, 0 \leq l \leq 7$, prepare new variables $V_{i,j,k,l}$, l corresponds the assignments of $V_i, V_j.V_k$ in binary.

Next, for each $i < j < k$, make following clauses

$\bigvee_{l=0}^{7} V_{i,j,k,l}$

$V_{i,j,k,0} \vee V_{i,j,k,1} \vee V_{i,j,k,2} \vee V_{i,j,k,3} \vee V_i$

$V_{i,j,k,4} \vee V_{i,j,k,5} \vee V_{i,j,k,6} \vee V_{i,j,k,7} \vee \bar{V}_i$

$V_{i,j,k,0} \vee V_{i,j,k,1} \vee V_{i,j,k,4} \vee V_{i,j,k,5} \vee V_j$

$V_{i,j,k,2} \vee V_{i,j,k,3} \vee V_{i,j,k,6} \vee V_{i,j,k,7} \vee \bar{V}_j$

$V_{i,j,k,0} \vee V_{i,j,k,2} \vee V_{i,j,k,4} \vee V_{i,j,k,6} \vee V_k$

$V_{i,j,k,1} \vee V_{i,j,k,3} \vee V_{i,j,k,5} \vee V_{i,j,k,7} \vee \bar{V}_k$

Last, for each clauses of the original 3SAT, make a clause which rejects unsatisfying assignment like $\bar{V}_{i,j,k,3}$ for $V_i \vee \bar{V}_j \vee \bar{V}_k$. ($V_i = 0, V_j = 1, V_k = 1$ is the unsatisfying assignment.)

Note that this reduction is parsimonious.

Reduction from JUST-ONE SAT to ONE-IN-THREE 3SAT

First, prepare 4 clauses $T \vee F_1 \vee F_2, T \vee F_2 \vee F_3, T \vee F_3 \vee F_1$

The assignment must be $T = 1, F_1 = F_2 = F_3 = 0$.

Next, for each clauses made of more than 3 literals, do following repeatedly.

> Prepare a new variable, (Let the variable D.) and devide the clause
> $L_1 \vee L_2 \vee L_3 \vee \ldots L_{n-1} \vee L_n$ to 2 clauses
> $L_1 \vee L_2 \vee D$ and $\bar{D} \vee L_3 \vee \ldots L_{n-1} \vee L_n$

Last, for each clauses made of less than 3 literals, put F_1, F_2 and let the clause made of 3 clauses.

Note that this reduction is parsiomnious, and the previous reduction from 3SAT to JUST-ONE SAT was also parsimonious. So, now, we have a parsimonious reduction from 3SAT to ONE-IN-THREE-3SAT and we got the NP-completeness of ASP 3SAT, too.

Removal of negative literal

Rewrite $\bar{A}$ to NA, by preparing $A \vee NA \vee F_1$. F_1 is the variable always assigned to be 0 appeared above. We call the constructed problem as positive ONE-IN-THREE 3SAT, and it's also defined as follows

Input A set of variables V and a family of clauses $\{C_1, \ldots, C_n\}, |C_i| = 3, C_i \subset V$.

Question Is there a assignment $A \subset V$ which satisfies $|A \cap C_i| = 1$ for all i.

Note that this reduction is parsimonious.

Reduction from positive ONE-IN-THREE 3SAT to ASP ONE-IN-THREE 3SAT

First, let the original ONE-IN-THREE 3SAT problem be made of N variables $V_i, 1 \leq i \leq N$ and M clauses $V_{j,1} \vee V_{j,2} \vee V_{j,3}, 1 \leq j \leq M$.

Next, prepare new variables A and $D_j, 1 \leq j \leq M$

Last, devide all clauses $V_{j,1} \vee V_{j,2} \vee V_{j,3}$ to 2 clauses $V_{j,1} \vee V_{j,2} \vee \bar{D}_j, D_j \vee V_{j,3} \vee \bar{A}$, and give the solution of all 0 as the input solution for ASP.

Note that this reduction is parsiomnious, and the previous reduction from ONE-IN-THREE 3SAT to positive ONE-IN-THREE 3SAT was also parsimonious. So, now, we have a parsimonious reduction from ONE-IN-THREE-3SAT to ASP ONE-IN-THREE-3SAT, too.

3.2.5 ASP PARTITION INTO CIRCUITS

We prove the NP-completeness of ASP PARTITION INTO CIRCUITS restricted to max 3 neighbor planar directed graph.

proof Reduction from ASP ONE-IN-THREE 3SAT

Restiricted ASP PARTITION INTO CIRCUITS is obviously in P, and the reduction (with Select gadgets) shown in Chapter.2 is parsimonious.

3.2.6 ASP CROSS SUM

proof Reduction from restricted ASP PARTITION INTO CIRCUITS

ASP CROSS SUM is obviously in P, and the reduction shown in Chapter.2 is parsimonious.

Now, we got 3 puzzle problems whose ASP is also NP-complete. Besides, since most of the reductions above are parsionious and there found parsimonious reductions from 3SAT, whose # problem #3SAT is #P-complete, to CROSS SUM and SLITHER LINK, so we got the #P-completeness of CROSS SUM and SLITHER LINK, too. So, it is easily expected the ASPs of many puzzles are NP-complete and the # problems are #P-complete as the extension of the idea that the interest of puzzles is related to the complexity.

3.3 ASP^n: ASP of ASP

In the previous section we defined ASP as follows

ASP of problem Π

Input A instance of Π and one of its solution.

Question Is there a solution other than the given solution?

But, we could have been define ASP as follows

A possible definition : ASP of problem Π

Input A instance of Π and $\boldsymbol{n}$ of its solutions.

Question Is there a solution other than the given solutions?

We define this variant of ASP as ASP^n. In this description the original problems will be ASP^0 and the original ASP will be ASP^1.

Then a question appears naturally.

How about the complexity of ASP^n?

But, be careful to the reductions shown in the previous chapter. We have already proved the NP-completeness of ASP^n ONE-IN-THREE 3SAT, since the reduction from ONE-IN-THREE 3SAT to ASP ONE-IN-THREE 3SAT was parsionious. Parsimonious reduction is defined as a reduction which maps solutions from the orioginal problem to reduced problem one-to-one and onto, which means if we have a parsimonious reduction from a problem Π to ASP Π and Π is proved to be NP-complete, then we will get the NP-completeness of ASP^n Π for all n.

Note we also have been proved the NP-completeness of PARTITION INTO CIRCUITS and CROSS SUM, since we showed parsimonious reductions from ONE-IN-THREE 3SAT. It is also important: If we have a parsimonious reduction from a problem Π_1 to another problem Π_2 and ASP^n Π_1 is NP-complete, then we will get the NP-completeness of ASP^n Π_2.

Then, how about 3SAT? We can also prove the NP-completeness of ASP^n 3SAT by showing a parsiomonious reduction from ONE-IN-THREE 3SAT to 3SAT.

Parsimonious reduction from ONE-IN-THREE 3SAT to 3SAT

> First, prepare 3 clauses $T_1 \vee T_2 \vee \bar{F}$, $T_1 \vee T_2 \vee F$, $T_1 \vee \bar{T_2} \vee \bar{F}$, $T_1 \vee \bar{T_2} \vee F$, $\bar{T_1} \vee T_2 \vee \bar{F}$, $\bar{T_1} \vee T_2 \vee F$, $\bar{T_1} \vee \bar{T_2} \vee \bar{F}$. These clauses is satisfied only by assingment $T_1 = T_2 = 1, F = 0$.
>
> Next, for each clause of ONE-IN-THREE 3SAT $L_1 \vee L_2 \vee L_3$, add 4 clauses. $L_1 \vee L_2 \vee L_3$, $\bar{L_1} \vee \bar{L_2} \vee F$, $\bar{L_2} \vee \bar{L_3} \vee F$, $\bar{L_3} \vee \bar{L_1} \vee F$, $\bar{L}$ means V, if L is a negative literal $\bar{V}$. These clauses is satisfied iff just one of 3 literals is assigned to 1.

Since ASP^n 3SAT is NP-complete and there exists parsimonious reductions from it, we also proved the NP-completeness of ASP^n HC and ASP^n SLITHER LINK.

Now, we proved the NP-completeness of ASP^{n}s of all problems whose ASP is proved to be NP-complete in this chapter. Besides, the problems include 3SAT, the most important problem on considering NP-completeness. So, it may be expected ASP^n ($n > 0$) has the same comlexity to ASP. We may need some more works on this idea.

3.4 Summary of Chapter.3

- A problem group called ASP has strong relation to puzzle design.
- Some ASPs are proved to be NP-complete, some # problems are proved to be #P-complete.
  - 3SAT
  - HC
  - SLITHER LINK
  - ONE-IN-THREE 3SAT
  - PARTITION INTO CIRCUITS
  - CROSS SUM
- ASPs of many puzzles are expected to be NP-complete.
- # problems of many puzzles are expected to be #P-complete.
- ASP^n is defined.
- ASP^n seems to have same complexity to ASP.

Chapter 4

Puzzle Design

As it was descrived in previous chappter and [12], the ASP problem is storongly related to puzzle construction. And, some ASPs of puzzles are proved to be NP-complete. Then, it may be tiring for humans to confirm a puzzle problem is unambiguous. In that case, some supports by computers will be very helpful.

So, in this chapter, we will consider automatical puzzle design and puzzle design support by computers.

4.1 preliminaries

Before considering automatical puzzle design, we need to review how humans makes puzzles. It goes as follows.[12]

1. Prepare one solution.
2. Make a puzzle problem consistent with the solution.
3. Confirm the puzzle is unambiguous, or have no other solutions, and it is not too difficult or too easy for humans.

For example, if one trying to make a jigsaw puzzle, he/she will do as follows

1. Prepare a picture.
2. Cut the picture into pieces.
3. Confirm it is not too difficult or too easy for humans.

In this case, confirming the uniquness of the solution at the 3rd step might be not neccessary. But, how about with fossil bones? There may be some candidates of restoring the whole dinosour. And, so may be pictures.

Anyway, when one designs a puzzle, he/she need to confirm the puzzle problem is valid, unamibiguous and having a nice difficulty.

4.2 A classification of puzzles focusing puzzle design

If we see puzzles from the viewpoint of puzzle design, we will find a classification by "hints". There are two types in hint giving.

- No freedom for hints. When the solution is given, all hints are determined.
- Some freedom for hints. The constructor can determine how many hints to give.

For example, CROSS SUM, NONOGRAM and NUMBER LINK are the former type of puzzles. SLITHER LINK and NUMBER PLACE are the latter type of puzzles.

There is another classification by hints.

- Hints are part of the answer.
- Hints are just hints, not the answer directly.

For example, NUMBER PLACE and hinted CROSS SUM are the former type of puzzles. CROSS SUM, NONOGRAM, SLITHER LINK, NUMBER LINK and most of puzzles are the latter type of puzzles.

There is also a classification by ambiguity.

- If enough hints are given, then the puzzle becomes unambiguous.
- There are some cases the puzzle remains ambiguous even if all hints are given.

If hints are part of the answer, the puzzle is the former one, and most puuzzles of no freedom for hints are expected to be latter case.

There is a classification by difficulty, too.

- If enough hints are given, then the puzzle becomes easy enough.
- There are some cases the puzzle remains too difficult even if all hints are given.

Like the classification by the ambiguity, if hints are part of the answer, the puzzle is the former one.

If a puzzle is in the former case of the classification by ambiguity and also the former of that by difficulty, then we may design the puzzle easily, by the algorithm described in the following section. And, there seem to be many puzzles in that case. As described above, if the hints are part of the answer, the puzzle is in the case, and there are some more puzzles. SLITHER LINK is one of them.

Prop. SLITHER LINK becomes unambiguous and easy enough with enough hints.

SLITHER LINK itself may be unambiguous if all hints are given as shown in Figure.4.1. But, any SLITHER LINK with size $H \times W$ becomes unambiguous if one more row is added to the top of it and all hints are given. And the $(H+1) \times W$ puzzle is solvable in linear time from top to bottom, left to right as shown in Figure.4.2.

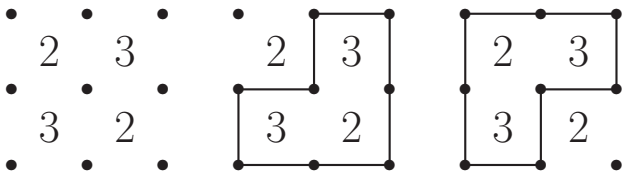


Figure 4.1: These figures show an example of ambiguous instance of SLITHER LINK. Note that all hints are given.

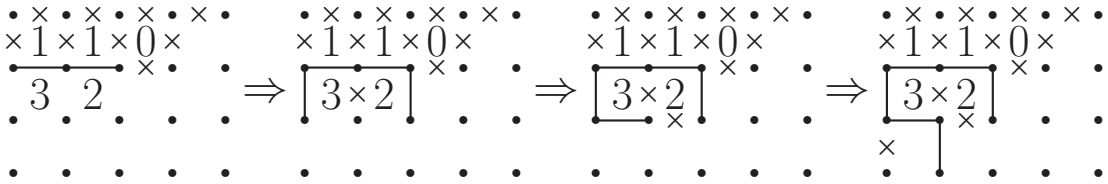


Figure 4.2: These figures show how fully hinted $(H+1) \times W$ is solved in linear time.

If some other puzzles are known to be in the case described above, then those puzzles will be known to be easy to design. So, it's worth considering how the puzzle become if all hints are given.

4.3 An algorithm for puzzle design

If a puzzle becomes unambiguous and easy enough with all hints, it is expected on that puzzle that problems with desirable difficulty is composed with a high probability by the following algorithm.

1. Prepare a solver program which can solve easy problems **only**. We can make this program by limiting the depth of the search or knowledge of well-known cases called *joseki*.

2. Determine a solution, and make a problem with minimum hints consistent with it.

3. Give the problem and some hints to the program.

4. Run the program. If the problem is solved, then a nice problem is designed, else give some more hints and return to 4. It may good to let the program choose which hints to be given.

So, with this algorithm some nice problems may be designed on SLITHER LINK, NUMBER PLACE, hinted CROSS SUM and some other problems.

4.4 Design of CROSS SUM

Last, we will consider the design of CROSS SUM. CROSS SUM is classified to the group of puzzles of no freedom for hints. There are many cases the problem is ambiguous. ASP CROSS SUM is NP-complete, not always NO. And, the ratio of unambiguous instances to all the YES instances seems to become exponensially small as the problem size grows, since for the unambiguity of whole problems, all small parts of puzzles need to be unambiguous. So, we can't apply the algorithm described in Section.4.3.

Then, what can we do with computers on the design of CROSS SUM? Easily we can find the following things.

- First, we can apply the previous algorithm to hinted CROSS SUM. So, we may make nice problems on hinted CROSS SUM.

- Second, we can use computers at the 3rd step of confirming described in Section.4.1 with the same program as the one in Section.4.3. So, we can use computers as a support.

- Third, we can use computers at the 1st step of preparing solution described in Section.4.1 to check if the solution obeys the rules.

But, all of above not so desirable usage of computers. First one is not completely on CROSS SUM, second one needs skills of designer, third one is too easy.

So, here, I suggest a new problem on CROSS SUM, Solution Design Problem.

Input A board of white boxes and black boxes with no numbers in them.

Question Is there a setting of numbers in the white boxes, which make the corresponding CROSS SUM problem unambiguous? If YES, show one of it.

If this problem is easily solved, we may be able to design CROSS SUM problems of nice look. But, the complexity of this question has been not clarified.

4.5 Summary of Chapter.4

- Some classification of puzzles are suggested.
  - By hint type
  - By ambiguity
  - By difficulty
- On some puzzles, nice problems can be design by computers.
- Design of CROSS SUM seems to be difficult.
- Solution Design Problem on CROSS SUM may be important fo design of CROSS SUM.

Chapter 5

Conclusions

In this thesis, first, we considered complexities of puzzles and their ASPs, especially on CROSS SUM, and proved the NP-completeness of some problems and their ASPs as shown in Figure.5.1. So, we got some clues to show the relation between the complexity and the interest of puzzles. We also found that the NP-completeness of ASPs of puzzles may correspond the difficulty of puzzle design, and it is expected that ASPs of many puzzles are also NP-complete.

Second, we defined ASP^n. and found the importance of parsimonious reductions. We also proved that ASP^{n}s of problems shown in Figure.5.1 are all NP-complete, and we got to an idea that ASP^n seems to have the same complexity to ASP. We may need some more studies on this idea.

Third, we defined and considered some extensions of CROSS SUM, in order to find where the complexity of CROSS SUM comes from, and we found some problems are in P, and others are NP-complete. But, significant trends are not found. So, we may need some more study on these extensions.

Last, We considered puzzle design, and we found there are puzzles on which we can easily design problems. And, we saw the difficulty of CROSS SUM design, and we also saw the impotance of Solution Make Problem on CROSS SUM. Further reserches on classifying puzzles are desired, and those on clarifying the complexity of Solusion Design Problem may also be desired.

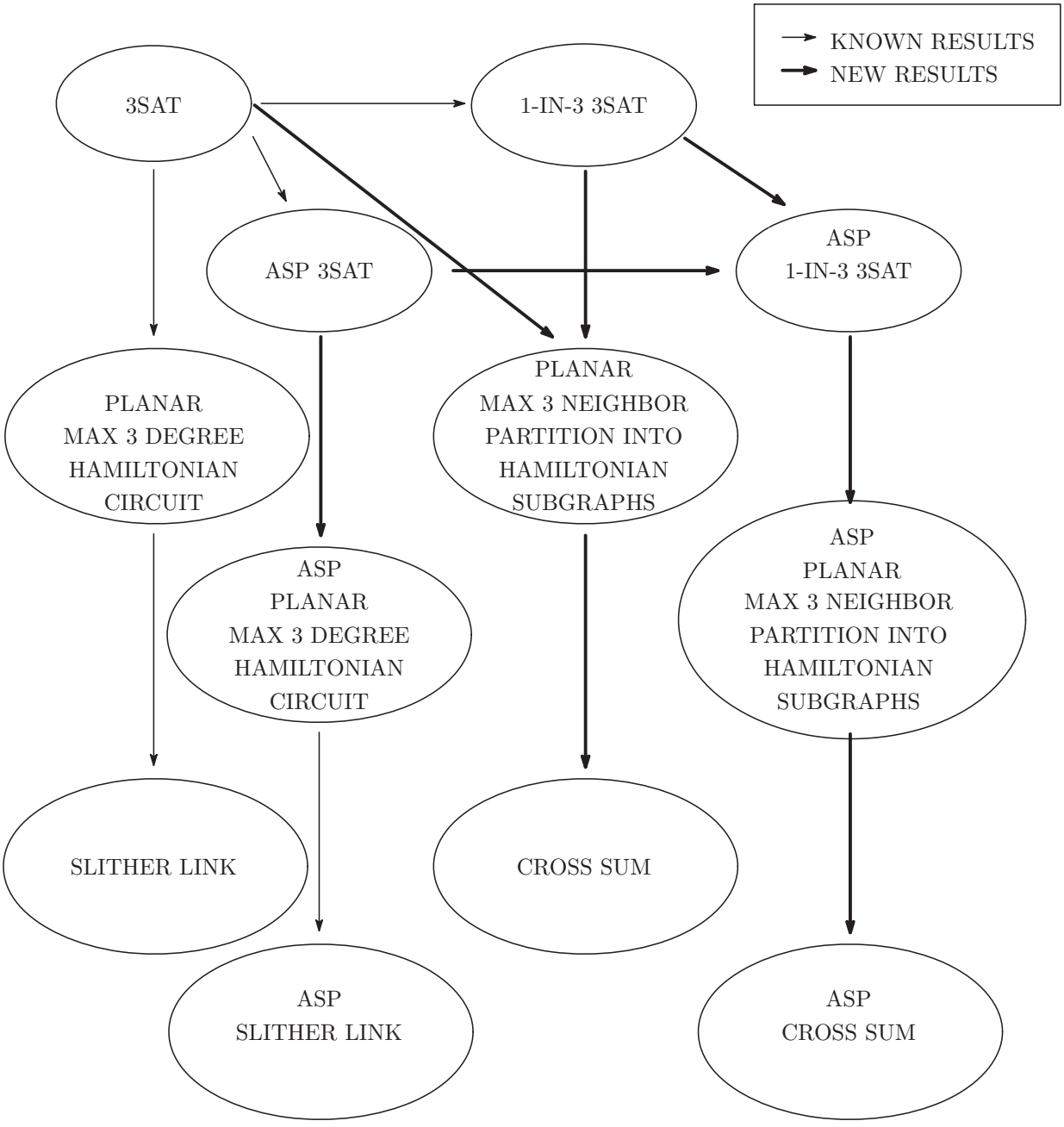


Figure 5.1: This figure shows the problems which were proved to be NP-complete in this thesis and how they were proved. They are drawn with thick lines.

References

[1] Battista, G. D., Eadesa, P., Tamassia, R., and Tollis, I. *Graph Drawing.* Prentice Hall, 1999.

[2] Biedl, T. C., Demaine, E. D., Demaine, M. L., Fleischer, R., Jacobsen, L., and Munro, J. I. The Complexity of Clickomania.

[3] Culberson, J. Sokoban is PSPACE-complete.

[4] Demaine, E. D., Demaine, M. L., and O'Rourke, J. PushPush and Push-1 are NP-hard in 2D.

[5] Eppstein, D. Computational complexity of games and puzzles. http://www1.ics.uci.edu/ẽppstein/cgt/hard.html.

[6] Eppstein, D. On the NP-completeness of cryptarithms. *SIGACT News 18*, 3 (1987), 38–40.

[7] Garay, M. R., and Johonson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., 1979.

[8] Garey, M. R., Johnson, D. S., and Tarjan, R. E. THE PLANAR HAMILTONIAN CIRCUIT PROBLEM IS NP-COMPLETE. *SIAM J. COMPUT. 5*, 4 (December 1976).

[9] Papadimitriou, C. H. *Computational complexity.* Addison-Wesley, 1994.

[10] Ratner, D., and Warmuth, M. Finding a shortest solution for the n*n-extension of the 15-puzzle is intractable. *J. Symb. Comp. 10* (1990), 111–137.

[11] Schaefer, T. J. The complexity of satisfiability problems. In *ACM Symposium on Theory of Computing* (1978), pp. 216–226.

[12] Ueda, N., and Nagao, T. NP-completeness Results for NONOGRAM via Parsimonious Reductions.

[13] Yato, T. On the NP-completeness of slither link puzzle. *IPSJ SIGNotes ALgorithms* (2000).

[14] Puzzle palace. http://www.puzzle.gr.jp/.

[15] WEB Nikoli. http://www.nikoli.co.jp/.