組合せ最適化問題に対する効率的な厳密解法のための問題の記述法に関する研究

乾　伸雄

博士 (情報学)

総合研究大学院大学
複合科学研究科
情報学専攻

平成 23 年度
(2011)

本論文は総合研究大学院大学複合科学研究科情報学専攻に博士（情報学）授与の要件として提出した博士論文である.

審査委員会

速水　謙 (主査)	国立情報学研究所/総合研究大学院大学
相澤　彰子	国立情報学研究所/東京大学
井上　克己	国立情報学研究所/総合研究大学院大学
宇野　毅明	国立情報学研究所/総合研究大学院大学
大山　敬三	国立情報学研究所/総合研究大学院大学
定兼　邦彦	国立情報学研究所/総合研究大学院大学
佐藤　健	国立情報学研究所/総合研究大学院大学

（主査以外は五十音順）

A study on describing combinatorial optimization problems for efficient exact solution methods

Nobuo Inui

DOCTOR OF PHILOSOPHY

Department of Informatics
School of Multidisciplinary Sciences
The Graduate University for Advanced Studies (SOKENDAI)

2011

A dissertation submitted to the Department of Informatics, School of Multidisciplinary Sciences, The Graduate University for Advanced Studies (SOKENDAI) in partial fullfilment of the requirements for the degree of Doctor of Philosophy

Advisory Commitee

Ken Hayami(chair)	National Institute of Informatics/
	The Graduate University for Advanced Studies
Akiko Aizawa	National Institute of Informatics/
	The University of Tokyo
Katsumi Inoue	National Institute of Informatics/
	The Graduate University for Advanced Studies
Keizo Oyama	National Institute of Informatics/
	The Graduate University for Advanced Studies
Kunihiko Sadakane	National Institute of Informatics/
	The Graduate University for Advanced Studies
Ken Satoh	National Institute of Informatics/
	The Graduate University for Advanced Studies
Takeaki Uno	National Institute of Informatics/
	The Graduate University for Advanced Studies

(In alphabet order of last name except chair)

要旨

本論文は，組合せ最適化問題の効率的な厳密解法に関するものである．背景として，近年のハードウェア・ソフトウェア技術の非常な進歩が挙げられる．これらの進歩に伴って，混合整数線形計画問題 (MILP) や充足可能性問題 (SAT) などを解くためのソルバーの性能が飛躍的に向上した．その結果，これまで解くことが出来なかった問題が実用的な時間内に解けたり，厳密解がわからなかった問題の厳密解がわかるようになってきた．そのため，ソルバーは幅広い分野で使われ始めている．

しかし，ソルバーを使って問題を解こうとしても，問題によって解を得るまでの時間にばらつきがあったり，非常に時間がかかることがある．得られた解が問題の厳密解であることを保証している問題記述であっても，このようなことは問題記述が効率的に厳密解を求めるのには不十分である場合に起きる．本論文ではこの点に着目し，計算時間に特化して組合せ最適化問題を効率的に解くための手法を研究した．本論文では三つの重要な既存問題に関して，次の二点を研究した．

1. ソルバーの違いによる効率性の比較
2. 計算時間的効率性を重視した問題記述

ソルバーの違いに関しては，特に近年の性能向上が著しい二つのソルバー，MILP ソルバーと SAT ソルバー，を個々の問題に適用し，問題記述の効率性を比較する．そして，問題記述独自の効率化手法を開発することで，さらに効率的に問題を解くための手法を確立する．

本論文は 6 章からなる．1 章では本研究の動機について述べ，2 章では関連研究について述べる．3 章から 5 章が具体的な研究成果となる．そして，6 章では本論文の研究成果をまとめる．

3 章では決定性有限状態オートマトン (DFA) の学習問題の一つである最小無矛盾 DFA 生成問題を取り上げる．この問題は古典的な研究問題であり，正・負のラベルが与えられた二つの文字列集合に無矛盾な状態数最小の DFA を求めることが目的である．この問題については，近年 SAT によるアプローチが提案され，現在最も効率的な手法であると考えられるが，文字列集合が疎な場合は解くことが難しい問題となってしまう．本章では MILP と SAT による適用を評価し，SAT が効果的であることを実験的に明らかにした．この結果から，SAT を使ったアプローチを更に改善するため，変数の削減，対称性除去制約，ハイパーエッジ彩色問題制約などの新たな問題記述法を導入した．そして，これらの手法について，ベンチマーク問題を使って計算時間を評価した．その結果，提案手法を用いることによって，特に従来手法によって解くことが難しい問題を高速に解くことができるようになった．

4 章では，走査型半導体露光装置における移動順最適化問題を取り上げる．この問題は，ウェハ上のすべての所定の位置に回路パタンを露光する最短経路を決定する問題である．ウェハ一枚の露光時間が少しでも短いと一日の生産量を高めることができ，コスト削減に寄与す

る．走査型半導体露光装置では一枚のウェハ上の複数箇所に回路パタンを走査露光するので，走査方向が上と下の場合，ある露光位置から別の露光位置までの移動時間には（上上，上下，下上，下下）の 4 種類が存在し，最短経路では高々一つの移動時間が選ばれることになる．このような移動順最適化問題は巡回セールスマン問題 (TSP) の一種と考えることができるが，TSP による表現方法は過去に提案されてなく，厳密解を得ることが難しい問題であった．まず，MILP ソルバーと MaxSAT ソルバーの TSP に対する適用可能性を予備実験により評価し，MILP ソルバーの適用が適切であると判断した．次に TSP ソルバーを適用するため，二つの所定位置間の移動時間に対して 3 種類の表現方法を提案し，実用に近い問題で評価した．その結果，MILP で記述するよりも，TSP ソルバーを用いた提案手法によって最短時間の厳密解が効率よく得られることがわかった．

5 章ではナーススケジューリング問題を取り上げる．ナーススケジューリング問題は，病院の看護師の勤務スケジュールを決定する問題であり，病院での勤務体制の効率化のために実用的に重要な研究である．看護師の数を N，シフト数を S，勤務日数を D としたとき，S^{ND} 通りのスケジュールにおいて，勤務に関する制約条件を満たす解の中から最も看護師数に対する要求を満たす解 (ペナルティ最小化) を求める問題である．この問題では，様々な制約が $L \leq \sum_i X_i \leq U$ のような和の形で現れるので，MILP による問題記述は容易である．しかし，実際に MILP ソルバーで解くことは難しいベンチマーク問題が存在した．そこで，SAT による記述を行った．SAT による記述では前記のような和を表現することが効率に影響を与えるが，これに対して二つの方法を提案し，実験的に効果を検証した．この結果，ペナルティが小さい問題に対しては，MILP ソルバーが解けない問題でも SAT ソルバーを使うことによって高速に解が求まることがわかった．

6 章では，個々の問題に関する研究成果をまとめる．最小無矛盾 DFA 生成問題は SAT によるアプローチが有効であり，移動順最適化問題は MILP によるアプローチが有効であった．また，ナーススケジューリング問題は問題によって SAT が有効であったり，MILP が有効である問題であったが，効率的に厳密解を得るための手法をまとめる．

本論文の成果として，三つの既存問題に関して，問題を分析することによって得られるアドホックな条件に関する新規の提案を行い，実験的にそれらの条件の計算時間短縮への寄与を確認した．そして，MILP と SAT による問題記述を行い，実験結果から問題記述の適用可能性を検討した．そこから，組合せ最適化問題の厳密解法に対して効率的な記述方法を示した．

Abstract

In this thesis, I address efficient exact solution methods for several combinatorial optimization problems. The rapid advances of hardware and software technologies are stimulating the development of optimization solvers such as for the mixed integer linear programming problem (MILP) and the satisfiability problem (SAT). In the current state, these solvers have an ability, in the computational time, to solve problems that were recognized, especically in MILP, as unsolved problems 5 years ago. As the result of the development, it began to use many solvers for the methods of many practical and industrial applications.

However, we meet several difficulties when we apply an solver to problems that we want to solve. The biggest difficulty is that we cannot estimate the total running time to get the exact solution. Though this mainly comes from the NP-hardness of the problem, it is sometimes possible to reduce the running time by introducing the alternative description of the problem or using another solver. I, in this thesis, studied how I reduce the total running time to get an exact solution in using solvers. For this purpose, this thesis explores the following two issues for three important existing problems:

1. comparing the efficiency among solvers and descriptions,
2. describing problems in order to make the running-time shorter.

For the first issue, I mainly present two descriptions, MILP and SAT. For the second issue, I introduce various representation methods of problems, additional constraints and so on.

This thesis is composed of 6 sections. The section 1 states the motivation of this thesis and the section 2 shows some related studies. The main results of this thesis are presented in sections from 3 to 5 that are devoted for case studies. Finally, I summarize this thesis in the section 6.

The section 3 addresses the problem of learning deterministic finite state automaton (DFA), the one of classical grammatical inference problems. The goal of this problems is to generate the minimum-state DFA consistent with a positive and a negative set of strings given. Recently, a SAT approach was proposed and is one of the most efficient methods for this problem. But this approach is not sufficient to solver various kinds of benchmarks with the sparseness characteristics. This section makes clear the effectiveness of the SAT approach against the MILP approach, experimentally. Then, I introduce the new description, such as the reduction of variables, the symmetry breaking, the hyper-edge coloring constraint and the combinations of these, for the SAT approach to obtain better running time. Finally, I show the effectiveness of my proposed methods, especially in sparse benchmarks.

I deal with the problem of the movement sequence optimization for the step-and-scan lithography equipment in the section 4. The main purpose of this problem is to reduce the turn-around time to expose several ten or hundred circuit patterns on a wafer. This problem is important for the productivity of the equipment. This problem differs from the traveling salesperson problem (TSP) in the existence of 4 kinds of arcs from a exposure position to another one, i.e. each position can be exposed in one of two scanning directions and the movement time is determined by the combination of scanning directions. I propose a new representation of this problem and evaluate it by MaxSAT, MILP and TSP. The results show that TSP-approach is the best one in the current state of art.

In the section 5, I state the nurse scheduling problem which goal is to make a 1-month working schedule for several ten nurses. This problem is practical and important for improving hospital's environment. The main feature of this problem is to minimize the number of unsatisfied conditions. Especially, I formulate this problem as the minimization of the number of lacked nurses. I use 2 approaches, MILP and SAT. Experimental results shows that each approach has merits and demerits depending on the statistics of benchmarks. If a lot of lacked nurses are required, MILP approach is better than SAT approach, while SAT approach is better than MILP if the number of shifts are large.

The section 6 summarize the contribution of this thesis. The main result of this thesis is to propose new methods and new constraints, which are given by analyzing problems, for three combinatorial problems. I prove the efficiency of proposed methods by experiments. All problems in this thesis are compared by descriptions in MILP and SAT. I present the efficient descriptions of combinatorial problems by issues above.

目 次

表目次

図目次

第1章　はじめに

1.1　研究の背景

本論文は，組み合わせ最適化問題に対する効率的な厳密解法の確立を目的としている．本論文で取り扱う組合せ最適化問題は，解空間 X の要素 $\boldsymbol{x}$ について目的関数 $f(\boldsymbol{x})$ を最小化 (あるいは最大化) する問題として，次のように定義される．

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{x} \in X \end{array} \tag{1.1}$$

ここで，$\boldsymbol{x} \in X$ となる $\boldsymbol{x}$ を実行可能解と呼ぶ．本論文で扱う組合せ最適化問題は，X が離散空間となっている．すなわち，ある実行可能解 $\boldsymbol{x}$ の近傍には実行可能解が存在しない．厳密解または最適解とは，式 1.1 を満たす解 $\boldsymbol{x}$ である．また，近似解とは最適解であることは証明されていないが，ある程度最適解に近い解である．目的関数を $-f(\boldsymbol{x})$ とすることで，最大化問題は最小化問題として扱えることから，特に注意する必要がある場合を除いて，本論文では最小化問題を扱う．

組合せ最適化問題の厳密解は，式 1.1 から次の性質を満たしている．

1. その目的関数値が実行可能解の中で最小であること，
2. その目的関数値よりも小さい目的関数値を持つ $\boldsymbol{x}$ が実行可能解でないこと．

このため，それぞれに対応して，組合せ最適化問題の厳密解は次の二つの方法で求められる．

1. 実行可能解を列挙して目的関数値が最小の実行可能解を見つける，
2. ある実行可能解よりも目的関数値が小さい解 $\boldsymbol{x}$ はすべて $\boldsymbol{x} \notin X$ であることを証明する．

これらの方法は場合によっては区別できないが，1 は状態遷移空間のヒューリスティック探索を用いることで厳密解を求める手法であり，状態遷移空間が列挙可能なときに用いることができる．この方法では，実際に探索しなければならない解空間を小さくすることが解を効率的に求めることにつながる．

これに対して，2 は整数計画問題の整数緩和問題を使った LP ベース分枝限定法 [6] で使われている手法であり，ある値よりも小さい目的関数値を持つ線形計画問題の解が実行可能解

ではないことを証明することにより，厳密解を求める．先のヒューリスティック探索が明示的 (explicit) に実行可能解を列挙するのに対して，LP ベース分枝限定法は暗黙的 (implicit) に解を列挙することで厳密解であることを証明する．この方法では，実行可能解ではない線形計画問題の解を少なくすることが解を効率的に求めることにつながる．

これらの方法のどちらが効率的かということについては，問題によって異なると考えられるため，一概には決められない．例えば，隣接頂点を同じ色に彩色しないで最小の彩色数を求めるグラフ彩色問題 (頂点彩色問題)[66] では，両方のアプローチに関して研究が進められている．1 について，充足可能性問題に基づくアプローチの研究が行われている [32]．また，2 について，混合整数線形計画問題に基づくアプローチの研究が行われている [18]．グラフ彩色問題における全く異なるこれらのアプローチのどちらが優れているかはまだ決着のついていない問題であると言える．

また，一つの問題の記述方法には様々なバリエーションがある．例えば，グラフ彩色問題においては頂点の色を表現するために様々なやり方が試みられている [32]．このような従来研究の結果からは，ある一つの手法が他の手法よりも絶対的に効率が良いという結果は得られていない．これは，問題インスタンスの違いやソルバーの違いなど様々な要因が複雑に絡まっていることが原因であると考えられる．そのため．現状では一つの問題について，様々な問題記述方法を考えることが，効率的に厳密解を求めるのに有効であると考えられる．

一般に問題の規模が大きくなると，組合せ最適化問題の厳密解は実用的な計算時間で求められなくなっていく．そのため，求められた解よりも良い解が存在しないことを証明しない近似解法に比べて，厳密解を求める厳密解法では解ける問題の規模は小さくなってしまう．多くの問題では，計算時間の短さが重視されるため，厳密解を求めるよりは近似解を求めるほうが好まれることが考えられる．しかしながら，近似解法の良さ，すなわち，近似の精度は厳密解がわからなくては詳細な評価を行うことは難しい．また，問題によっては近似解と厳密解の差が実用的な違いとなって現れる場合もありうるので，ある程度の計算時間をかけても厳密解が必要な場合がある．そのため，近似解法の研究は重要であるが，厳密解法の研究も重要であると考えられる．

例えば，本論文で扱うナーススケジューリング問題や移動順最適化問題のような組合せ最適化問題は厳密解が重要である問題の一つである．ナーススケジューリング問題は看護師の勤務表を作成する問題であるが，勤務表の質が看護師たちの作業負荷に密接に関係すると共に，看護師の過不足は患者にとっても大きな問題となる．このような特徴を持つナーススケジューリング問題では，ある程度の計算時間を使っても厳密解を求めることが必要である．移動順最適化問題は半導体露光装置において一枚のウェハを露光するターンアラウンド時間を改善する問題であるが，近似解と厳密解の違いがコンマ数秒程度であっても，複数台の半導体露光装置が 24 時間稼動した場合に生産できるウェハの数には大きな違いがある．決定性有限オートマトンについては，システムの仕様を記述する言語に用いるという応用例があ

る [62]．この応用例では，設計仕様が要求仕様に矛盾していないことを検証するタスクにおいて，最小無矛盾決定性有限オートマトンが効率性を高める役割を果たしている．この場合，厳密解であることが効率性を保証することになる．以上のように，組合せ最適化問題では問題の規模が大きくなると厳密解を求めることは難しくなるが，厳密解を効率的に求めることは大きな意義がある．

組合せ最適化問題の厳密解を求める手法としては，大きく分けて，問題依存のアルゴリズムを用いる方法とソルバーを使う方法の二種類があると考えられる．本論文では，前者ではなく，後者のオープンに利用でき，問題記述さえ与えれば色々な問題適用できるようなソルバーを使った解法を扱う．問題依存のアルゴリズムは，問題の特徴を直接プログラムに反映することができ，また，問題に特化したプログラムを開発できることから，短い計算時間で厳密解を求められる可能性がある．また，個々の問題に依存した特徴を組み込みやすく，それによって効率化を行いやすいという特徴を持っている．これに対して，ソルバーは細かいプログラム制御を行うなどには不向きなものの，その実装では多くの研究の成果が含まれているため，個人で実装する問題依存のアルゴリズムに比べて，少ない手間で大きな利益を得られる可能性がある．また，一つの解法に対して，様々なコンペティションや共通のベンチマーク問題に対して性能を競うことが行われており，問題に依存したアルゴリズムを開発するよりも，多くの異なるソルバーを使って厳密解を求めることを試みることができるという利点がある．

1.2 本研究の目的

1.1 節で述べた背景のもとで，本研究はソルバーを使った効率的な厳密解法を研究する．個別の問題においてどのような問題記述を使えば効率的に問題を解くことが出来るか，また，更に問題を効率的に解くためにはどうすれば良いのかという問題に答えを与え，そこから組合せ最適化問題の厳密解を効率的に求める手法を提案することを研究目的とする．

本研究の第一は，1.1 節で述べた厳密解を求める二つのアプローチ，すなわち，実行可能解を列挙することで厳密解を求めるか，あるいは，ある実行可能解より目的関数値が小さい場合に実行可能解が存在しないことを証明するか，どちらのアプローチが効率的に厳密解を求めるのに効果的であるかを明らかにすることである．この二つのアプローチの効率性を比較することによって，問題の持つ構造的な特徴が明らかになる．すなわち，実行可能解を列挙しやすい問題であるのか，それとも，実行可能解ではない解が列挙しやすいのかが明らかになる．このため，3 章の最小無矛盾 DFA 問題，4 章の移動順最適化問題，5 章のナースケジューリング問題では，それぞれ，混合整数線形計画問題と充足可能性問題によって問題を記述し，どちらの手法が効率的に厳密解を求めることができるかを研究する．

本研究の第二は．問題の表現方法が厳密解法の効率性に与える影響を明らかにすることで，

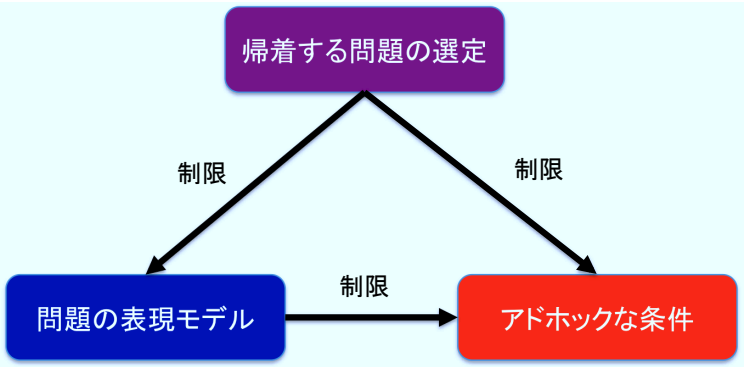


図 1.1: ソルバーによって組合せ最適化問題の厳密解を得る 3 要素

ソルバーを使った手法の効率性を評価することである．一つの手法で問題を表現しても，それが最適な問題の表現方法であるとは限らない．ソルバーを使った厳密解法では，問題の表現方法が効率性と密接に関係してくるため，あらゆる問題の表現方法を試してみなければ，正確な評価を行うことはできない．このため，最小無矛盾 DFA 問題では状態の色を表現する方法に関して，移動順最適化問題では一つの頂点を表現する方法に関して，ナーススケジューリング問題では和の表現方法について複数の表現方法を考案し，厳密解法の効率性を研究する．

本研究の第三は，アドホックな条件を記述することによって，厳密解法の効率化がどのように変わるかを明らかにすることである．現在のソルバーは非常にパフォーマンスの高いものとなっているが，問題を解析することによって得られるアドホックな条件を追加する必要があるのかどうかは大きな問題である．このため，3 章の最小無矛盾 DFA 問題では，カット，対称性除去，複数条件のバイパスなどのアドホックな条件を研究する．

以上をまとめたものを図 1.1 に示す．最初に重要なのは，問題記述を選ぶことである．これは，他の要素で使う方法を制限することになるからである．例えば，複雑な式が現れるような問題は一般に論理式によって記述することは困難である．次に重要なのは，問題の表現方法である．多くのソルバーでは同じ問題を解く場合でも，少しの表現方法の違いが計算時間に大きな影響を与える．実際に掛かる計算時間をあらかじめ予測するのは困難であるため，実験的に試してみる必要がある．最後に重要なのは，問題を効率的に解くための条件である．基本的に解を全列挙できれば，最適解を選ぶことは容易である．もし，解を全列挙するのに近い条件を入れられれば，それは解を効率的に求めることにつながる．これら三つの要素は複雑に絡み合っているが，組合せ最適化問題の厳密解を効率的に求めるためには考慮する必要がある．

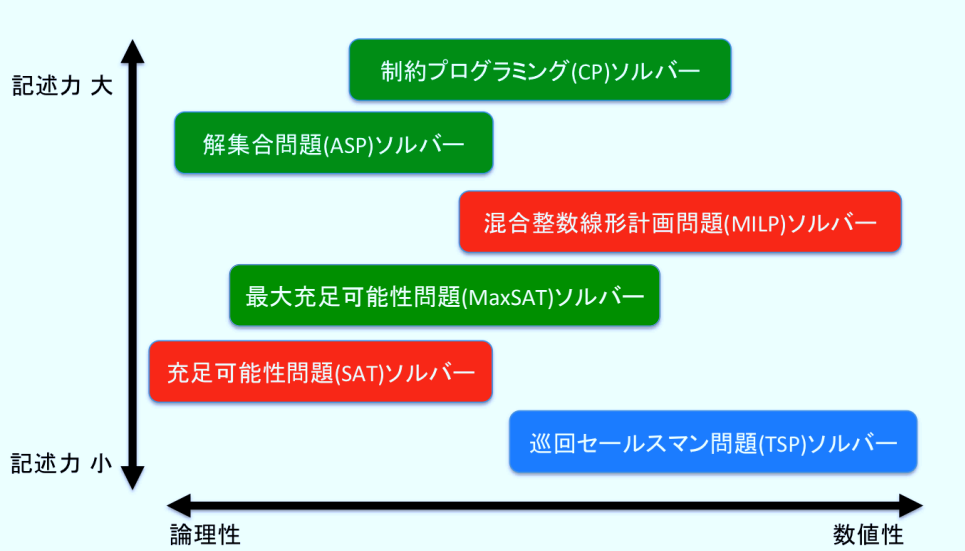


図 1.2: ソルバーとその関係

1.3 ソルバーについて

本節では，研究がよく行われているソルバーについて述べる．本論文が関係するソルバーは，混合整数線形計画問題ソルバー，充足可能性問題ソルバー，および巡回セールスマン問題ソルバーであるが，これらのソルバーに関連したソルバーについても述べる．本節で取り上げるソルバーの直感的関係を図 1.2 に示す．ソルバー間の関係を示すために，横軸に論理性と数値性の軸を設けた．これは，論理表現（真か偽を取る変数間の関係を記述）が基盤にある問題を扱っているか，数値計算が基盤にある問題を扱っているかを示す．縦軸は，記述力に関する軸で，記述力が大きい問題は，少ない変数・条件で問題を記述できることを示している．

1.3.1 MILP ソルバー

混合整数線形計画問題 (MILP: Mixed Integer Linear Programming) は次の式によって与えられる問題である．

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{c}_{\boldsymbol{y}}^{\mathrm{T}}\boldsymbol{y} \\ \text{s.t.} \quad & A_x\boldsymbol{x} + A_y\boldsymbol{y} = B \\ & \boldsymbol{x} \in N^n \\ & \boldsymbol{y} \in Z^m \end{aligned} \tag{1.2}$$

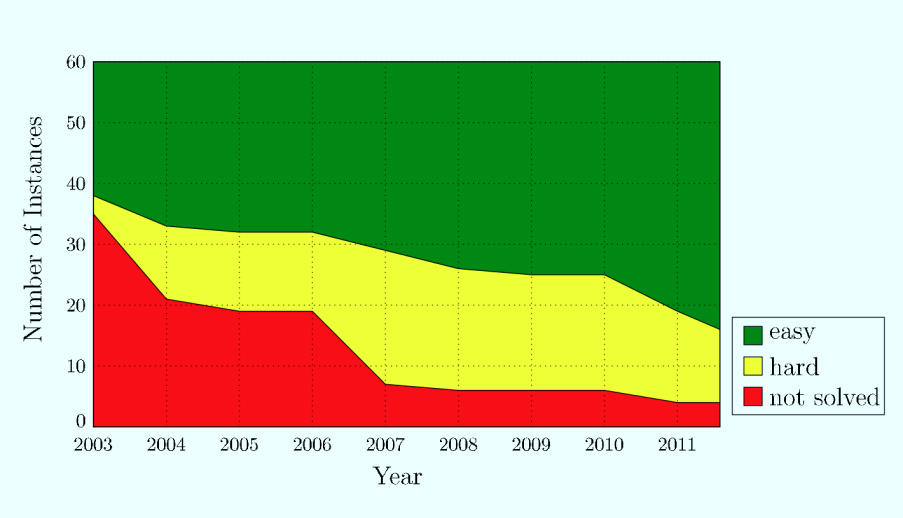


図 1.3: MILP のベンチマーク問題 MIPLIB2003 における既に解けた問題の数の推移 ([71] より引用)

本論文で扱う問題は整数変数しか現れないが，現状の MILP ソルバーは問題が整数変数だけであっても，一般変数が混ざっていても，区別することなく問題を解くことができる．与えられた問題を MILP によって記述することに関しては膨大な従来研究が存在する．一般的に MILP を導く手法に関して，[42] では選言や”big-M” 形式の表現を問題の定義から導く手法を提案している．

MILP ソルバーの扱う MILP は NP 困難な問題である．そのため，大規模な問題を解くことは現状の技術でも難しいが，近年の技術的進歩によりかなりの問題が解けてきた．このことはベンチマーク問題である MIPLIB[71] における既知問題の数を見るとよく分かる．既に解けた問題の数の推移を図 1.3 に示す．このグラフからは，過去において計算時間がかかった問題が最近では短い時間で解けるようになったことが読み取れる．

以下に，IBM ILOG CPLEX(以下，CPLEX と表記する)[24] を例に，説明する．CPLEX のアルゴリズムの詳細は公開されていないが，切除空間法 [9] を用いた LP(Linear Programming: 線形計画法) ベース分枝限定法を用いていると考えられる．切除空間法は，制約条件の不等式と整数変数の条件から導かれる不等式 (カット) であり，LP 解が整数解となるように挿入される．CPLEX においては cliques, covers, disjunctives 他 11 種類のカットが用意されており，非常に強力である．カットは分枝限定法の前処理において挿入される．分枝限定法は，分枝操作と限定操作からなる．分枝操作とは，LP 解が整数とならなかった整数変数を二つの場合に分ける操作のことをいう．０１変数の場合は変数の値を 0 に固定する場合と，1 に固定する場合で場合分けされることになる．限定操作は，例えば目的関数を最小化する最小化問題の場合，分枝操作において LP 解がそれまでにわかっている実行可能解の目的関数の

値よりも大きかった場合は，分枝操作をすることなくそれより先の探索を打ち切り，前の分枝操作に戻ることを言う．一般に，LP 解が整数解になりやすく，最適解が直ぐに求まるような問題は解きやすい問題となる．

現在，組合せ最適化問題の多くの問題が MILP ソルバーによって研究されている．この理由としては，MILP の記述性の高さが挙げられる．他のソルバーと異なり，数式により様々な問題を記述できたり，０１整数変数だけでなく，整数変数や一般変数を扱うこともできる．しかし，枠組みが一般的であるのと裏表の関係であるが，解法が基本的に問題記述に関する前処理と分枝限定法に強く依存しているため，解きにくい問題と解きやすい問題の違いが出やすいという欠点も挙げられる．そのため，MILP ソルバーには制約伝搬などの他の技術と融合したソルバーが存在する [1].

1.3.2 TSP ソルバー

巡回セールスマン問題 (TSP: Travelling Salesman Problem) は，各都市を一度だけ訪問する最短距離の巡回路を求める組合せ最適化問題である．様々な問題の表現方法が考えられているが，都市間の距離に移動方向による違いのない対称型 TSP は次のように MILP によって問題を記述することができる．

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(v)} x_e = 2, v \in V \\ & x_e \in \{0,1\}, e \in E \\ & \{e \in E : x_e = 1\} \text{ contains no subtours.} \end{aligned} \tag{1.3}$$

[37] から引用

ここで，E は都市間を結ぶエッジの集合，c_e はエッジ e の距離，V は都市の集合，$\delta(v)$ はある都市 e を端点とするエッジの集合を示している．この問題記述では，各都市からちょうど２本のエッジが選ばれる条件と選ばれたエッジの部分集合が巡回路ではない条件 (部分巡回路除去条件) からなっている．また，次のようにも記述することができる．

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(v)} x_e = 2, v \in V \\ & x_e \in \{0,1\}, e \in E \\ & \sum_{e \in \bigcup_{v \in V'} \delta(v) \cap \bigcup_{v \in V - V'} \delta(v)} x_e \geq 2, V' \subset V \end{aligned} \tag{1.4}$$

この定式化では，部分巡回路除去を二つの都市集合の間で 2 本以上の経路が存在することにより表現している．先の記述に対して，こちらの記述は部分巡回路が存在しているかどうかを LP 解によって判定でき，空間切除法において整数解を求めることなく部分巡回路除去の条件を追加できるという利点がある．

巡回セールスマン問題の問題定義は，都市の集合および都市間の距離によって与えられることになる．特に，TSP の特殊な場合であるエッジが交差しない平面グラフにおける TSP は効率的に解を求めることができる [96]．しかし，多くの TSP ソルバーで扱われる問題には平面グラフという制限が無いので，様々な問題に応用することができる [61]．

有名な TSP ソルバー concorde[19] は基本的に式 1.4 の定式化を使っている．ホームページによれば，ベンチマーク問題である TSPLIB[92] の 110 問のうち 106 問を解くことができた．現在のコンピュータで実行すればもっと多くの問題を解けるかもしれない．Concorde の基本的なアルゴリズムはカットと分枝限定法を用いている [3]．特にカットは分枝する回数を少なくする効果が高く，効率化への寄与が非常に高い．TSP ソルバーにおけるカットは多面体理論をベースとしており，理論的に非常に美しい形をしているが，詳細を説明するのは本論文の趣旨から離れるので，ここでは述べないことにする．

1.3.3　SAT ソルバー

充足可能性問題 (SAT: Satisfiablity Testing Problem) は，論理式が真となる論理変数の値を求める組合せ最適化問題である．論理式が真になる論理変数が存在する場合，論理式のことを SAT，真になる論理変数が存在しない場合，論理式を UNSAT と呼ぶ．SAT ソルバーにおける論理式は乗法標準形 (CNF: Conjunction Normal Form) の形で表現される．乗法標準形は，論理変数または論理変数の否定の選言で作られる論理式 (選言節) の連言によって論理式を表現する形式である．$\vee$ を選言，$\wedge$ を連言，$\neg$ を否定，$x_i \in \{True, False\}$ を論理変数とおいたとき，次の論理式は CNF である．

$$(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \tag{1.5}$$

この論理式は $x_1 \equiv x_2$ であることを表現している．一般にどのような論理式であっても CNF に変換することができるが，$(x_1 \wedge \cdots \wedge x_n) \vee (x_{n+1} \wedge \cdots \wedge x_m)$ など CNF で表現すると長くなるような論理式も存在する．

SAT ソルバーは毎年開催されている SAT コンペティション [87] を通して，非常な進歩を遂げてきている．SAT コンペティションは，SAT 部門，UNSAT 部門,SAT+UNSAT 部門，並列処理部門など部門に分かれてソルバーを計算時間で評価する．最近のソルバーは SAT も UNSAT も同時に効率よく解けるようになってきたと思われる．図 1.4 に SAT コンペティ

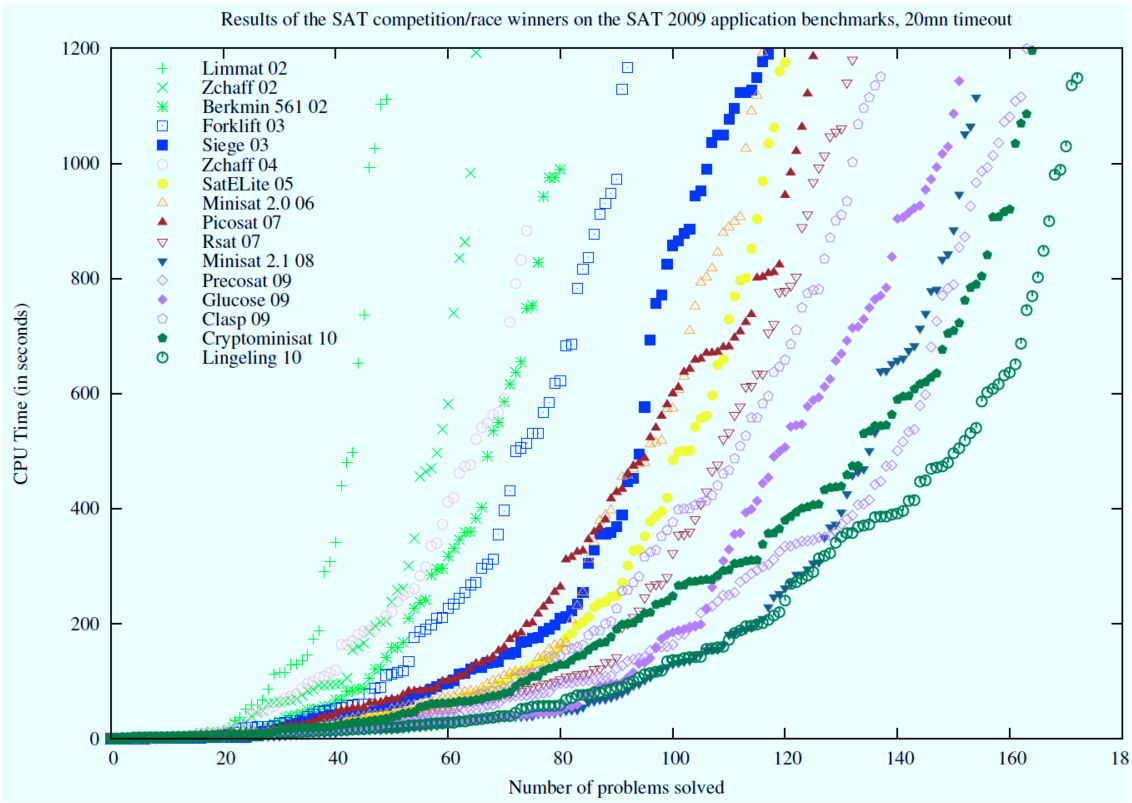


図 1.4: SAT コンペティション優勝ソルバーの経年的性能比較 (横軸は問題数，縦軸は計算時間，右のソルバーほど最近開発されたものになっている)([7] より引用)

ションが始まった 2002 年から 2009 年までの優勝ソルバーが解くことができる問題数と計算時間の関係を示す．このグラフから，同じ時間であれば最近のソルバーが非常に多くの問題を解けるようになったことがわかる．ある意味，SAT ソルバーは他のソルバーに比べ実現が容易 (基本的には個々の選言節が真となる変数対集合を作り，それらの積集合を導けば良い) であり，多くの参加者が参加しやすい．そのため，非常な進歩を遂げてきていると言える．

SAT ソルバーについて 2009 年の優勝ソルバーである clasp[31] を紹介する．clasp は 2011 年の SAT コンペティションでも入賞している息の長い SAT ソルバーであるが，一階述語論理における解を列挙する問題である Answer Set Programming(ASP) を元としている．clasp は他の SAT ソルバーと同じく制約伝搬に基づいたシステムである．制約伝搬とは，ある変数を固定することによってそれに依存した変数の値を制約するというプロセスをすべての変数が固定されるかある変数が値を持たなくなるまで繰り返す手法である．制約伝搬において，矛盾が発見された場合，矛盾の原因となった原因を計算する．これは nogoods と呼ばれ，clasp は実行可能解が満たさない nogoods を制約伝搬において保持し，非常に効率的に論理式の充足可能性を証明する．詳細は参考文献を参照してもらいたいが，現在では百万変数程

度の論理式を扱うことができるになった．現在の技術動向は参考文献 [45] に詳しく解説されているので，参照されたい．

1.3.4 MaxSAT ソルバー

SAT と同じく最大充足可能性問題 (MaxSAT) は論理式の充足可能性を証明するソルバーであるが，SAT が論理式全体を充足する論理変数の値を求めるのに対して，MaxSAT は充足可能な選言節の数を最大化する論理変数の値を求める．このため，SAT が判定問題であるのに対して，MaxSAT は最適化問題と考えることができる．SAT 同様に，MaxSAT についても 2006 年より毎年コンペティションが開かれており [67]，開発競争が活発である．

論理式を証明するという数学的な意味において，SAT が NP 完全な判定問題を扱っているのに対し，MaxSAT は NP 困難な最適化問題として扱うため，一般に MaxSAT は非効率なアプローチである．実際に各々のコンペティションにおいて，SAT は最大で数百万変数までの規模の問題を扱っているのに対し，MaxSAT で扱われている問題は数千変数規模である．しかしながら，組合せ最適化問題は最適化問題に属する問題が多く存在し，また，実際のアプリケーションでもすべての論理式を充足する必要がないことが多いため [56]，MaxSAT が有効なアプリケーションが存在する．

MaxSAT を一般化したバリエーションとして，個々の選言節に重みをつけ，充足しない選言節の重みの総和を最小化する問題を weighted MaxSAT 問題と呼ぶ．weight MaxSAT 問題は適用範囲が非常に広い問題であると考えられる．一般に巡回セールスマン問題のような目的関数が何らかの正の数値の和で与えられるような問題を SAT および MaxSAT で記述することは困難であるが，weighted MaxSAT 問題では非常に簡単に問題を記述することができる．例えば，対称型巡回セールスマン問題では，都市間のエッジ $x_1, x_2, \ldots$ および距離 $c_1, c_2, \ldots$ に対して，次のように問題を表現することができる．

$$
\begin{array}{ll}
c_1 : \neg x_1 & \text{(a)} \\
c_2 : \neg x_2 & \\
\ldots & \\
maxW : \bigvee_{e \in \delta(v) \setminus u} x_e . v \in V, u \in \delta(v) & \text{(b)} \\
\{e \in E : x_e = \mathrm{T}\} \text{ contains no subtours} & \text{(c)}
\end{array}
\tag{1.6}
$$

この表現 (a) では，選言節が充足されないエッジ e_1 が真（すなわち，エッジ e_1 が結ぶ都市間が巡回路の経路上にある）のとき，重み c_1 が重みの総和に加えられる．(b) ではある都市のエッジから一つを除いたものの中に少なくとも一つ真となるエッジが存在することを表し，一つの都市から少なくとも 2 本のエッジが出ることを表現する．(c) は部分巡回路除去を行う論理式である．(b) と (c) はすべて必ず満たしていなければならない論理式である．そのた

め，それらの重みは非常に大きい数を与え，充足しない場合大きな重みを総和に与えることによって，必ず充足するようにする．weighetd MaxSAT をさらに特殊化した問題に partial weighted MaxSAT がある．これは，必ず満たさなければならないハード条件と満たすべき条件であるソフト条件に分けて論理式を記述できる．先の対称型巡回セールスマン問題では重み maxW が与えられた条件をハード条件として明示的に与えることができる．

MaxSAT ソルバーは，MILP と同じく，分枝限定法を用いて実現されることが多い．しかしながら，MILP では LP を解くことにより下界値をもとめることができたが，MaxSAT ではそれは困難であり，ヒューリスティックを使って求めることになる [40].

1.3.5 ASP ソルバー

解集合プログラミング (ASP: Answer Set Programming) は宣言的プログラミング言語である．ASP は LPARSE と呼ばれる論理型言語 Prolog のスタイルを使って記述されるが，実際のプログラムの実行方法は全く異なるものである [63]. ASP において SMODEL とは真となる述語の組を表す．例えば，次のプログラムの場合，$1\{s,t\}$ は真となる述語を少なくとも一つ選ぶことを表すので，SMODEL は $\{s\},\{t\},\{s,t\}$ となる．ASP ソルバーは SMODEL を高速に生成することを目的としたソルバーである．

$$1\{s,t\} \ :- \ not\ p. \tag{1.7}$$

同様に，次のような場合，SMODEL は $\{s\},\{t\}$ となる．

$$1\{s,t\}1 \ :- \ not\ p. \tag{1.8}$$

ASP を使って次のように巡回セールスマン問題を記述することができる ([30] より抜粋).

$$\begin{array}{ll}
1\{inPath(X,Y) : arc(X,Y)\}1 \leftarrow vertex(Y), not\ start(Y) & \text{(a)} \\
\{inPath(X,Y) : arc(X,Y)\}1 \leftarrow vertex(X) & \text{(b)} \\
reached(X) \leftarrow start(X) & \text{(c)} \\
reached(X) \leftarrow reached(Y), inPath(Y,X) & \text{(d)} \\
\leftarrow vertex(X), not\ reached(X) & \text{(e)} \\
\#minimize\{inPath(X,Y) : cost(X,Y,C) = C\}. & \text{(f)}
\end{array} \tag{1.9}$$

ここで使われている述語の意味は次のとおりである．$vertex(X)$ は都市 X が存在することを示す事実節である．$start(X)$ は都市 X が唯一の出発都市であることを示す事実節である．$arc(X,Y)$ は都市 X と Y が隣り合った都市であることを示す事実節である．$cost(X,Y,C)$ は都市 X と Y の距離が C であることを示す事実節である．

上記のプログラムで述語 $inPath(X,Y)$ は巡回路において，都市 X から Y への移動を表す述語である．(a) では，出発都市でない都市 Y の前に訪れる都市 X がちょうど一つ存在すること

を表す．(b) では，都市 X の次の都市が高々一つ存在することを表す．述語 $reached(X)$ は巡回路上にある都市を表現する．(c) は出発都市は巡回路上にあることを示し，(d) は $reached(Y)$ かつ $inPath(Y, X)$ であれば $reached(X)$ であることを示す．MILP および MaxSAT による表現では，手続き的に巡回路を表現することができないため，部分巡回路が生成されてしまう問題を解決しなければならないが，ASP では手続き的に $reach(X)$ が真となるような都市を順番に決めていくことができるため，問題を表現する上では部分巡回路の問題を考慮しなくてもよい．(e) は制約条件を表し，矢印の右側は成立しない．そのため，(e) は X が都市ならば $reach(X)$ であることを示す．(f) は clasp で採用されている最適化関数であり，$inPath(X, Y)$ であるような都市間のコスト C の総和を最小化する．

ASP の記述はホーン節をベースとしているが，ソルバーの親和性は SAT や MaxSAT と非常に高い．そのため，clasp は ASP を解くためのソルバーであるが，SAT と MaxSAT にも適用され，非常に高性能であることをコンペティションの結果は示している．

1.3.6　CP ソルバー

制約プログラミング (CP: Constraint Programming) は汎用的なプログラミング言語であるが，組合せ最適化問題を解くのに有効な枠組みである．制約プログラミングといった場合，処理系によって様々な定義がされ，統一的な枠組みは存在しないが，共通の機能としてドメイン変数と遅延評価機構がある．ドメイン変数とは離散的な値を取ることができる変数であり，遅延評価はドメイン変数間の関係を記述することで手続き型言語のような実行手順を考慮することなく，ドメイン変数に値を割り当てる機構である．また，制約プログラミングは，最大化・最小化の機能を持っているため，最適化問題を容易に記述することができる．代表的な制約プログラミングソルバーには IBM ILOG CP[23] がある．

1.4　本論文の構成と要旨

本論文は 6 章からなる．1 章では本研究の動機について述べ，2 章では関連研究について述べる．3 章から 5 章が具体的な研究成果となる．そして，6 章では本論文の研究成果をまとめる．

3 章では決定性有限状態オートマトン (DFA) の学習問題の一つである最小無矛盾 DFA 生成問題を取り上げる．この問題は古典的な研究問題であり，正・負のラベルが与えられた二つの文字列集合に無矛盾な状態数最小の DFA を求めることが目的である．この問題については，近年 SAT によるアプローチが提案され，現在最も効率的な手法であると考えられるが，文字列集合が疎な場合は解くことが難しい問題となってしまう．本章では MILP と SAT による適用を評価し，SAT が効果的であることを実験的に明らかにした．この結果から，SAT を

使ったアプローチを更に改善するため，変数の削減，対称性除去制約，ハイパーエッジ彩色問題制約などの新たな問題記述法を導入した．そして，これらの手法について，ベンチマーク問題を使って計算時間を評価した．その結果，提案手法を用いることによって，特に従来手法によって解くことが難しい問題を高速に解くことができるようになった．

4 章では，走査型半導体露光装置における移動順最適化問題を取り上げる．この問題は，ウェハ上のすべての所定の位置に回路パタンを露光する最短経路を決定する問題である．ウェハ一枚の露光時間が少しでも短いと一日の生産量を高めることができ，コスト削減に寄与する．走査型半導体露光装置では一枚のウェハ上の複数箇所に回路パタンを走査露光するので，走査方向が上と下の場合，ある露光位置から別の露光位置までの移動時間には（上上，上下，下上，下下）の 4 種類が存在し，最短経路では高々一つの移動時間が選ばれることになる．このような移動順最適化問題は巡回セールスマン問題 (TSP) の一種と考えることができるが，TSP による表現方法は過去に提案されてなく，厳密解を得ることが難しい問題であった．まず，MILP ソルバーと MaxSAT ソルバーの TSP に対する適用可能性を予備実験により評価し，MILP ソルバーの適用が適切であると判断した．次に TSP ソルバーを適用するため，二つの所定位置間の移動時間に対して 3 種類の表現方法を提案し，実用に近い問題で評価した．その結果，MILP で記述するよりも，TSP ソルバーを用いた提案手法によって最短時間の厳密解が効率よく得られることがわかった．

5 章ではナーススケジューリング問題を取り上げる．ナーススケジューリング問題は，病院の看護師の勤務スケジュールを決定する問題であり，病院での勤務体制の効率化のために実用的に重要な研究である．看護師の数を N，シフト数を S，勤務日数を D としたとき，S^{ND} 通りのスケジュールにおいて，勤務に関する制約条件を満たす解の中から最も看護師数に対する要求を満たす解 (ペナルティ最小化) を求める問題である．この問題では，様々な制約が $L \leq \sum_i X_i \leq U$ のような和の形で現れるので，MILP による問題記述は容易である．しかし，実際に MILP ソルバーで解くことは難しいベンチマーク問題が存在した．そこで，SAT による記述を行った．SAT による記述では前記のような和を表現することが効率に影響を与えるが，これに対して二つの方法を提案し，実験的に効果を検証した．この結果，ペナルティが小さい問題に対しては，MILP ソルバーが解けない問題でも SAT ソルバーを使うことによって高速に解が求まることがわかった．

6 章では，個々の問題に関する研究成果をまとめる．最小無矛盾 DFA 生成問題は SAT によるアプローチが有効であり，移動順最適化問題は MILP によるアプローチが有効であった．また，ナーススケジューリング問題は問題によって SAT が有効であったり，MILP が有効である問題であったが，効率的に厳密解を得るための手法をまとめる．

第2章　関連研究

本章では，本研究の目的である「組み合わせ最適化問題の厳密解法の確立」に関連した従来研究を見てみる．

2.1　どのようなソルバーを選ぶべきか

本研究では，この研究目的を「どのような問題に対してはどのような解法を選択すべきか」という問題に答えることと考えている．この問題の答えを得るためには，いくつかの問題をいくつかの手法で解き，その中から共通の特徴を導き出すということが必要である．しかし，個別の問題を解くさいには，単に問題を解くだけでは不十分であり，個々の解法を 100%有効に使わなければ，その解法を評価したことにはならない．

これに対して，通常の研究は一つの問題に対する効率性を追求しているので，その問題の中での解法の効率性は議論できるが，異なった特徴を持つ問題に対してその議論は直接適用できないと考えられる．多くの研究では一つの問題を解くための手法を議論しているが，その手法を他の問題に適用したとき，元の問題での手法の評価がそのまま適用できるかは疑問である．

このような中で，特定の汎用ソルバーについて，どういう問題に適用すべきかを議論した論文がある [73]．この論文は整数計画問題を対象にしているが，問題を発見的な手法で解くことに比べて，MILP ソルバーを使って解くことには，次のような利点があると述べている．

1. 最適性の証明が得られる，
2. 下界／上界が出る，
3. 問題が不能 (解がない) の場合，
4. プログラム経験が不要．

特に組み合わせ最適化問題のような実行可能解の全列挙が困難な問題では，MILP ソルバーが行う最適性の証明は重要である．これらの特徴に加えて，MILP ソルバーのような汎用ソルバーは日進月歩の進歩をしており，手軽に最新の研究成果を試すことができるという点は見逃せない．ただし，効率よく問題を解こうとした場合は，問題を深く解析することは必要であり，すべてを MILP ソルバーに任せるというわけにはいかない．

この論文の著者は，MILP ソルバーによって問題が解けない場合の対処方法について議論している．そこでは次のように述べている．

1. あきらめが肝心，
2. なぜ解けないか分析する．

この中で，なぜ解けないか分析するということは非常に重要である．MILP ソルバーの場合は，特に，整数緩和問題 (整数変数を一般変数として線形計画問題) の解を分析することを奨励している．具体的には，整数になっていない変数の割合を見るべきということを主張している．

この論文のように，ある問題の解法として MILP ソルバーを用いるべきかどうかという議論は豊富な経験の裏打ちがないとできないものであり，非常に大切である．しかしながら，このような議論が複数のソルバーに対してなされた論文は，筆者の知る限り，存在しない．

2.2　グラフ彩色問題の場合

前節の議論を踏まえて，グラフ彩色問題おける各種汎用ソルバーの利用状況についてみてみる．グラフ彩色問題は，非常に簡単な構造をしており，多くの応用分野を持つ問題である．例えば，コンパイラーにおいて各変数を CPU のレジスターに割り当てるレジスター割り当て問題はグラフ彩色問題として記述されることが知られており [2]，このレジスター割り当て問題を対象にしたグラフ彩色問題の解法については多くの研究がある (例えば，[14])．また，数々の手法が試みられてる問題でもある．本節では，このようなグラフ彩色問題の汎用ソルバーによる解法についてみる．

2.2.1　MILP

MILP を用いた解法として，空間切除法を用いたアプローチ [69] について述べる．

この手法は次の MILP による問題記述を出発点としたアプローチである．

$$
\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{j=0}^{n} w_j \\
\text{s.t.} & \sum_{j=1}^{n} x_{ij} = 1, \forall i \in V, \\
& x_{ij} + x_{kj} \leq w_j, (i,k) \in E, 1 \leq j \leq n, \\
& x_{ij} \in \{0,1\}, \forall i \in V, 1 \leq j \leq n, \\
& w_j \in \{0,1\}, 1 \leq j \leq n.
\end{array}
\tag{2.1}
$$

表 2.1: 挿入したカット ([69] から引用)

C1	Clieque
C2	Clieque+Block Color+MultPath
C3	Clique+Block Color+MultPath+MultCli
C4	Clique+MultCli+Hole
C5	Clique+Block Color+MultPath+MultCli+Hole

ここで，V は頂点の集合，E は隣接辺の集合，x_{ij} は頂点 i が色 i になるとき 1 となる変数，w_j は色 i が使われるとき 1 となる変数である．この問題の整数緩和問題は記述力が弱い（整数になるべき変数がＬＰ解において小数となる）ことが知られているため，この整数緩和問題に逐次カットと呼ばれる条件を挿入する．筆者らは independent set, multicolor hole,multicolor clique, multicolor path によるカットを導入し，空間切除法 (整数緩和問題を繰り返し解き，カットに違反する頂点集合に対してカットを挿入する手法) によりグラフ彩色問題を解いた．そして，ベンチマーク問題を使ってカットの組み合わせを評価した．この論文はカットがどの程度下界値を上げるかを評価しているため，グラフ彩色問題の厳密解を求めているわけではい．ただし，整数緩和問題から得られる下界値とヒューリスティック探索による見つかる上界値のギャップが 0 の場合は厳密解が求まったことになる．

評価はグラフ密度に応じて行われた．図 2.1 に論文から引用した実験の要約を示す．ここで，ギャップはカットを入れていない整数緩和問題の目的関数とカットを入れた後の整数緩和問題の平均ギャップである．平均ギャップが大きいほどカットによって目的関数値が上がったことになる．用いたカットについて，各記号は表 2.1 に示したとおりである．この図から，カットは多種類挿入することで効果が高いことがわかる (その代わり，カット生成の時間はかかる)．また，C3 と C4 については高密度の場合ギャップの違いは小さいが，低密度の場合大きくなるなど，グラフ密度によってカットの効果が異なることがわかる．

2.2.2 集合被覆問題

集合被覆問題も MILP ソルバーを用いるアプローチである．本節では，[38] による手法でこのアプローチを紹介する．集合被覆問題では，前節の MILP の問題記述とは大きく異なる次の問題記述を用いる．

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in S_{max}} x_S, \\ \text{s.t.} \quad & \sum_{S \in S_{max}: v \in S} x_S \geq 1, \forall v \in V, \\ & x_S \in \{0,1\}, \forall S \in S_{max}. \end{aligned} \tag{2.2}$$

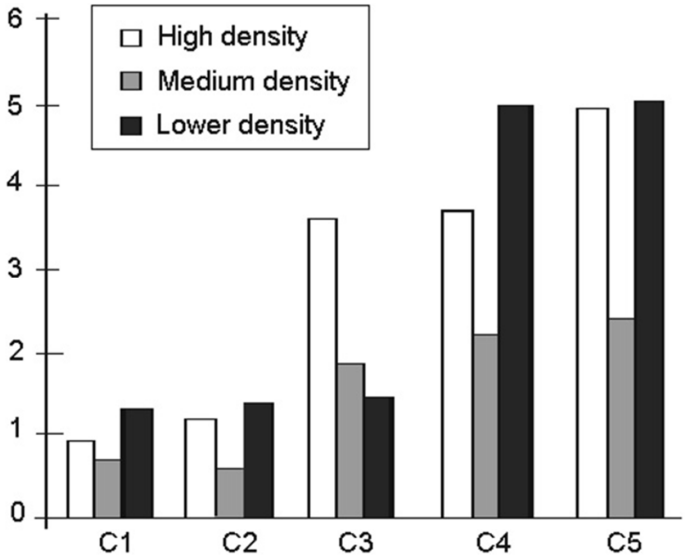


図 2.1: 初期整数緩和問題とカット挿入後の整数緩和問題の目的関数値の平均ギャップ ([69] から引用)

ここで，V は頂点の集合，$S_{max} \subset 2^V$ は極大独立集合 (互いに辺をもたない極大な独立集合の集合) の集合である．集合被覆問題では適当な初期極大独立集合から開始し，ＬＰ緩和問題から求められる双対変数を使い，ヒューリスティック探索によって新たに集合に加えるべき極大集合（これは，目的関数値を下げることが期待されている）を求める．さらに，論文では上記の問題記述の整数緩和問題に対して極大集合間の重複した頂点を利用したカットを考案した．そして，カットによって無駄な分枝を避ける分枝カット法を実装している．

図 2.2 に実行結果を示した．この図では，各問題でランダムに生成したグラフについての彩色数の平均，初期整数緩和問題での目的関数の平均，カットを挿入しない場合，および挿入した場合の平均計算時間が示されている．カットを用いることによって，分枝カット木のノード数が減っているが，計算時間が増えていることが報告されている．

2.2.3　制約プログラミング

制約プログラミングは，ドメイン変数を用いて問題を記述する手法である．ドメイン変数とは，ある集合の要素をとる変数である．制約プログラミングに関する汎用ソルバーは，ある制約を満たす解を求めるような判定問題やある式を最小化する最適化問題を解けるのが普通である．本節では [4] による問題記述を示す (便宜上，一部変更している)．

$$\begin{array}{ll} \text{minimize} & max(L_1, L_2 \ldots, L_n) \\ \text{s.t.} & all\_different(C), \forall C \text{ は極大クリーク}. \end{array} \tag{2.3}$$

			No Cuts		With cuts			
Graph	χ	χ_f	Time	Nodes	Time	Nodes	Cuts	Back.
rand_70_0.3	7.88	6.83	489(9)	2020	655(8)	821	2.11	29
rand_70_0.5	11.8	10.7	41.3	402	55.8	250	5.34	26.2
rand_70_0.7	17.1	16.3	12.9	7.4	13.1	3.4	5.47	0.4
rand_80_0.3	8.1	7.35	225	4438	20.1(9)	2.55	0.35	0.11
rand_80_0.5	12.7	11.6	178(9)	1197	258(9)	833	5.53	75.4
rand_80_0.7	19	17.9	23.9	151	27.2	68.2	9.17	9.6
rand_85_0.3	8.5	7.63	22.5(2)	1	22.4(2)	1	0	0
rand_85_0.5	13	12.0	57.2(9)	204	86.6(9)	139	6.03	11.1
rand_85_0.7	19.8	18.7	37.1	315	37.1	124	7.75	24.6
rand_90_0.3	9	7.94	239(6)	130	264(6)	125	0.57	1.33
rand_90_0.5	13.8	12.6	160(5)	642	204(5)	377	5.6	33.4
rand_90_0.7	20.5	19.3	78.3	801	85.3	313	9.25	60.3
g_300_0.1	10.4	10.4	11	1	11.3	1	0	0
g_300_0.5	80.5	80.3	348	121	380	123	2.41	0
g_300_0.9	206	206	487	1	487	1	0	0
gr_300_0.1	71.7	71.3	76.7	5.8	76.3	5.8	0.1	0.1
gr_300_0.5	7.1	6.82	56	70	53.3	30.6	0.63	2.2
gr_300_0.9	3.8	3.67	31.1	1	31.1	1	0	0
queen8_8	9	8.44	7.45	1	7.33	1	0	0
queen8_9	9	9	–	–	–	–	–	–
queen9_9	10	9	78.7	55	89.2	31	10.1	4
myciel_4	5	3.24	1.35	659	1.88	339	2.79	88

図 2.2: 集合被覆問題によるカットの効果 ([38] から引用)

この問題記述において，極大クリークは隣接グラフにおける完全グラフである．この論文はアプリケーションに関する論文であるため，ベンチマークを使った評価は行われていない．制約プログラミングソルバーは現在でも活発に研究が行われているが，現在では制約プログラミングによる手法によるグラフ彩色問題の解法についてポジティブな意見はない [36]．これは，よい下界値を計算したり，最適解へ向かった探索を行うメカニズムがないことが原因となっている．

2.2.4 SAT

SAT は制約プログラミングにおいて変数のドメインを 2 値に限定したものととらえることができる．記述できるのが論理式に限られるので，活発に研究が行われた結果，現在の SAT ソルバーの性能はかなり高い．本節では [32] によるアプローチを紹介する．SAT におけるグラフ彩色問題は 2.2.1 節でのべた問題記述と同様の問題記述が行われる．

$$\begin{aligned} &\neg v_c \vee \neg u_c, \forall (v,c) \in E \\ &v_0 \vee v_1 \vee \ldots v_{K-1}, \forall v \in V \end{aligned} \tag{2.4}$$

ここで，V は頂点，E は隣接辺集合，$c = 0, 1, \ldots, K-1$ は K 色の色を表している．v_c は頂点 v の色が c となるとき真となる論理変数である．SAT の場合は，他の形式と異なり，このほかにも頂点がある色になることを 2 進数で表現する方法も考えられている．色が L ビッ

トで表現可能ならば，エッジは次のように表現できる．

$$
\begin{aligned}
&v_0 \vee v_1 \vee \ldots v_{L-1} \vee u_0 \vee u_1 \vee \ldots u_{L-1} \\
&\neg v_0 \vee v_1 \vee \ldots v_{L-1} \vee \neg u_0 \vee u_1 \vee \ldots u_{L-1} \\
&\ldots \\
&\neg v_0 \vee \neg v_1 \vee \ldots \neg v_{L-1} \vee \neg u_0 \vee \neg u_1 \vee \ldots \neg u_{L-1}
\end{aligned}
\tag{2.5}
$$

SAT は判定問題であるので直接最適化問題を解くことはできない．そのため，繰り返し判定問題を解くことで厳密解を求める．具体的には，色の数の初期値を最大クリークのサイズにし，SAT ソルバーが充足可能と判定した場合は厳密解が見つかったことになる．充足不可能の場合は色の数を増やして，再度 SAT ソルバーで解を求める．論文では，特に充足可能な解が存在しない場合の計算を効率的に行うため，対称性除去の規則を導入した．また，３種類の SAT ソルバーで計算時間を比較した．

図 2.3 にその一つの例を示す．SAT の場合，最大クリークから彩色数が見つかるまでの問題は充足不可能なので UNSAT，彩色数についての問題は充足可能なので SAT である．tr は上記の問題記述の論理変数を使った場合，xg は各状態について色を２進数で表現した場合，xe は２進表現ではあるが，排他的論理和を表す変数 x_{uvb} を導入し，次の論理式によってエッジの条件を表す．

$$
\begin{aligned}
&\left.
\begin{aligned}
&u_b \vee v_b \vee \neg x_{uvb}, \forall (u,v) \in E \\
&\neg u_b \vee \neg v_b \vee \neg x_{uvb} \\
&\neg u_b \vee v_b \vee x_{uvb} \\
&u_b \vee \neg v_b \vee x_{uvb}
\end{aligned}
\right\}, \forall (u,v) \in E, \forall b \in \{0,1,\ldots,L-1\} \\
&x_{uv0} \vee x_{uv1} \vee \cdots \vee x_{uv(L-1)}, \forall (u,v) \in E
\end{aligned}
\tag{2.6}
$$

？は計算時間がタイムアウトしたことを表している．この実験の評価では，ほとんどの場合 tr が高速であるが，xg,xe が高速である場合も存在することを指摘している．また，SAT ソルバーによっては xg,xe 表現で効率が悪い場合があった．

2.2.5　解法選択の指針

本節では，グラフ彩色問題についての各種のアプローチを述べた．これらのアプローチはどれも数学的には同じであるが，ソルバーによってその表現方法が異なる．これほど多くの研究がある分野は珍しいともいえるが，各ソルバーに関してどのような問題が苦手で，どのような問題が得意であるかといった評価は，筆者の知る限りでは，行われていない．もし，問題ごとの効率性を評価できるのであれば，グラフ彩色問題の個々の問題について，どの方法が適しているのかというような指針を立てられるのではないかと考えられる．

Chaff runs with *b2* symmetry-breaking heuristic, critical color numbers.

	no. of	Unsat CPU secs. by Encodings			no. of	Sat CPU secs. by Encodings		
Graph	colors	tr	xg	xe	colors	tr	xg	xe
abb313GPIA	8	81647	8	50305+?	9	1772	15412+?	30638+?
ash331GPIA	3	0	0	0	4	0	0	17
ash608GPIA	3	0	0	0	4	0	1	235
ash958GPIA	3	0	0	1	4	0	4	733
fpsol2.i.1	64	3	12	9	65	18000+?	17	16
fpsol2.i.2	29	1	155	2089	30	1	5	11
fpsol2.i.3	29	1	5	2366	30	1	5	7
inithx.i.1	53	3	18	44	54	7200+?	36	1384
inithx.i.2	30	1	8	34	31	1	13	2063
inithx.i.3	30	1	7	46	31	1	13	7053
mulsol.i.1	48	1	4	1	49	1	4	6
mulsol.i.2	30	0	3	7200+?	31	0	3	5
mulsol.i.3	30	0	10	4363+?	31	0	3	5
mulsol.i.4	30	0	16	3651+?	31	0	4	6
mulsol.i.5	30	0	1	5	31	0	3	1
myciel5	5	20	8	6	6	0	0	0
myciel6	6	3978+?	3253+?	2247+?	7	0	0	0
myciel7	5	22	18000+?	50	8	0	0	0
school1_nsh	13	1	54	1702	14	1	1253	156
school1_sh	13	1	4	294	14	1	23	111
will199GPIA	6	0	0	1	7	0	2	158
zeroin.i.1	48	1	4	2	49	1	4	1
zeroin.i.2	29	0	1	1	30	0	1	8
zeroin.i.3	29	0	1	1	30	0	1	9
DSJC125.1	4	0	0	0	5	0	1	1
DSJC125.5	12	1176	823	3768	19	18000+?	6269	18000+?
DSJC125.9	37	6529	18000+?	18000+?	46	18000+?	10133+?	18000+?

図 2.3: SAT ソルバー Chaff を用いた各記述法についての計算時間 ([32] から引用)

グラフ彩色問題がそうであったように，このような解法の比較を行うには各問題ごとにベストであるような手法を取り入れる必要がある．本論文では，特定の問題に関して研究するのではなく，複数の問題に対して現状でベストである解法を提案する．そして，その結果から汎用ソルバーを選択する指針を構築したいと考えている．

第3章　最小無矛盾決定性有限オートマトン生成問題

3.1　はじめに

DFA(Deterministic finite state automaton, 決定性有限状態オートマトン) は，状態と状態間の遷移を使った文字列集合 (または，記号列集合) の表現方法であり，最後に遷移した状態が出力するラベルによって文字列を分類する．本章は，与えられたラベル付き文字列集合を分類し，状態数が最小である DFA(以下，**最小無矛盾 DFA**: Minimum-consistent DFA) を効率よく生成する問題に関するものである．最小無矛盾 DFA はラベル付き文字列集合を生成する DFA を推定したものであり，ラベル付き文字列集合に含まれない文字列のラベルも決定することができる．最小無矛盾 DFA は直ちにラベル付き文字列を生成する真の DFA となるわけではないが，文法学習における極限同定 [33] を与える基本的な概念であり，PAC-identification においても，事例の最も単純な表現であるとみなされる [8]．そのため，この問題は文法推論における重要な問題の一つであり，正規文法の帰納的学習 [93] をはじめ，大規模ハードウェア・ソフトウェアシステムに対するモデル検証 [15] など多くの応用例が存在する．

特定の文字列長まで全てのラベルがわかっているラベル付き文字列集合については，多項式時間アルゴリズムを使ってその最小無矛盾 DFA を求めることができる [81]．しかし，特定の状態数の無矛盾 DFA を求める問題は NP 困難であり [33]，その最も小さい状態数を持つ最小無矛盾 DFA を求める問題も NP 困難な問題であることが知られている．

このとき，本章は正負のような二値のラベルが付与されたラベル付き文字列集合に対する最小無矛盾 DFA の効率的な生成を目的とする．この目的のため，本章では MILP によるアプローチと SAT によるアプローチを示す．

本章の構成は次のとおりである．3.2 節では関連研究について述べる．3.3 節では最小無矛盾 DFA 問題を定義する．

3.4 節では，MILP によるアプローチを示す．MILP によるアプローチでは，状態のマージによる記述方法とグラフ彩色問題による記述方法が考えられる．状態マージによる方法では，すべての可能な無矛盾 DFA より最小無矛盾 DFA を求める一つの問題記述が使われる．この記述方法は問題記述に必要となる変数の数および制約式の数において現実的な方法でないことを示す．次に，グラフ彩色問題による問題記述方法を示す．この方法では，ヒューリ

スティック探索により見つかった最小無矛盾 DFA の状態数を上限とした一つの問題記述が使われる．この方法は，整数変数を用いることで現実的なサイズで問題記述が出来ることを示す．しかしながら，実験の結果，初期の LP 解（すべての整数変数を整数緩和したときの LP 解）の目的関数値が最小無矛盾 DFA の状態数に近くないため，定式化としては適切でないことがわかった．そのため，グラフ彩色問題による記述方法では，カットを導入することで効率化を図った．

3.5 節では，SAT によるアプローチを示す．現在，論理式が充足可能かどうかを判定する SAT(Satisfiability problem, 充足可能性問題) を使った解法がもっとも効率的な最小無矛盾 DFA 問題に対する解法であると考えられる [41]．このアプローチでは，ある状態数の最小無矛盾 DFA が存在するかどうかという判定問題を，状態をグラフの頂点と見なしたグラフ彩色問題 (Graph coloring problem) に帰着して記述し，SAT ソルバーを使って解く．本節では，MILP によって求まった最大クリークを用いた手法 (3.5.2 節) とモデルの改良による手法 (3.5.3 節) について述べる．

公開されているベンチマーク問題 ([59]) を対象に行われた評価では，SAT によるアプローチによって非常に効率的に最小無矛盾 DFA が求められることが示されている．これは手法の改良による所が大きいが，ベンチマーク問題が解きやすい性質を持っている点も理由の一つであると考えられる．すなわち，ある長さのラベル付き文字列について先頭から任意の長さの部分文字列のラベルも与えられていることが解を求めやすくしている．そこで本章では，このような性質を備えていない問題に対しても提案手法の評価を行う．

3.2　関連研究

最小無矛盾 DFA を生成するアルゴリズムは，近似か厳密かにより研究の方向性は大きく二つに分類される．近似手法の場合，求められた DFA の状態数が最小であるかの証明は行われないが，厳密手法の場合は求められた DFA の状態数よりも小さい無矛盾 DFA が存在しないことが保証される．いずれの手法でも，ラベル付き文字列集合から直接生成される木構造 DFA である APTA(Augmented prefix tree acceptor, 3.3 を参照) から無矛盾 DFA は求められる．

まず，近似手法について述べる．Blue-frindge アルゴリズム [16] は，APTA のルートに近い状態から状態マージ (二つの状態を任意の文字列を等しく判別する等価な状態とすること) を繰り返していき，無矛盾 DFA を求める．EDSM(Evidence-Driven State Merge)[59] では，適切にマージする状態を選択するヒューリスティックスを使って無矛盾 DFA を求める．Lambeau らはさらに状態の同値性に関する領域依存の知識を付与することで，高品質の DFA が得られることを示した [58]．これらは多項式時間のアルゴリズムである．また，進化型計算に代表される最適化手法を使った研究も多く行われた (例えば [26])．

これら近似手法に対して，Oliveira らは厳密に最小無矛盾 DFA を効率良く求める手法を提案した [78]．彼らは APTA の各状態をドメイン変数によって表現した．このドメイン変数は，最小の状態数を n と仮定したとき，0 から $n-1$ までの整数値を取る．同じ値を持つ状態は状態マージされたとみなすが，このとき各状態に振られた数字はグラフ彩色問題における**色**を表現している．そして，縦型探索により各状態に色を割り当てる．このとき，制約充足問題 (Constraint satisfaction problem) を効率良く解くための基本的なテクニックである conflict diagnosis を実装し，効率的なバックトラックにより高速化を図った．

さらに，近年，効率化が著しい汎用ソルバーを使った研究も行われている (混合整数線形計画法 [47], SAT[41])．これらの研究は線形関数や論理式などで APTA と無矛盾 DFA の間の数学的関係を記述する．その記述方法は二種類ある．ひとつは，APTA における 2 つの状態がマージするかしないかを変数として表現する．もう一つは，APTA の各状態に色を付与するグラフ彩色問題に帰着して表現する．これまで行われた研究ではグラフ彩色問題に帰着する方が記述量の点から優位であることが示されている．そのため，本研究では最小無矛盾 DFA 問題をグラフ彩色問題に帰着して解く．

3.3 定義

本章では，参考文献 [15] における記法に従って，本章で用いる用語を定義する．文字集合を Σ により表記する．Σ 上の全文字列集合は Σ^* と表記する．文字列 $u, v \in \Sigma^*$ に対して，uv は二つの文字列の連結を表し，$u^0 \equiv \lambda$，$u^n \equiv uu^{n-1}$ と定義する．ここで，λ は長さが 0 の文字を意味する．集合 S に対して $|S|$ は集合の大きさを表す．

1. DFA
   DFA を $A = (\Sigma, S, s_0, \delta, F)$ と定義する．ここで，S を状態集合，$s_0 \in S$ を開始状態，$\delta : S \times \Sigma \to S \cup \{\emptyset\}$ を遷移関数，$F \subseteq S$ を受理状態集合と呼ぶ．$\emptyset$ は S の要素ではない特別な状態であり, 便宜上，任意の $a \in \Sigma$ について $\delta(\emptyset, a) = \emptyset$ と定義する．文字列 u について，$\delta(s_0, u) \in F$ であれば，文字列 u は受理，それ以外は不受理と呼ぶ．ここで，文字列 $u = a_1 a_2 \dots a_n$ について，$\delta(s_0, u) \equiv \delta(\delta(\dots\delta(s_0, a_1)\dots, a_{n-1}), a_n)$ である．そして，文字列集合 $L(A) = \{u \in \Sigma^* | \delta(s_0, u) \in F\}$ を正規言語と呼ぶ．
2. 3DFA
   判別不能の文字列を扱えるように一般化した DFA である 3 値 DFA(Three-valued DFA, 以下，**3DFA**) について述べる．3DFA を $C = (\Sigma, S, s_0, \delta, Acc, Rej, Dont)$ と定義する．ここで，$Acc, Rej, Dont \subseteq S$ は互いに共通の状態を持たない状態部分集合であり，$Acc \cup Rej \cup Dont = S$ である．$u \in \Sigma^*$ について，$\delta(s_0, u) \in Acc$ の場合，u は受理，$\delta(s_0, u) \in Rej$ の場合，u は不受理という．$\delta(s_0, u) \in Dont \cup \{\emptyset\}$ の場合，u は判別不能という．この 3DFAC に対して 3DFA$D = (\Sigma, S', s'_0, \delta', Acc', Rej', Dont')$ が**無矛盾**で

あるとは，任意の文字列 $u^* \in \Sigma^*$ について，$\delta(s_0, u) \in Acc$ ならば $\delta'(s'_0, u) \in Acc'$ であり，かつ，$\delta(s_0, u) \in Rej$ ならば $\delta'(s'_0, u) \in Rej'$ であることと定義する．つまり，C で受理される文字列は D でも受理され，C で不受理の文字列は D でも不受理である．

3. APTA
   3DFAC が木構造，すなわち，任意の異なる文字列 $u, v \in \Sigma^*$ に対して, $\delta(s_0, u) \in S$ かつ $\delta(s_0, v) \in S$ ならば $\delta(s_0, u) \neq \delta(s_0, v)$ であるとき，3DFA を APTA と呼ぶ．
   APTA は，二値のラベル付き文字列集合が与えられれば，一意に生成できる 3DFA である．本章で求める最小無矛盾 DFA とは，この APTA に無矛盾な状態数が最小の 3DFA である．このため，求められた 3DFA の状態数は必ず APTA の状態数以下のものとなる．なお，3.1 節で述べたベンチマーク問題では $Dont$ が空集合である APTA が与えられている．
   図 3.1 に受理される文字列集合 E^+ と受理されない文字列集合 E^- で作られる APTA を示す．この例では，判別不能となる状態を全て受理状態とした場合と不受理状態にした場合の APTA も示す．また，最小無矛盾 DFA とこの場合の APTA の各状態のラベルを示す．図 3.1 が示すように，最小無矛盾 DFA によって APTA の判別不能状態に受理状態か不受理状態のラベルが割り当たることになる．

3.4　最小無矛盾 DFA 問題に対する MILP アプローチ

本節では MILP を用いて，最小無矛盾 DFA 問題を記述する．最小無矛盾 DFA 問題は，状態を分類するという意味において，グラフ彩色問題に非常によく似た問題であり，グラフ彩色問題の特殊なケースと考えてもよい．グラフ彩色問題において，状態の色を表現する方法は大きく分けて次の 3 種類が考えられる．

1. 状態を特定の色に割り当てる方法 [18]
2. 二つの状態を同じ色に割り当てるかどうかを変数で表現する方法 [79]
3. ある状態の集まりを一つの変数で表現する方法 [68]

この中で，3 番目の表現方法は集合分割問題としてグラフ彩色問題を記述する方法であり，列生成法と共に用いられる．集合分割問題とは，一つの変数が同じ色となる状態集合を表現しており，各色について一つの状態集合を選ぶ問題である．最小無矛盾 DFA 問題について

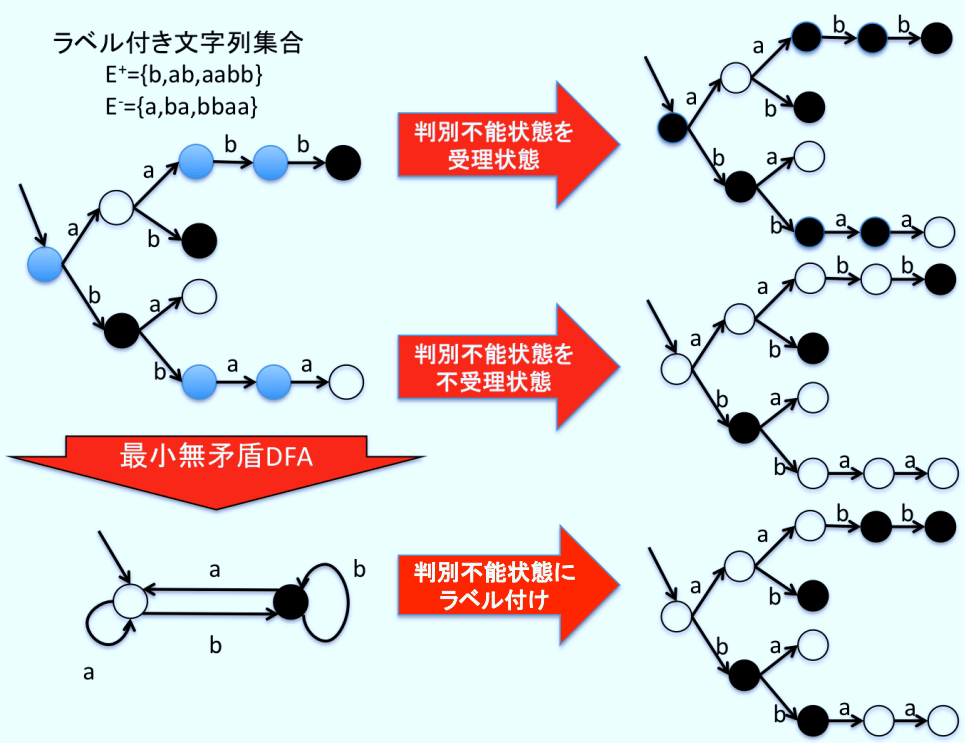


図 3.1: ラベル付き文字列集合とその APTA および無矛盾 DFA(黒丸は受理状態，白丸は不受理状態，青丸は判別不能状態)

は，この集合分割問題は次のように書くことができる．

$$
\begin{aligned}
\text{minimize} \quad & \sum_{t \in T} p_t \\
\text{s.t.} \quad & \sum_{s \in \forall t \in T} p_t = 1, \forall s \in S \\
& \sum_{\delta(t,a) \subseteq t' \in T} p_{t'} \geq p_t, \forall a \in \Sigma, \forall t \in T \\
& p_t \in \{0,1\}, \forall t \in T
\end{aligned}
\tag{3.1}
$$

ここで，$T \in 2^S$ はマージ可能な状態集合の集合である．変数 p_t はマージ可能な状態集合 $t \in T$ が最小無矛盾 DFA において一つの色となる状態集合である場合 1 となり，そうでない場合 0 となる変数とする．そのため，目的関数は色の最小化を意味する．1 番目の制約式は，状態 $s \in S$ を含む状態集合が一つ選ばれることを意味する．2 番目の制約式は，状態集合 $t \in T$ が最小無矛盾 DFA の一つの色を形成する状態集合の場合，その状態集合の各状態がある文字 $a \in \Sigma$ で遷移することで構成される状態集合 $\delta(t,a)$ を部分集合に持つ状態集合 $t' \in T$ の一つも最小無矛盾 DFA の状態集合になっていなければならないことを示す．

この集合分割問題あるいは複数の色の集合で一つの状態の共有を認める集合被覆問題は，グラフ彩色問題において成功した方法である．このとき，T をすべて列挙することは不可能なため，列生成法を用いている．列生成法では，上記の問題の整数緩和問題から導き出される双対変数の値から，整数緩和問題の目的関数値を下げる新たな変数が導きだされる．この新たに考慮すべき状態集合に対応する変数を導き出す問題は，列生成子問題と呼ばれる．グラフ彩色問題の場合，この列生成子問題が解きやすい問題となって現れる．これに対して，最小無矛盾 DFA 問題の場合，２番目の制約式の存在から，列生成子問題において一つの状態集合を求めるだけでは十分でなく，その状態集合が一つの色を生成する無矛盾 DFA に関するすべての状態集合に関する変数が生成される必要がある．これは結局，列生成子問題においても最小無矛盾 DFA を求めるのに匹敵する問題を解くことが要求されるため，困難な問題となってしまう．

以上の理由から，本節では 1 番目の方法と 2 番目の方法に関して，MILP によって最小無矛盾 DFA 問題を記述し，２番目の方法に関して予備的な実験結果を示す．

本節の問題の記述では，$APTA = \{\Sigma, S, s_0, \delta, Acc, Rej, Dont\}$ が与えられるとする．状態集合 $S = \{0, 1, \ldots, |S| - 1\}$ とする．そして，APTA から得られる隣接グラフのエッジ集合を E とする．

3.4.1　状態マージによる方法

ここでは，変数 $p_{s,t} \in \{0, 1\}, s, t \in S$ によって，状態 s と t がマージするかどうかを表現する．

$$p_{s,t} = \begin{cases} 1 & \text{if } s \text{ と } t \text{ がマージ} \\ 0 & \text{上記以外} \end{cases} \tag{3.2}$$

変数 $z_s \in \{0, 1\}, s \in S$ は無矛盾 DFA の状態数を数えるために使い，s にマージする変数の中で s が最も小さい場合は 1，そうでない場合は 0 とする．これらの変数を使って，最小無矛盾 DFA を求める MILP は次のように記述される．

$$
\begin{aligned}
\text{minimize} \quad & \sum_{s \in S} z_s \\
\text{s.t.} \quad & \sum_{t=0}^{s-1} p_{s,t} + z_s \geq 1, \forall s \in S \\
& p_{s,t} \leq p_{\delta(s,a),\delta(t,a)}, \forall s,t \in S, \forall a \in \Sigma, \exists \delta(s,a), \delta(t,a) \in S \\
& p_{s,t} + p_{t,u} + p_{u,s} \neq 2, \forall s,t,u \in S \\
& p_{s,t} = 0, \forall (s,t) \in E \\
& p_{s,t} \in \{0,1\}, \forall s,t \in S \\
& z_s \in \{0,1\}, \forall s \in S
\end{aligned}
\tag{3.3}
$$

第 1 制約条件は，色の数を数えるために使われる．もし，状態 s が s より小さい状態とマージしない場合は，s は同じ色に割り当てられる状態の中で最小の数の状態となる．この制約条件では，最小の数の状態を数えることで状態数を数えている．第 2 制約条件は状態 s，t がマージするときは，同じ文字 a で遷移する状態 $\delta(s,a)$ と $\delta(t,a)$ もマージすることを示す．第 3 制約条件は三つの状態間でマージ数の合計が 2 とならない，すなわち二組の状態がマージして残りの一組の状態がマージしないとうことが起きないことを示している．この条件は実際の定式化では次のように書くことができる．

$$
\begin{aligned}
& +p_{s,t} + p_{t,u} - p_{u,s} \leq 1 \\
& +p_{s,t} - p_{t,u} + p_{u,s} \leq 1 \\
& -p_{s,t} + p_{t,u} + p_{u,s} \leq 1
\end{aligned}
\tag{3.4}
$$

第 4 制約条件は隣接グラフのエッジである一組の状態はマージできないことを示している．この問題記述では，変数の数は $O(|S|^2)$，制約式の数は $O(|S|^3)$ となり，状態数に対して問題の大きさが非常に大きいという欠点を持っており，現実的に実現困難な問題記述となっている．

3.4.2 グラフ彩色問題による定式化

APTA と無矛盾な DFA は,$(\Sigma, K, k_0, \delta^K, Acc^K, Rej^K, Dont^K)$ によって与えられるとする．ここでは，$x_{s,i} \in \{0,1\}, s \in S, i \in K$ は 1 のとき状態 s に色 i が割り当たるとする．$y_{a,i,j} \in \{0,1\}, a \in \Sigma, i,j \in K$ は 1 のとき $\delta^K(i,a) = j$ とする．$z_i \in \{0,1\}, i \in K$ は 1 のとき $\delta(k_0,u) = i$ となる $u \in \Sigma^*$ が存在するとする．$K = \{0,1,\ldots,k-1\}$ のとき，最小無矛盾 DFA 問題を求める問題は次のように記述できる．

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=0}^{k-1} z_i \\
\text{s.t.} \quad & \sum_{i=0}^{k-1} x_{s,i} = 1, \forall s \in S \\
& \sum_{j=0}^{k-1} y_{a,i,j} = 1, \forall a \in \Sigma, \forall i \in K \\
& x_{s,i} + x_{\delta(s,a),j} - y_{a,i,j} \leq 1, \forall s \in S, \forall a \in \Sigma, \exists \delta(s,a) \in S, \forall i,j \in K \\
& x_{s,i} - x_{\delta(s,a),j} + y_{a,i,j} \leq 1, \forall s \in S, \forall a \in \Sigma, \exists \delta(s,a) \in S, \forall i,j \in K \\
& x_{s,i} + x_{t,i} \leq z_i, \forall (s,t) \in E, \forall i \in K \\
& x_{s,i} \in \{0,1\}, \forall s \in S, \forall i \in K \\
& y_{a,i,j} \in \{0,1\}, \forall a \in \Sigma, \forall i,j \in K \\
& z_i \in \{0,1\}, \forall i \in K
\end{aligned}
\tag{3.5}
$$

この問題記述において，第1制約条件は各状態に色を割り当てることを意味する．第2制約条件は各色における遷移の決定性を表している．つまり，各色においてある文字に対する次の状態は一意に決定される．第3制約条件，第4制約条件はAPTAにおける遷移とDFAにおける遷移の対応付けを行なっている．第5制約条件はエッジの両端の状態には同じ色を割り当てない条件であり，いわゆるグラフ彩色問題制約と呼ばれる．この問題記述では，変数の数は$O(|S||K|)$となり，制約条件の数は$O(|S||\Sigma||K|^2)$となる．

この問題記述に対して，整数変数を$x_s \in K$および$y_{a,i} \in K$を導入することで制約式の数を減らすことができる．ここで，x_sは色を表す整数変数であり，$y_{a,i}$は状態iの文字aに対する遷移先の状態を表している．このとき，次の問題記述が得られる．

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=0}^{k-1} z_i \\
\text{s.t.} \quad & \sum_{i=0}^{k-1} x_{s,i} = 1, \forall s \in S \\
& x_s = \sum_{i=0}^{k-1} i x_{s,i}, \forall s \in S \\
& (k-1)x_{s,i} + x_{\delta(s,a)} - y_{a,i} \leq k-1, \forall s \in S, \forall a \in \Sigma, \exists \delta(s,a) \in S, \forall i \in K \\
& (k-1)x_{s,i} - x_{\delta(s,a)} + y_{a,i} \leq k-1, \forall s \in S, \forall a \in \Sigma, \exists \delta(s,a) \in S, \forall i \in K \\
& x_{s,i} + x_{t,i} \leq z_i, \forall (s,t) \in E, \forall i \in K \\
& x_{s,i} \in \{0,1\}, \forall s \in S, \forall i \in K \\
& x_s \in K, \forall s \in S \\
& y_{a,i} \in K, \forall a \in \Sigma, \forall i \in K \\
& z_i \in \{0,1\}, \forall i \in K
\end{aligned}
\tag{3.6}
$$

第 2 制約条件，第 3 制約条件は $x_{s,i} = 1$ のとき，$x_{\delta s,a} = y_{a,i}$ とする式であり，APTA と無矛盾 DFA の遷移の対応付けを行なっている．この式は”big-M” 形式の制約式であり，3.5 式の第 3 制約条件，第 4 制約条件と比べて変数の拘束力が弱いと考えられる．しかしながら，MILP を解く際の一回あたりの LP の実行時間は短くなるため，実行時間と制約式の拘束力はこの場合トレードオフの関係にある．この問題記述の場合，変数の数は $O(|S||K|)$ で先の問題記述を同じであるが，制約条件数は $O(|S||\Sigma||K|)$ となる．

この節の問題記述の場合は，あらかじめ色の数 k を決めておく必要がある．一般にこの k は最小無矛盾 DFA の状態数以上の数を指定しなければならないが，なるべく小さい数を決めておきたい．そのため，ヒューリスティック探索に最小無矛盾 DFA を予め求めておき，その数を使うことができる．本節で述べる実験においては，APTA において開始状態に近い状態から状態を並べ，先頭の状態から順番に，前にある状態に対してマージを行なっていき，最小無矛盾 DFA を求めるというヒューリスティック探索を用いている．

3.4.3 カット

3.4.2 節ではグラフ彩色問題に帰着した MILP による問題記述について説明した．3.4.1 節に比べると，実験的に効率的な解法であることが確かめられている [47]．しかし，分枝限定法における分枝数が非常に多いものとなっている．これは問題記述が実行可能解を求めるのに適していないことが原因であり，各分枝で実行される LP において整数解が得られていない．このように，分枝における LP において整数解が得られにくい場合は，単に分枝するのではなく，整数解を得やすくするためのカットを入れることが効果的である．効果的なカッ

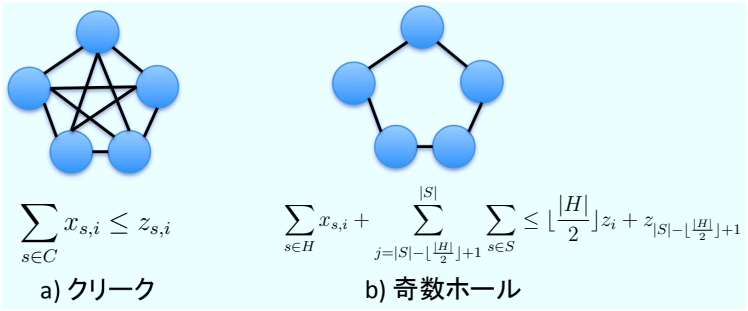


図 3.2: グラフ彩色問題における基本的なカット

トを入れることが出来れば分枝する数を減らすことが可能になる．最小無矛盾 DFA 問題の場合はグラフ彩色問題のバリエーションの一つに問題記述を帰着できるので，グラフ彩色問題で考えられているカット [69],[13] はそのまま利用することが出来る．例えば，クリークカットのような隣接グラフにおけるクリークのメンバーが同じ色に割り当てられないことに基づくカットやホールを構成する状態のメンバーに関する同じ色に割り当てられる状態数の制約 (図 3.2) など，そのまま利用することが出来る．

3.4.4　評価

筆者らは，事例数が少なく，短い文字列で作られる APTA から 3.4.2 節で述べた問題記述を使い，CPLEX で MILP ソルバーを実行しているときの分枝時に基本的なカット (クリーク，ホール，アンチホール，安定集合) を問題記述に挿入する分枝カット法を実装した．導入したカットの制約式を次に示す．また，隣接グラフにおけるカットの形状を図 3.3 に示す．ここで，e) のマージに関しては条件が成立する APTA の例を示した．

クリーク 隣接グラフにおける完全部分グラフをクリークと呼ぶ．クリーク $C \subset S$ について，次の式が成り立つ．

$$\sum_{s\in C} x_{s,i} \leq 1, i \in K \tag{3.7}$$

独立集合 $I \subset S, \forall i,j \in I, (i,j) \notin E$ となる I を独立集合と呼ぶ．I が最大独立集合 (状態数が最大の独立集合) とすると，任意の $S' \subset S$ について次の式が成り立つ．

$$\sum_{s\in S'} x_{s,i} \leq |I|, i \in K \tag{3.8}$$

ホール $H \subset S$ において，$|H|$ が奇数である場合を考える．H がホールであるとき，次の式が成り立つ．

$$\sum_{s\in H} x_{s,i} \leq \lfloor\frac{|H|}{2}\rfloor \tag{3.9}$$

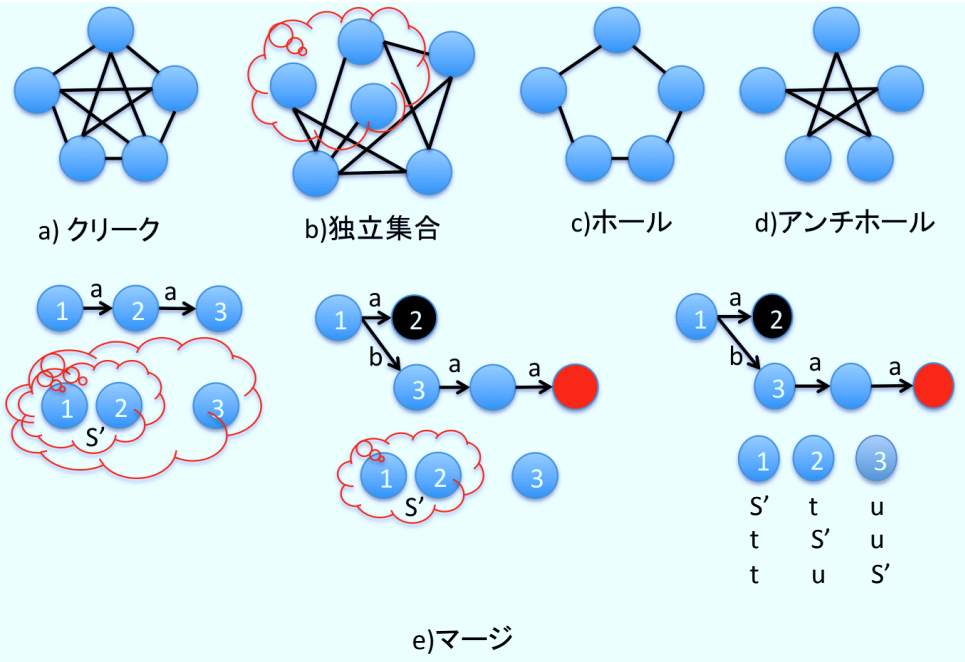


図 3.3: 無矛盾 DFA 問題のためのカット

アンチホール $A \subset S$ がアンチホール (ホールのエッジに関する補集合) であるとき，次の式が成り立つ

$$\sum_{s \in A} x_{s,i} \leq 2, i \in K \tag{3.10}$$

マージ すべて同じ色に彩色可能な $S' \subset S$ について考える．このとき，$t \in S - S'$ も同じ色になる場合，違う色になる場合，$t, u \in S - S'$ が異なる色になる場合について，それぞれ次の式が成り立つ．

$$\begin{aligned} \sum_{s \in S'} x_{s,i} - x_{t,i} &\leq |S'| - 1 \\ \sum_{s \in S'} x_{s,i} + x_{t,i} &\leq |S'| \\ \sum_{s \in S'} x_{s,i} + x_{t,i} + x_{u,i} &\leq |S'| + 1 \end{aligned} \tag{3.11}$$

この制約条件は無矛盾 DFA に関して独特のカットであり，グラフ彩色問題では成立しない．

これらのカットが実際に問題を解く過程でどの程度現れるかを実験的に調査した．実験では同じ条件でランダムに事例集合を生成する．CPLEX における分枝限定法の各ノードにおいてカットの制約違反を発見し，カットを挿入する．実験において，実行時間を最大３０分

表 3.1: カットの生成個数：APTA 状態数:243.4　DFA 状態数:10(5 回の平均)

	なし	+クリーク	+独立集合	+ホール	+アンチホール	+マージ
クリーク		551	580.6	569.6	578.6	673.6
独立集合			22.4	18.6	23	4
ホール				106.8	96.8	433.2
アンチホール					4	130.8
マージ						1863.8
LP 回数	237.4	54	105.2	83.4	93.2	85.6
Lower	3.616	3.438	3.438	3.438	3.438	3.438
最大目的関数値	7.2	5.625	5.85	6.4	5.253	3.438

表 3.2: カットの生成個数：APTA 状態数:132.2 DFA 状態数:13.2(5 回の平均)

	なし	+クリーク	+独立集合	+ホール	+アンチホール	+マージ
クリーク		2563	2528	2671	1758	8062
独立集合			12.2	24.8	0.8	13.6
ホール				1060	34.4	396.8
アンチホール					15.6	42.8
マージ						172.6
LP 回数	766	158.6	166.4	213.8	28.8	151.8
Lower	6.2	7.55	7.55	7.55	7.35	7.95
最大目的関数値	12.8	12	11.8	10.62	7.35	7.95

にした．実験結果として，同じ条件で各 5 回試行した平均値を示す．表 3.1 と表 3.1 に結果を示す．左から右にカットの種類が増えていく．表からは多くの各カットが挿入されていることがわかる．これにより，カットは MILP での解法の効率化に貢献することが期待できることがわかる．30 分実行した結果では，表 3.1 では下限値を向上する（Lower は分枝前の下界値を示す）ことには貢献していないが，表 3.2 では貢献している結果が出ている．これは，カット生成はコストの高い処理であるため，整数緩和問題をたくさん解けないことが原因だと考えられる．その他に，表 3.2 では最大の目的関数値を下げる，つまり無駄な分枝を抑制している効果が得られている．

しかしながら，MILP によるアプローチは現在のところ次節で述べる SAT によるアプローチにに比べて計算時間の点で劣ると考えている．これは，MILP の分枝限定法において整数解が得られにくく，結果として分枝操作の回数が多くなり，最適性の証明に非常に時間がかかるためである．

3.4.5　まとめ

本節では，MILP による最小無矛盾 DFA の問題記述について述べた．実験結果は MILP によるアプローチが本問題に対してあまり強力ではないことを示している．この理由としては，MILP を解いている過程で，下界値が上がらないことが挙げられる．本節ではグラフ彩色問

題で用いられているものを中心にカットを導入した．カットの導入によって，下界値が向上することが実験的に認められたが，大きな問題が解けるほどは有効でないと考えられる．グラフ彩色問題では MILP によるアプローチがかなり有効であることから，遷移に関する規則が最適値と下界値のギャップを生み出していると考えられる．

3.5 最小無矛盾 DFA 問題に対する SAT アプローチ

本節では，まず，従来手法の説明を行い，二つの方法での改良について述べる．一つは，下界値の改良であり，MILP により最大クリーク問題の厳密解を求めることによる改良である．もう一つは，問題記述に関する改良について述べる．

3.5.1 SAT による従来解法

本章では，SAT を使った Heule[41] らの解法について述べる．本研究はこの解法を基礎として新しい手法を提案しているので，詳しく述べる．

SAT ソルバーは SAT competition などの活動と共に，近年非常に進歩している [87]．SAT ソルバーはすべての論理変数に真偽値を割り当てられるかどうか探索する．割り当てられる場合を充足可能と呼ぶ．厳密に充足可能かどうかを判定する SAT ソルバーの代表的なアルゴリズムとして，Davis-Putnam-Longmann-Loveland(DPLL) アルゴリズムがある [82, 45]．DPLL アルゴリズムは基本的に縦型探索で変数に真偽値を割り当てていく．論理式が偽となった場合は，その原因を求め，バックトラックポイントを探し，探索を続行する．SAT 自体は NP 完全な問題である [20] が，節学習やリスタートなど様々な手法が開発され，ソルバーの高速化に寄与している [74]．

SAT を使って最小無矛盾 DFA を求めるためには，グラフ彩色問題に帰着した問題の定式化 [21] を使う方法が効率的である．図 3.1 では，２状態からなる最小無矛盾 DFA を構成した．最小無矛盾 DFA の状態数を色の数と考えれば，APTA の各状態が２色で色分けされたことになる．

3.5.1.1 無矛盾 DFA の表現

ここで，最小無矛盾 DFA 問題における状態数 k の無矛盾 DFA の判定問題に対応する k 彩色問題を述べる．この場合は，APTA の各状態に $K = \{0, 1, \ldots, k-1\}$ によって定められた色を割りつける．$APTA = (\Sigma, S, s_0, \delta, Acc, Rej, Dont)$ に対して，無矛盾 DFA$(\Sigma, K, k_0, \delta^K, Acc^K, Rej^K, Dont^K)$ を定義する．論理変数 $x_{s,i}, s \in S, i \in K$ は真のと

き，状態 s は色 i を持ち，偽のときその色を持たないことを表現する．このとき，各状態はちょうど一色を持つので，$s \in S$ について次の論理式が成り立つ．

$$x_{s,0} \vee x_{s,1} \vee \cdots \vee x_{s,k-1} \tag{3.12}$$

$$\neg x_{s,i} \vee \neg x_{s,j}, 0 \leq i < j \leq k-1 \tag{3.13}$$

式 3.12 は状態 s について少なくともひとつの色が割り当てられることを表し，式 3.13 は同時に色 i と j が状態 s に割り当てられないことを表している．式 3.12 は必須であるが，式 3.13 は複数の色が割り当てられても適切な色を選ばよいので任意である．しかしながら，このような任意の条件は変数に対する不必要な値の割り当てを抑制する効果があるので導入される．

論理変数 $z_i, i \in K$ は，真のとき色 i は受理状態 $(i \in Acc^K)$，偽のとき不受理状態 $(i \in Rej^K)$ とする．この時，次の論理式が成り立つ．

$$\begin{cases} \neg x_{s,i} \vee z_i & \text{if } s \in Acc \\ \neg x_{s,i} \vee \neg z_i & \text{if } s \in Rej \end{cases} \tag{3.14}$$

この論理式では，第一の式によって s が受理状態のとき s の色が i ならば，i は受理状態になる．第二の式は不受理状態についてである．

論理変数 $y_{a,i,j}, a \in \Sigma, i, j \in K$ は真のとき $j = \delta^K(i, a)$ とする．偽のときはそうでないことを表現する．このとき，式 3.15 および式 3.16 が成り立つ．

$$y_{a,i,j} \vee \neg x_{s,i} \vee \neg x_{\delta(s,a),j} \tag{3.15}$$

$$\neg y_{a,i,j} \vee \neg x_{s,i} \vee x_{\delta(s,a),j} \tag{3.16}$$

式 3.15 は，状態 s の色が i, 状態 $\delta(s, a)$ の色が j ならば，$y_{a,i,j}$ が真になることを表し，次の式 3.17 と共に，色で作られる DFA における遷移の一意性を表現している．式 3.16 は任意の式であるが効率化のために導入される．

3DFA において，特定の文字に対するある状態からの遷移先は高々一つとなるので，次の論理式は必須である．

$$\neg y_{a,i,j_1} \vee \neg y_{a,i,j_2}, 0 \leq j_1 < j_2 \leq k-1 \tag{3.17}$$

この式では，色 i において文字 a に対する遷移先は同時に色 j_1 と j_2 にはならないことを示している．

任意の条件として，各色 i において入力記号 a に対して少なくともひとつの遷移を持つ条件を導入する．式 3.18 は式 3.12 と式 3.15 より自動的に成り立つが，導入することで SAT ソルバーは解を効率よく求めることができる．

$$y_{a,i,0} \vee \cdots \vee y_{a,i,k-1} \tag{3.18}$$

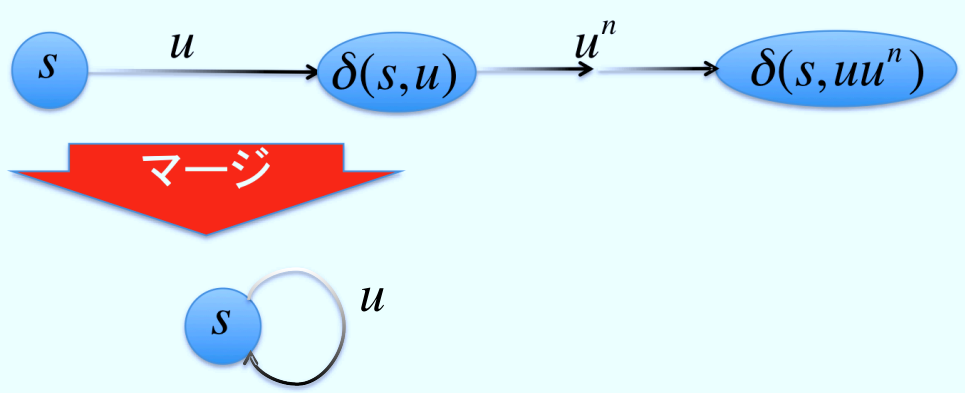


図 3.4: 親子関係にある状態間のマージ

これらのほかに，二つの状態が同じ色にならない条件が導入された．

$$\neg x_{s,i} \vee \neg x_{t,i}, (s,t) \in E, i \in K \tag{3.19}$$

ここで，E はグラフ彩色問題におけるエッジ集合 (同じ色に割り当てられない状態対の集合) である．無矛盾 DFA において，このエッジ集合は $s,t \in S$ について次のように再帰的に求めることができる．ここで求められるグラフ $G =< S, E >$ は隣接グラフと呼ばれる．

$$\begin{array}{ll} (s,t) \in E & \text{if } s \in Acc, t \in Rej \\ (s,t) \in E & \text{if } \exists a \in \Sigma((\delta(s,a),\delta(t,a)) \in E) \\ (s,\delta(s,u)) \in E & \text{if } \exists u \in \Sigma^* \exists n \in \mathbb{N}^+(\\ & \quad (s,\delta(s,uu^n)) \in E) \end{array} \tag{3.20}$$

最初の式は一つの状態が受理状態であり，もう片方が不受理状態であれば，状態対はエッジであることを示す．二番目の式は，共通の文字で遷移する先の状態対がエッジであるときは，遷移元の状態対もエッジであることを示す．三番目の式は，図 3.4 のような場合であり，状態 s と文字列 uu^n に対する状態 $\delta(s,uu^n)$ がエッジならば，s と $\delta(s,u)$ もエッジであることを示す．本章では，このような状態対に関する条件を**グラフ彩色問題制約**と呼ぶ．グラフ彩色問題制約は効率的に解くために重要な制約である．本章では，後で述べるようにグラフ彩色問題制約を一般化し，**ハイパーグラフ彩色問題制約** (3.5.3.3 節) を導入する．

3.5.1.2 最小無矛盾 DFA の生成

前節で述べた論理式によって，$|K|$ 状態数以下の無矛盾 DFA の判定問題を SAT ソルバーは解くことができる．最小無矛盾 DFA を求めるときは，適当な状態数から一つづつ状態数を増やしていき，はじめに無矛盾 DFA が存在した時の状態数を最小の状態数として返す．こ

のアルゴリズムによって，その状態数未満には無矛盾 DFA がないことが証明され，厳密に最小無矛盾 DFA を求めることができる．

3.5.1.3　クリークによる状態数の初期値と対称性除去

前節の最小無矛盾 DFA を求めるアルゴリズムは，最小状態数に近い状態数から開始した方が効率的である．これは隣接グラフにおけるクリーク (完全部分グラフ:すべての状態間にエッジのある部分グラフ) を求めることで解決できる．クリークのメンバーとなる状態は全て異なった色を持つため，少なくともクリークの状態数未満には最小無矛盾 DFA が存在しないことが保証されるからである．

また，グラフ彩色問題においてはクリークが対称性除去に効果があることが知られている [84]．同値な解を表現する対称性は最小無矛盾 DFA 問題を解くための大きな障害であり，対称性除去とは同値である解が生成されないようにすることである．ここでいう同値な解とは，例えば，状態集合 S が 3 色 0,1,2 に塗り分けられたときに，0 に塗られた状態集合を 1，1 には 2，2 には 0 としたような意味が変わらない解のことをいう．

この対称性除去が解を求めるのに効果的であることを示す．式 3.12 においては，色の割り当て方について $k!$ の同値な解が存在することになる．ここで n 状態のクリーク $\{s_1, s_2, \ldots, s_n\} \subseteq S$ が求められた場合，これらの状態は全て異なった色を持つため，色を固定することができる (式 3.21)[1]．

$$x_{s_1,0} \wedge x_{s_2,1} \wedge \ldots x_{s_n,n-1} \tag{3.21}$$

この式によって，同値な解の数は $(k-n)!$ となる．

最小無矛盾 DFA を求める時間を短縮するには厳密な最大クリークの利用が効果的であることが示されている [53]．しかし，$(k-n)$ が大きい，すなわち最小無矛盾 DFA の状態数と最大クリークのサイズの差が大きい場合，この対称性除去の効果は限定的である．そこで，本章では新たな対称性除去制約を導入する (3.5.3.2)．

3.5.2　厳密な最大クリークの導入による改良

前節では最大クリークを導入することによって，調べるべき状態数の最小値の初期値が得られることと色が固定されることにより対称性除去に効果があることを示した．最大クリークはヒューリスティックなアルゴリズムによっても求めることができるが，MILP を使うことによって厳密に求めることができる．本節では，ヒューリスティックに求めた最大クリー

[1]実際の問題を表現するさいに，これらのクリークのメンバーに関する変数は記述する必要はない．また，クリークのメンバーである s_c について,$(s_c, t) \in E$ である状態 t の変数 $x_{t,c-1}$ は常に偽であるため，記述する必要はない．さらに，変数 $y_{a,i,j}$ についても同様に値が固定される場合がある．

クと厳密に求めた最大クリークを使ったアルゴリズムについて述べる．次に，最大クリークのヒューリスティック解法，MILP による厳密解法について述べる．そして，厳密に最小無矛盾 DFA を求める手法のバリエーションについて述べ，実験結果について述べる．

3.5.2.1 最小無矛盾 DFA の厳密解法

次に厳密解法のアルゴリズムを示す．

アルゴリズム　最小無矛盾 DFA の厳密解の計算
入力：APTA，出力：最小無矛盾 DFA
　APTA の各頂点間について，状態マージが可能か求める
　ヒューリスティックに最小無矛盾 DFA(D1) を求める
　ヒューリスティックに最大クリーク (C1) を求める
　if D1 の状態数=C1 のサイズ: return D1
　C1 を初期解として，MILP ソルバーにより最大クリーク (C2) を求める
　if D1 の状態数=C2 のサイズ: return D1
　upperSize=D1 の状態数-1
　lowerSize=C2 のサイズ
　lowerSize 以上，upperSize 以下で最小無矛盾 DFA(D2) を求める
　if D2 が存在：return D2
　return D1

提案するアルゴリズムはヒューリスティックな手法と MILP による手法と SAT による手法を用いたハイブリッドな手法である．個々の手法は，全体の計算時間を短くするために適所において使われる．まず，ヒューリスティックに最小無矛盾 DFA を求める．これは，状態数の上界を与える．次に，APTA の状態間の隣接辺を求め，最大クリークを求める．クリークは異なる色に割り当てなければならない状態の集合を表現している．そのため，最小無矛盾 DFA の下界を与える．上界＝下界であれば，解が見つかったことになる．最大クリークは，ヒューリスティックに求めることができるが，MILP ソルバーを使って厳密に求めることもできる．このとき，ヒューリスティックに求めた解を初期解として与えることで，効率化を図ることができる．最大クリークの頂点数よりも上界が高ければ，その間に最小無矛盾 DFA が存在することになる．従来研究では上界を用いていないが，本研究では上界も用いて，最小無矛盾 DFA を探す．この方法には，下界から順次求める，上界から順次求める，二分探索を用いる手法を実現した．下界からの場合は，特定の彩色数で解があるかを調べ，上界からの場合は，ある彩色数以下に解があるかを調べ，２分探索の場合はある彩色数の範囲に解が存在するかを調べる．これには，SAT を使うことができる．次節より，各処理で使われている手法について説明する．

3.5.2.2　最小無矛盾 DFA 問題のヒューリスティック解の導出

まず，提案アルゴリズムではヒューリスティックな手法を使って最小無矛盾を求める．これは，Blue-fringe[60] アルゴリズムを使って行うことができる．APTA の状態数を $|S|$ としたとき，APTA は $[0,\ldots,|S|-1]$ というリストで表現できる．APTA の各状態は開始状態から各入力記号に対して幅優先で番号づけられているものとする．このリストでは，状態 s が s 未満のどの状態とマージしたかを表現する．番号 s 未満の状態とマージしていなければ，s とする．この表現を使って，次のようにアルゴリズムを書くことができる．関数 stateMerge は map で表現された DFA において状態 s と t をマージし，作成された DFA を返す．

アルゴリズム 最小無矛盾 DFA のヒューリスティック解の計算
入力：APTA, 出力：DFA
　map=[0,1,. . . ,|S|-1]
　for s in [1,. . . ,|S|-1]:
　　if map[s]==s: s 未満の状態とマージしていないことを表現
　　for t in [0,. . . ,s-1]:
　　if 状態 s が状態 t にマージ可能:
　　map=stateMerge(map,i,j)
　　break
　return map

3.5.2.3　最大クリーク問題のヒューリスティック解の導出

ヒューリスティックにクリークを求めるためには，APTA の各状態を隣接辺の数を使って降順にソートし，先頭からクリークを作成する．

アルゴリズム最大クリークヒューリスティック解の計算
入力: 隣接辺数で降順にソートされた APTA の状態のリスト，出力：クリーク
　l=[q_0,q_1,. . . ,$q_{|S|-1}$]
　C=[q_0]
　for s in [1,. . . , |S|-1]:
　　if 状態 l[s] は C のどの状態ともマージできない: C.append(s)
return C

3.5.2.4 最大クリーク問題の厳密解の導出

厳密な最大クリークを算出する問題は NP 困難な問題であるため，多くの研究が行われている [85]．本研究では，今回のベンチマーク問題については非常に効率的であった MILP を使った手法を用いる．MILP を用いた方法が効率的である理由には，対象としているベンチマーク問題における隣接グラフのグラフ密度の高さが挙げられる．グラフ密度が高い場合，一般的に最大クリーク問題は解きやすい問題となる [90]．具体的には次の MILP の問題を解く [80]．この MILP は初期値を与えることによって効率的に解を求めることができ，実際の解を求める場合は，前節で述べた最大クリークのヒューリスティック解を MILP ソルバーに初期解として与えている．次の定式化で，1 の値を持つ変数 c_s が最大クリークのメンバーとなる．$E \subset S \times S$ は APTA における隣接行列であり，同じ色にできない状態間にエッジが存在する．

$$\begin{aligned} \text{maximize} \quad & \sum_{s \in S} c_s \\ \text{s.t.} \quad & c_s + c_t \leq 1, \forall s, t \in S, (s,t) \notin E \\ & c_s \in \{0,1\}, \forall s \in S \end{aligned} \tag{3.22}$$

3.5.2.5 数値実験

本節では，提案手法に関する数値実験について述べる．数値実験は，二つのベンチマークを用いた．一つは，Abbadingo においてベンチマーク問題[2] として公開されているものである．もう一つは，ランダムに生成した DFA から生成される事例を使う．実験では，使用言語は Python2.5.5 を用い，MILP ソルバーとして CPLEX12.1，SAT ソルバーとして CLASP13.1[17] を利用した．

3.5.2.5.1 Abbadingo コンペティションでのベンチマークによる評価

このベンチマーク問題は 4 状態から２１状態の DFA からランダムに生成された長さ 0 から２９の文字列が提示されている．ラベル数が 2 以上（実際は 2）の 770 個のデータセットからなっている．入力記号数は 2，出力記号数は 2 である．記号列数は 516 から 549 であった．作成された APTA に関して，状態数は 497 から 530 であった．密度（実際の遷移数／可能な遷移数）の平均はすべての事例についてほぼ 0.96 であり，ほとんどの状態で 2 種類の入力記号に対して遷移が定義されていることになる．また，全ての状態について，一つの出力記号を持っている．隣接グラフ（二つの状態がマージできない場合は辺を持ち，マージできる場合は辺を持たない）の密度（実際の辺の数／可能な辺の数）は，最低 0.02，最大 0.64，平均 0.51 であった．770 個のデータセット中，ヒューリスティックに求めた最小無矛盾の彩

[2] http://algos.inesc.pt/ãml/tar_files/moore_dfas.tar.gz

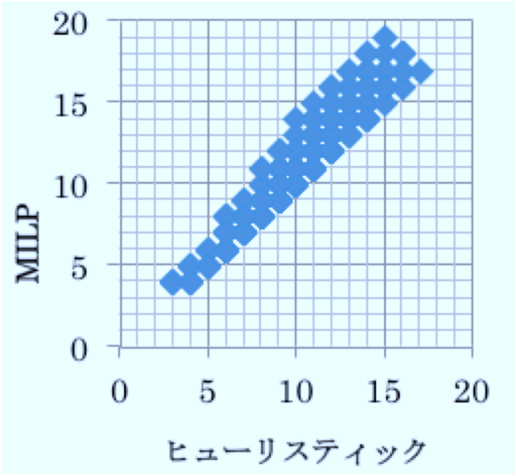


図 3.5: 最大クリークのサイズの比較

色数と最大クリークの状態数が一致した事例が 286 個あり，これらは評価より除外する．まず，ヒューリスティックに求めた最大クリークのサイズと MILP ソルバーを使って厳密に求めた最大クリークのサイズを比較する．図 3.5 は全体的な傾向をグラフ化したものであるが，(厳密な最大クリーク-近似による最大クリーク) は最大 4，最小 0，平均 0.92，分散 0.99 であり，ヒューリスティックな手法により求められた最大クリークはかなり良い近似を与えている．図 3.6 は (最小の彩色数-最大クリークの状態数) をグラフ化した．図 3.6 を見ると，最大クリークの最適解を求めることで，最適解との差が縮まっていることがわかる．

この実験データにおいては，最大クリークと最適解の差が小さいことから，SAT を使った厳密解の算出では，最大クリークの数から一つづつ彩色数を増やし，最初に見つけた無矛盾 DFA を最小無矛盾 DFA 問題の解とする．このとき，ヒューリスティックに求めた最大クリークを使って SAT で解いた場合に必要な時間と MILP で最大クリークを厳密に求め SAT で解いた場合の合計時間を比較する．図 3.7 に示したように，一次式で近似した場合の傾きから，平均的には MILP の厳密解を使うことで 50%の時間で解が求められていることがわかる．図 3.8 には MILP による最大クリークがヒューリスティックによる最小無矛盾 DFA の彩色数と一致しなかった 484 データセットについて，SAT 一回当たりの平均時間をグラフ化した．このグラフより，MILP による最大クリークを用いることによって平均 70%の時間で解が求まることがわかる．これは SAT において固定される色が増えたことによると考えられる．

このデータセットの場合，ヒューリスティックに最大クリークを求める平均時間は 0.31 秒であるのに対し，MILP により最大クリークを求める平均時間は 6.46 秒と非常に高速であり，SAT で解く時間を考えた時に MILP による最大クリークを用いる利点は大きい．図 3.9 に最小彩色数に対する計算時間を示した．SAT で解くべき変数が増えることから最小彩色数が増えると計算時間も増える傾向にあるが，MILP によって最大クリークの最適解を得ることで，

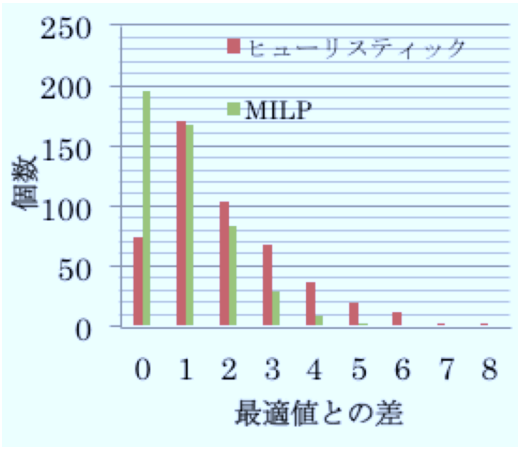


図 3.6: 最適値と最大クリークの比較

計算時間の短縮に効果があることが分かった．

3.5.2.5.2 ランダムに生成したデータセットによる数値実験

次に，ランダムに生成した DFA を使った実験結果について述べる．この場合，事例を生成する元となる DFA の状態数は 15 とした．入力記号・出力記号ともに 2 種類である．最大記号列長は 15 とし，最低は 1 とした．生成した事例数は，300，400，800 とし，各 10 回ずつ同じ DFA より事例を生成した．この実験では，比較的スパースな APTA を生成している．APTA の平均密度は 0.8，出力記号が一つである状態数の割合は 0.35，隣接グラフの密度は 0.08 である．APTA の状態数は，各事例数について，平均 965，1227，2101 であった．

各事例数について，(最小彩色数-最大クリークサイズ）の平均は，ヒューリスティックによる最大クリークでは，4.4，2.7，2.1 であり，MILP による最大クリークでは，2.9，2.3，1.8 となった．事例数が増えるとヒューリスティックによる最大クリークの値がよくなっている．

SAT による探索の手法として，最大クリークから一つずつ彩色数を増やす場合と，ヒューリスティックに求めた最小無矛盾 DFA の彩色数から一つずつ減らす場合と，2 分法で探索場合の実行時間を比較する．図 3.10 はヒューリスティックによる最大クリークを用いた場合のグラフである．各データセットについて，下界値より探索した場合を 1 とした時間をプロットしている．これらをみると，上から探索したものの計算時間が最も短いことがわかる．これは，多くのデータセットでヒューリスティックに生成した最小無矛盾 DFA が厳密解であるか，彩色数が近いことが原因である．MILP によって最大クリーク問題の厳密解を求める手法についても実験を行った．この場合，SAT についての計算時間自体は，最高で 45 %，最悪で 1.1 倍，平均 90%の計算時間で済む．ただし，隣接グラフがスパースであるため，MILP で最大クリーク問題を解くのに多くの時間を必要とした．この場合，ヒューリスティックな

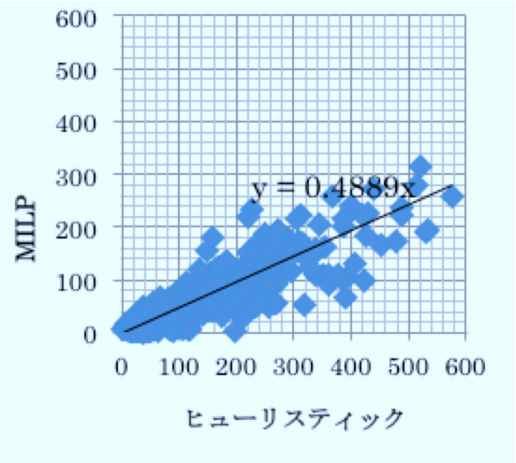


図 3.7: 解が求まるまでの時間比較（秒）

最大クリークを使って SAT で解く時間に比べ，最良で 63%，最悪で 41.1 倍，平均 12.1 倍の計算時間を必要とした．このような場合は，最大クリーク問題を MILP で解くのは好ましくないことを示している．

3.5.2.6　まとめ

ヒューリスティックな手法，MILP による手法，SAT による手法を組み合わせた最小無矛盾 DFA 問題のハイブリッド解法を提案し，有効性を数値実験で検証した．Abbadingo ベンチマーク問題のような密な隣接グラフを持つ APTA の場合，MILP によって高速に最大クリーク問題を解くことができ，解を高速に求めることが示せた．隣接グラフが疎な場合は，最大クリーク問題を解く時間が長いため，MILP を用いるのは不利である．また，ヒューリスティックに見つけた上界とクリークによる下界を使った手法を提案し，効果があることを確かめた．

3.5.3　問題記述の改良

前章で述べた従来手法をより効率的にするため，本章では次の三つの手法を提案する．3.5.3.1 節では 2 進数によって色を表現する．3.5.3.2 節では前章で述べた最大クリークで割り当てられなかった色に関する対称性除去の手法について述べる．3.5.3.3 節では，前章で導入したグラフ彩色問題制約をハイパーエッジに拡張した制約について述べる．

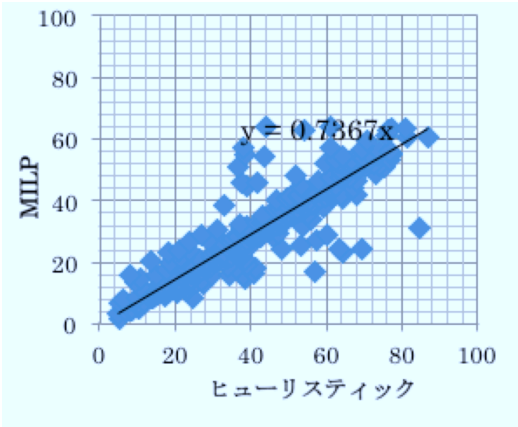


図 3.8: SAT 一回当たりの平均時間（秒）

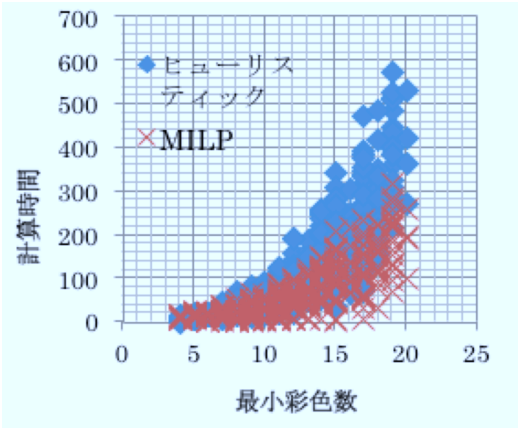


図 3.9: 最小彩色数に対する計算時間（秒）

3.5.3.1 変数の削減

SAT ソルバーの性能はすべての論理変数の真偽値を決定する効率で決まるので，決定すべき変数の数が少ないほうが有利である．そこで，グラフ彩色問題では，一つの頂点を k 色に塗り分けるときに式 3.12 のように k 個の変数を排他的に使う (以下，**排他表現**) のではなく，２進数として表現する方法 (以下，**２進表現**) が提案された [32]．この方法では２進数として 0 から $k-1$ の数を表現することにより，一つの頂点に対する変数の数を $\lceil log_2(k) \rceil$ に押さえることができる．

k 色で塗り分ける最小無矛盾 DFA 問題を考える (すなわち，$|K| = k$)．状態 $s \in S$ の色を $L(= \lceil log_2(k) \rceil)$ 個の変数 $b_s^{L-1}, b_s^{L-2}, \ldots, b_s^0$ によって表現する．このとき，論理式 $B_{s,i}, s \in$

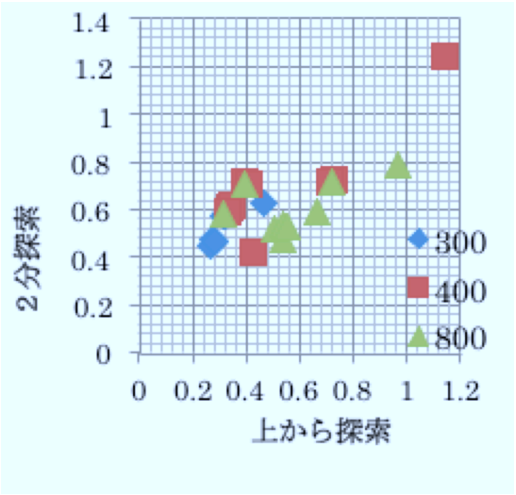


図 3.10: 下から探索を 1 としたときの時間比

$S, i \in K$ を次のように定義する．$B_{s,i}$ は真のとき，状態 s の色が i となる．

$$\begin{aligned}
B_{s,0} &\equiv (\neg b_s^{L-1} \wedge \neg b_s^{L-2} \wedge \cdots \wedge \neg b_s^1 \wedge \neg b_s^0) \\
B_{s,1} &\equiv (\neg b_s^{L-1} \wedge \neg b_s^{L-2} \wedge \cdots \wedge \neg b_s^1 \wedge b_s^0) \\
B_{s,2} &\equiv (\neg b_s^{L-1} \wedge \neg b_s^{L-2} \wedge \cdots \wedge b_s^1 \wedge \neg b_s^0) \\
&\cdots \\
B_{s,2^L-1} &\equiv (b_s^{L-1} \wedge b_s^{L-2} \wedge \cdots \wedge b_s^1 \wedge b_s^0)
\end{aligned} \tag{3.23}$$

k 色の割り付けを行うとき，$B_{s,k}, B_{s,k+1}, \ldots, B_{s,2^L-1}$ が真となるような色は使われないので，すべての論理式を偽とする．

$$\neg B_{s,k} \wedge \neg B_{s,k+1} \wedge \cdots \wedge \neg B_{s,2^L-1} \tag{3.24}$$

式 3.14 は次のように表現できる．

$$\begin{cases}
\neg B_{s,i} \vee z_i & \text{if } s \in Acc \\
\neg B_{s,i} \vee \neg z_i & \text{if } s \in Rej
\end{cases} \tag{3.25}$$

式 3.15 で用いている $y_{a,i,j}$ についても $x_{s,i}$ と同じく 2 進表現を使うことができる．論理式 $A_{a,i,j}$ を L 個の変数 $a_{a,i}^{L-1}, a_{a,i}^{L-2}, \ldots, a_{a,i}^0$ によって表現する．$A_{a,i,j}$ は真のとき，$j = \delta^K(i, a)$ となる．

$$\begin{aligned}
A_{a,i,0} &\equiv (\neg a_{a,i}^{L-1} \wedge \cdots \wedge \neg a_{a,i}^1 \wedge \neg a_{a,i}^0) \\
A_{a,i,1} &\equiv (\neg a_{a,i}^{L-1} \wedge \cdots \wedge \neg a_{a,i}^1 \wedge a_{a,i}^0) \\
A_{a,i,2} &\equiv (\neg a_{a,i}^{L-1} \wedge \cdots \wedge a_{a,i}^1 \wedge \neg a_{a,i}^0) \\
&\cdots \\
A_{a,i,2^L-1} &\equiv (a_{a,i}^{L-1} \wedge \cdots \wedge a_{a,i}^1 \wedge a_{a,i}^0)
\end{aligned} \tag{3.26}$$

$A_{a,i,j}[n]$ によって，$a^n_{a,i}$ に関する論理式を参照する．例えば，$A_{a,i,0}[L-1] = \neg a^{L-1}_{a,i}$，$A_{a,i,1}[0] = a^0_{a,i}$ である．このとき，式 3.15 は次のように表現できる．

$$\begin{cases} A_{a,i,j}[0] \vee \neg B_{s,i} \vee \neg B_{\delta(s,a),j} \\ A_{a,i,j}[1] \vee \neg B_{s,i} \vee \neg B_{\delta(s,a),j} \\ \dots \\ A_{a,i,j}[L-1] \vee \neg B_{s,i} \vee \neg B_{\delta(s,a),j} \end{cases} \tag{3.27}$$

式 3.16 も同様に表現することができる．

エッジ集合に関する式 3.19 は次のように表現できる．

$$\neg B_{s,i} \vee \neg B_{t,i}, (s,t) \in E, i \in K \tag{3.28}$$

以上述べたように，２進表現によって k 色の無矛盾 DFA の判定問題を表現できる．しかしながら，排他表現に比べて，例えば論理式の数が多くなったり，論理式が長くなってしまうという問題がある．例えば，排他表現ならば式 3.19 のように２変数でエッジを表現できるのに対し式 3.28 では $2\lceil \log_2(|K|) \rceil$ 個の個の変数が必要となる．これを避けるため，排他表現と２進表現の**ハイブリッド表現**を提案する．ハイブリッド表現では式 3.12,3.13,3.17 以外は排他表現を使う．このため，次の論理式を導入する．

$$\begin{cases} \neg B_{s_i,k} \vee x_{s_i,k} \\ B_{s,i}[l] \vee \neg x_{s,i}, l = 0, 1, \dots, L-1 \end{cases} \tag{3.29}$$

$$\begin{cases} \neg A_{a,i,j} \vee y_{a,i,j} \\ A_{a,i,j}[l] \vee \neg y_{a,i,j}, l = 0, 1, \dots, L-1 \end{cases} \tag{3.30}$$

ハイブリッド表現の特徴は，２進表現と比べて論理式の数が少なく，論理式の長さが短くなる点である．変数の数については，排他表現と２進表現に必要な変数の総和であるので多くなってしまう．しかし，排他表現および２進表現を併せ持った変数の依存関係があるので，実際の SAT ソルバーでの実験によってどの表現方法がよいかを評価する．具体的にハイブリッド表現が排他表現および２進表現とどの程度記述量が異なるかについては 3.5.3.4 で詳しく述べる．

3.5.3.2 対称性の除去

本節では，クリークによって状態が割り当てられない色に対する対称性除去の制約を導入する．図 3.11 には APTA とそれから作られる隣接グラフを示した．b) でエッジは異なる色に彩色しなければならない二つの状態を示している．赤線で結ばれた３状態はクリークを構

成している．c) は各状態に５色から色を割り当てることを示している．クリークとなる状態には左から固定した色が割り当てられている．右側の二つの色にはクリークの状態が割りつけられていないので，クリークの状態以外の状態へ色を割り当てることになる．

しかしながら，右側の２色は区別されていないため，対称になっている．この対称性を除去するために，クリークの状態以外の状態を適当な順番に並べて，右側の２色に割り当てる．状態の順番を $R =< s_1, s_2, \ldots, s_n >$ とする．また，クリークの状態が割り当てられる色の集合を $C = \{c_1, \ldots, c_c\}$, それ以外の色の集合を $D = \{d_1, \ldots, d_d\}$ とする．このとき，D の色に対して，R の状態の割り当て方に違いを作ることによって対称性を除去する．具体的には次の式のように，s_1 は C の要素か d_1，s_2 は C の要素か d_1, d_2 に割り付け可能なようにする．これは式 3.12 の代わりに使われる．

$$x_{s_i,c_1} \vee \cdots \vee x_{s_i,c_c} \vee x_{s_i,d_1} \vee \cdots \vee x_{s_i,d_{\min(i,d)}} \tag{3.31}$$

R の状態を D の色に割り当てるときは，可能な最も若い番号に割り当たるようにする．具体的には，s_2 は d_1, d_2 に割り当たる可能性があるが，s_1 が C の要素に割り付けられたときは s_2 は C の要素か d_1 に割り付けられ，d_2 には割り付けられないようにする (式 3.32).

$$\begin{aligned} &x_{s_i,d_i} \vee \cdots \vee x_{s_{i+k},d_i} \vee \neg x_{s_{i+k+1},d_{i+l+1}}, \\ &\quad i = 1, 2, \ldots, d-1, k = 0, 1, \ldots, n-i-1, l = 0, 1, \ldots, k \end{aligned} \tag{3.32}$$

図 3.11 を使って説明する．状態４は C 内に一つ，D 内に二つの候補があるが，D からは一つを選ぶ．状態５については D から二つ選べるようにする (式 3.31)．状態４の色が d_1 にならない場合は，状態５の色は d_2 にならないようにする (式 3.32)．この場合は，状態５は D の中からは一色しか選べない．

3.5.3.1 で２進表現について述べた．式 3.31 については，次のように書くことができる．

$$B_{s_i,c_1} \vee \cdots \vee B_{s_i,c_c} \vee B_{s_i,d_1} \vee \cdots \vee B_{s_i,d_{\min(i,d)}} \tag{3.33}$$

B_{s_i,c_j} は式 3.23 のように積の形で記述されているので，式 3.33 は論理積が論理和で結合しているので，SAT ソルバーで解釈できるように論理和の論理積の形に直さなければならない．一般に，m 個の変数が論理積で結合しているとし，そのような論理式が n 個論理和で結合している場合，m^n 個の n 個の変数の論理和ができ，n が大きい場合は実現が困難である．そのため，２進表現については式 3.31 の代わりに，式 3.34 を用いる．

$$\neg B_{s_i,d}, d \in \{d_{\min(i,d)+1}, \ldots, d_d\} \tag{3.34}$$

式 3.32 についても同じ理由で実現は難しく，２進表現については実現しないことにした．

この対称性除去においては，R を作るために，なんらかの状態の順序付けが必要である．直感的に言って，初めの方に位置する状態はクリークの状態と同じ色にならない方が良い．

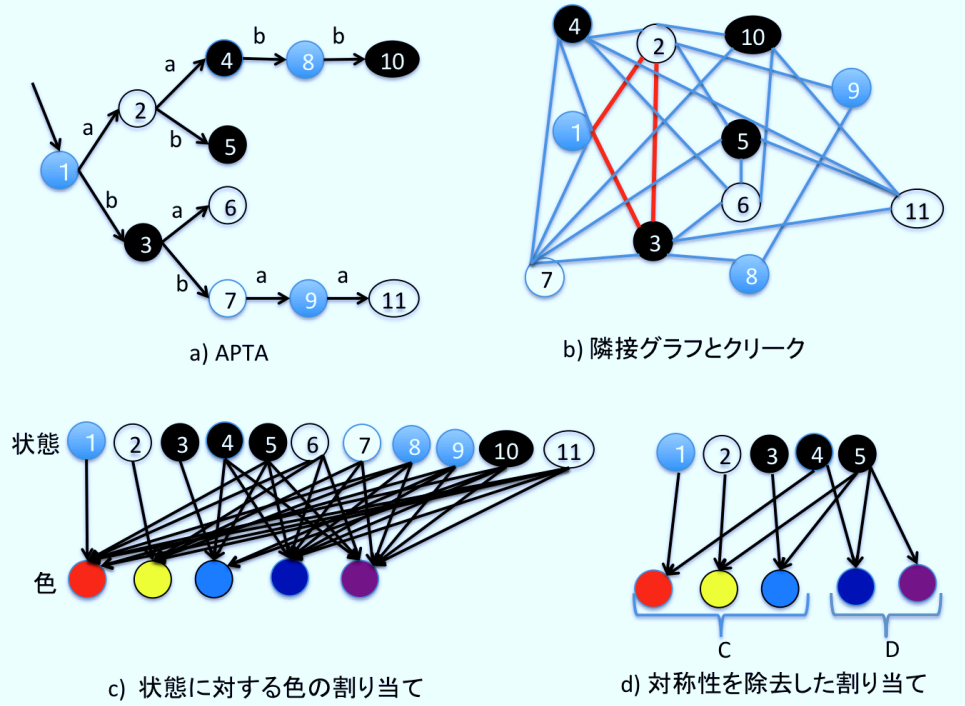


図 3.11: 対称性除去

これは，クリークと同じ色になる可能性が少ない状態については，D の中で割り当てられる色の数を少なくすることで，決定的に近く色を割り当てられるようになるからである．しかしながら，一般的に最適な状態の順番はわからないので，本章では二つの方法を試みる．一つは，エッジ数の多い状態から順番に並べたものである (**エッジ順**)．エッジ順では，エッジ数が多い状態ほど可能な色が少なくなる．もう一つは，各状態を含む最大クリークの近似解を求め (3.5.2.3 の手法で先頭の状態を最大クリークを求めたい状態とする)，その大きさによって状態をソートしたものである (**クリーク順**)．クリーク順では，大きいクリークに属する状態ほど可能な色が少なくなる．これらの計算時間に与える影響は実験的に評価する．

3.5.3.3 ハイパーグラフ彩色問題制約

3.5.1 節で述べた従来の方法では，式 3.14 だけでなく，式 3.19 によりマージできない状態対 (エッジ) を同じ色にしない制約を加えていた．これは冗長な式であるが，効率化の点からは効果的である．この制約を使わないで二つの状態がマージできないことを判定するためには，他の変数の値も決まっていなければならない．例えば，式 3.14 だけでは状態 $s, t \in S$ のとき，$s \in Acc, t \in Rej$ でなければ，直接二つの状態がマージ可能かどうかを判定することはできない．式 3.20 に示したように，ある文字列 $u \in \Sigma^*$ に対して，$\delta(s, u) \in Acc$，$\delta(t, u) \in Rej$

であればマージできないことがわかるが，これを知るためには s,t が割り当てられた色 i,j から $\delta^K(i,u)$ および $\delta^K(j,u)$ が決まっている必要がある．このように色についての遷移が決まっていないとマージ可能かどうかが判定できないが，式 3.19 は遷移が決まっていなくても適用可能である．

無矛盾 DFA における状態間の関係は，二つの状態で構成されるエッジだけではなく，三つ以上の状態で構成されるハイパーエッジも存在する．図 3.12 にハイパーエッジの例を示す．この場合，状態 $1,2,3$ について，マージ $(1,2),(1,3),(2,3)$ は可能であるが，３状態のマージ $(1,2,3)$ はできない．このことは，状態 $1,2,3$ について最低 2 色必要なことを意味している．このような状態集合 $\{1,2,3\}$ のことをハイパーエッジと呼ぶ [46]．ハイパーエッジはエッジの一般化になる．ハイパーエッジの集合を $\mathbb{H} \subset 2^S$ で表す．このとき，少なくとも一色は i ではない次の制約が成り立つ．

$$\neg x_{s_1,i} \vee \neg x_{s_2,i} \vee \cdots \vee \neg x_{s_h,i}, \{s_1, s_2, \ldots, s_h\} \in \mathbb{H} \tag{3.35}$$

ハイパーエッジ集合はエッジ集合と同じく再帰的に計算できる．ここで $h \subset S$ について $\delta(h,u) \equiv \{\delta(s,u) \in S | s \in h\}$ と定義する．さらに，$rec(h) = \{u \in \Sigma^* | \exists s,t \in h(t = \delta(s,u))\}$ を用いる．

$$\begin{array}{ll}
(s,t) \in \mathbb{H} & \text{if } (s,t) \in E \\
h \in \mathbb{H} & \text{if } \exists h \subset S(\\
 & \quad \nexists l \subset h(l \in \mathbb{H}) \wedge \\
 & \quad \exists a \in \Sigma(|h| = |\delta(h,a)| \wedge \delta(h,a) \in \mathbb{H})) \\
h \in \mathbb{H} & \text{if } \exists h \subset S(\\
 & \quad \nexists l \subset h(l \in \mathbb{H}) \wedge \\
 & \quad \exists s \in h(h \setminus \{s\} \cup \delta(s, rec(h)^*) \in \mathbb{H})) \\
h \cup \{s\} \in \mathbb{H} & \text{if } \exists h \subset S \exists s \in S - h(\\
 & \quad \nexists l \subset h \cup \{s\}(l \in \mathbb{H}) \wedge \\
 & \quad \exists t \in h \exists u, v \in rec(h \cup \{s\})^*(\\
 & \quad\quad (\delta(s,u), \delta(t,v)) \in E))
\end{array} \tag{3.36}$$

1 番目の式は，エッジはハイパーエッジであることを示している．2 番目の式は，ある記号によって遷移した状態集合がハイパーエッジであるときは，遷移元の状態集合もハイパーエッジであることを示す．３番目の式は，状態集合の一つの状態を出発点とするループした遷移によって導かれるハイパーエッジの条件である．４番目の式は，ハイパーエッジではない状態集合から導かれるハイパーエッジの条件である．図 3.12 において，$rec(\{1,2,3\}) = \{a,b\}$ となる．このとき，$(\delta(2,\lambda), \delta(3,aa)) \in E$ であるから，$\{1,2,3\}$ はハイパーエッジである．

APTA の状態数が大きいとき，すべてのハイパーエッジを導くことは難しいので，本章ではハイパーエッジの状態数が 3 の場合について実験を行った．

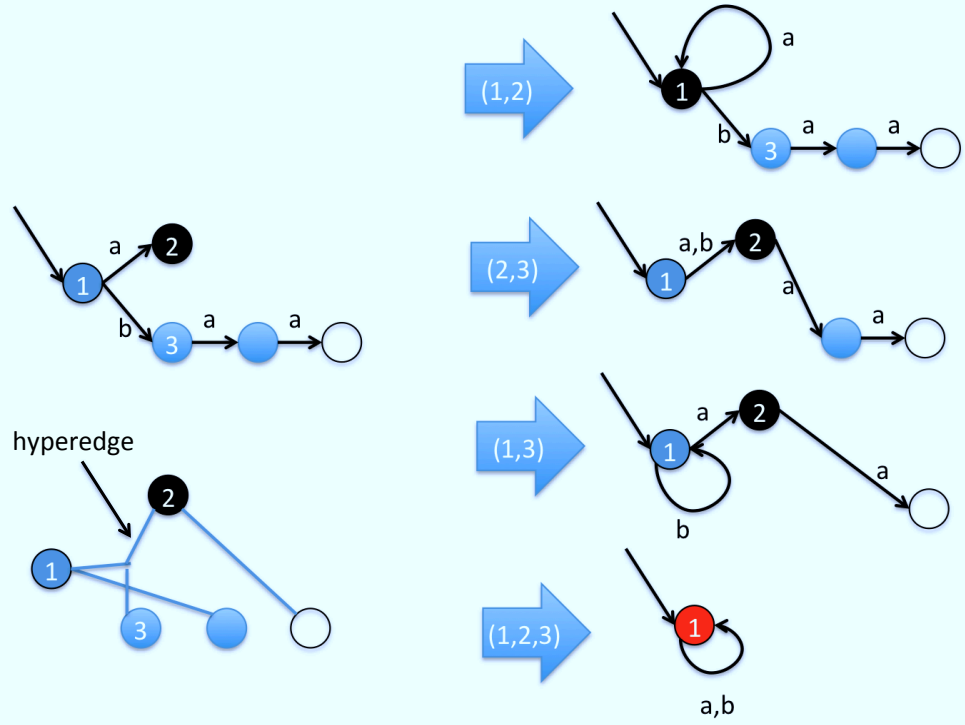


図 3.12: APTA とハイパーエッジ

3.5.3.4 問題の大きさについて

本節では問題の記述量の観点からこれまで述べた手法についてまとめる．SAT では，論理変数およびその否定をリテラルと呼び，リテラルの論理和の論理積で問題を表現する．3.3 には，排他表現，２進表現，ハイブリッド表現ごとに問題の記述量を示す．ここでは，$APTA = (\Sigma, S, s_0, \delta, Acc, Rej, Dont)$ が与えられ，それを無矛盾 DFA$(\Sigma, K, k_0, \delta^K, Acc^K, Rej^K, Dont^K)$ に変換する場合を示す．問題を記述する量は，APTA の状態数 S，文字の個数 Σ，受理状態の数 A，不受理状態の数 R，無矛盾 DFA の状態数 K，APTA のクリークサイズ C，グラフ彩色問題制約の数 E，状態数３のハイパーグラフ彩色問題制約の数 H である．また，２進表現の時の無矛盾 DFA の状態の記述長を $L = \log_2(K)$ とする．ここで，式の見通しを良くするために，$E \gg S \gg A, R, \Sigma$ を仮定する．

3.5.1 で述べた SAT による最小無矛盾 DFA 問題の記述を基本問題と呼ぶことにする．3.3 では，基本問題を排他表現，２進表現，ハイブリッド表現で記述した場合の問題の大きさを示した．グラフ彩色問題制約に関する項 EK を除いたとき，２進表現は変数の数が最も少ないが，基本問題の大きさは，論理和の数について排他表現の約 L 倍，リテラル数に関しては約 L^2 倍になっている．これに対して，色の割り当てを２進数で行うハイブリッド表現は，排他表現および２進表現の両方の変数を使う表現方法であるが，論理和の数およびリテラル数

表 3.3: 記述方法の比較

(a) 基本問題に関する記述

表現方法	変数の数	論理和の数	リテラル数
排他表現	$SK + \Sigma K^2 + K$	$2\Sigma SK^2 + EK$	$6\Sigma SK^2 + 2EK$
２進表現	$SL + \Sigma KL + K$	$2\Sigma SLK^2 + EK + 2^L S$	$4\Sigma SL^2K^2 + 2ELK + 2^L SL$
ハイブリッド表現	$S(K + L) + \Sigma(K^2 + KL) + K$	$\Sigma(\frac{1}{2}K + L)K^2 + EK + 2^L S$	$\Sigma(K + 2L)K^2 + 2EK + 2^L SL$

(b) 対称性除去に関する記述

表現方法	論理和の数	リテラル数
排他表現，ハイブリッド表現	$(K - C)S$	$\frac{1}{2}(K - C)S^2$
２進表現	$\frac{1}{2}(K - C)^2$	$\frac{1}{2}(K - C)^2 L$

(c) ハイパーグラフ彩色問題制約 (3 状態)

表現方法	論理和の数	リテラル数
排他表現，ハイブリッド表現	HK	$3HK$
２進表現	HK	$3LHK$

ともに排他表現の約 $\frac{S}{K}$ になっていて，記述量が最も少ない表現方法である．

例えば排他表現に関して，提案手法である対称性除去はクリークの大きさと無矛盾 DFA の状態数の差が広がるにつれ，リテラル数を S^2 に比例して増やす．また，ハイパーグラフ彩色問題制約は，無矛盾 DFA の状態数に対してリテラル数を H に比例して増やす．このような性質を持つ制約条件の実行時間に与える影響は次章で実験的に調べる．

3.5.3.5　実験と考察

本章では，提案手法に関する実験を行ったのでその結果を報告する．実験の目的は様々な問題設定に対して提案手法の効率性を実行時間で評価することである．実験は，QuadCore AMD Opteron Processor(TM) 8384(2.7GHz) CPU，64GB メモリ搭載の Linux ワークステーションを用いて行った．プログラムは Python2.6 を用いて作成し，Clasp1.3.6[17] を SAT ソルバーとして用いた．各々のプログラムでは近似的に求められた最大クリーク (3.5.2.3 参照) の状態数から無矛盾 DFA が見つかるまで状態数を一つづつ増やしていく．各々の無矛盾 DFA の状態数に関して，SAT の問題を生成し，SAT ソルバーにより充足可能かどうかを判定する．充足可能であれば，その状態数について無矛盾 DFA が存在したことになる．この SAT ソルバーの一回の実行について 30 分の制限時間を設けた．SAT ソルバーの実行時間が 30 分を超えた場合は，それ以上無矛盾 DFA を探すことはせずに実行を打ち切った．また，実行時間については SAT ソルバーの実行時間だけを評価する．これは，SAT ソルバー一回の実行のための問題を作成する時間は 1 秒〜4 秒の間でばらつきがあったが，SAT ソルバーで効率的に解くことができるかどうかを評価したかったためである．

プログラムの設定では次の与え方が考えられる．

1. 表現方法について 3 通り (E:排他表現，B: ２進表現，H:ハイブリッド表現)

2. 付加的な制約について３通り (なし，g:グラフ彩色問題制約，h：グラフ彩色問題制約および３状態のハイパーグラフ彩色問題制約)

3. 対称性除去について３通り (なし，e:エッジ順，c:クリーク順)(クリークの状態の色は固定．ある状態を含むクリークは近似手法で求める)

プログラムの設定を略号を使って表す．例えば，Egc は (排他表現，グラフ彩色問題制約，クリーク順) を表す．3.5.1 で述べた従来手法は，排他表現，グラフ彩色問題制約を使っているので Eg になる．本章では，E, Eg, Ege, Egc, Eh, Ehc, Bgc, Bhc, Hgc, Hhc についての実験結果を示す．従来手法の Eg に対して，これらは提案手法の効果があったものである．具体的には，実験した中で最も難しい問題が集まっている文字列数 300 において，Eg よりも多くの問題を解いたものを選んでいる．ただし，E はベースラインとして，Eh は Eg にハイパーグラフ彩色問題制約を加えたものとして，Ege は Egc に対して対称性除去の比較を行うために結果を示す．

3.5.3.5.1 ベンチマーク問題

本章では標準的なベンチマーク問題[3]を用いて評価を行う．このなかで，15 状態以上の DFA(最大は 21 状態) から生成されているベンチマーク問題 183 題を使った．表 3.4 にベンチマーク問題のデータを示す．これらのベンチマークでは，文字は二種類，ラベルは二種類，最大文字列長は 29，最小文字列長は 0 である．実験の設定では，すべてのラベル付き文字列を扱う (最小文字列数 516，最大文字列数 545) 場合とランダムに 450，400，300 文字列を選んだ場合を扱う．表 3.4 において，APTA の状態数，クリークの状態数，最小無矛盾 DFA の状態数のそれぞれ平均を示している．また，ラベルを持つ状態数 (受理状態数と不受理状態数の合計) の全状態数に対する割合 (ラベル密度) の平均，および APTA から作られる隣接グラフのエッジ数の可能なエッジ数に対する割合 (グラフ密度) の平均を示す．一般的に，状態数が多いと問題の大きさが大きくなり，クリークと最小無矛盾 DFA のギャップが大きいと問題が難しくなると考えられる．また，ラベル密度が小さいとラベルを決定すべき状態の数が多くなり，グラフ密度が小さい問題はグラフ彩色問題の観点からは解くことが難しいと考えられる．

ハイパーグラフ彩色問題制約に関しては，3 状態のものだけを使った．ベンチマーク問題におけるグラフ彩色問題制約とハイパーグラフ彩色問題制約の密度を 3.13 で比較する．ハイパーグラフ彩色問題制約についての密度は $\frac{\text{ハイパーグラフの数}}{\text{可能な３状態の組合せの数}}$ により算出した．この分母では 2 状態のグラフ彩色問題制約が含まれている組み合わせはカウントしない．3.13 から事例数が減ることによってグラフ密度は下がるが，ハイパーグラフ密度はあまり変わらないことがことがわかる．

[3] http://algos.inesc.pt/ãml/tar_files/moore_dfas.tar.gz

表 3.4: 実験データ (平均)

文字列数	すべて	450	400	300
状態数	515	511	509	498
最大クリーク	10.9	10.2	9.6	8.3
最小無矛盾 DFA	13.7	13.4	13.2	12.2
ラベル密度 (%)	100	85.3	76.4	58.9
グラフ密度 (%)	55.2	43.6	37.0	23.8

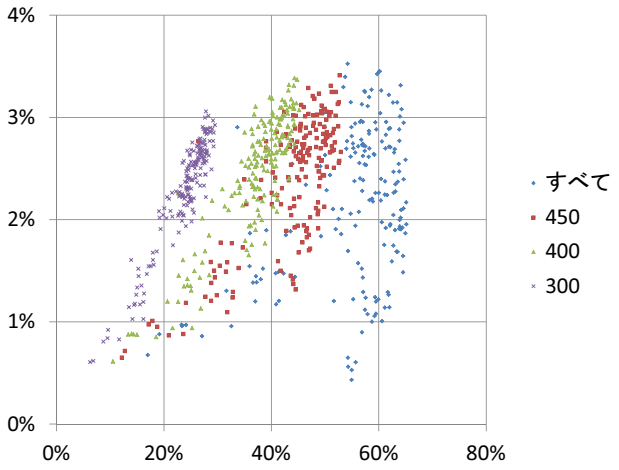


図 3.13: グラフ彩色制約とハイパーグラフ彩色制約 (3 状態) の密度比較．横軸がグラフ彩色制約，縦軸がハイパーグラフ彩色制約

3.5.3.5.2　実行可能性についての評価

本節では，SAT ソルバーの一回あたりの実行時間を 30 分で打ち切ったことによる評価を行う．表 3.5 に各プログラムに対する解が得られた問題の数と解が得られなかった問題のグラフ密度の平均およびクリークの大きさと最小無矛盾 DFA の状態数のギャップの平均を示す．

ベンチマーク問題に含まれるすべての文字列を使ったときは，ほとんどのプログラム設定ですべての問題を解くことができたが，文字列数を減らすにつれて，解ける問題の数が減少した．これは元のベンチマーク問題が解きやすいことを示している．解けなかった問題について表 3.4 のデータと比較すると，グラフ密度はほとんど変わっていないが，ギャップは非常に大きくなっている．このため，解けなかった原因はギャップの大きさにあると考えられる．

このため，提案手法の中ではギャップに関する手法である対称性除去が有効であると考えられる．表 3.5 によれば，クリーク順による対称性除去 (c を持つ手法) を入れた手法は対称性除去を入れていない手法 (E,Eg,Eh) に比べて解ける問題の数が多くなっており，その有効性を裏付けている．しかしながら，エッジ順を使った対称性除去である Ege は逆に解ける問

表 3.5: 解が得られた問題数 (総問題数 183) - 括弧内は解けなかった問題の平均グラフ密度 (%) と平均ギャップ．並列は少なくともひとつの手法で解けた問題数をカウント -

列数	すべて	450	400	300
E	181(56,11.0)	177(46,11.0)	164(40,10.8)	129(26,9.6)
Eg	183(-,-)	182(50,10.0)	173(40,11.0)	136(26,10.0)
Ege	179(56,10.5)	171(44,9.4)	160(38,10.5)	117(26,9.0)
Egc	183(-,-)	182(50,10.0)	179(40,11.2)	145(26,10.1)
Eh	183(-,-)	181(48,10.5)	172(40,11.5)	135(26,9.9)
Ehc	183(-,-)	182(50,10.0)	179(40,11.2)	141(26,9.9)
Bgc	183(-,-)	181(44,12.0)	174(40,10.9)	140(26,9.8)
Bhc	183(-,-)	182(46,11.0)	174(40,11.2)	140(26,9.8)
Hgc	183(-,-)	182(50,10.0)	174(40,10.8)	141(26,10.0)
Hhc	183(-,-)	181(48,10.5)	174(40,10.2)	138(26,9.9)
並列	183	183	180	151

題の数が少なくなっていた．このことから，提案手法である対称性除去の効果は状態の順番により大きく影響を受けるが，クリーク順がよいということがわかった．

排他表現やグラフ彩色問題制約などの従来手法に比べて，提案手法である２進表現などの色の表現方法やハイパーグラフ彩色問題制約は，解ける問題数を増やしていなかったが，文字列数 300 の問題について，Bhc は Egc の解けなかった 5 題の問題を解いていた．すなわち，単に問題数だけでなく，手法の違いが解ける問題の違いになって表れている．このような手法と解ける問題の間の関係を分析することは難しいが，他にはない問題が解けることから，２進表現やハイパーグラフ彩色問題制約を導入する意義は十分あると考えられる．

表 3.5 の” 並列” の欄には，少なくとも一つのプログラムで解けた問題の数を示した．これは各々のプログラムを並列で実行した場合に解ける問題数を示している．特に文字列数 300 の場合，従来手法の Eg が 136 題であったのに対して，最多はクリーク順による対称性除去を行った Egc の 145 題であったが，組み合わせることで 151 題解けるようになる．これは，各々のプログラムで解ける問題が少し異なるためである．このことについて次節で考察する．

3.5.3.5.3 実行時間に関する評価

本節では，前節において従来手法である Eg よりも多くの問題が解けていた解法，すなわち，Eg,Egc,Ehc,Bgc,Hgc について，実際に解くことができた問題に対する計算時間を評価する．

表 3.6 に従来手法である Eg に対する平均相対時間 (個々の問題での相対時間の平均) と合計相対時間 (合計計算時間の相対時間) によって結果をまとめた．平均相対時間を見ると，すべ

表 3.6: 従来手法 Eg に対する平均相対時間と合計相対時間 (%)

(a) 平均相対時間

文字列数	すべて	450	400	300
Egc	86.6	77.9	174.1	113.3
Ehc	90.1	92.2	312.5	136.0
Bgc	275.3	221.5	216.1	224.8
Hgc	338.4	346.1	519.8	437.6
並列 (Eg,Egc,Ehc)	79.7	69.0	67.8	66.6
並列 (Eg,Egc,Ehc,Bgc,Hgc)	79.7	69.0	67.1	66.5

(b) 合計相対時間

文字列数	すべて	450	400	300
Egc	55.9	21.7	28.6	25.9
Ehc	44.8	49.6	44.8	22.1
Bgc	317.0	112.7	86.9	132.5
Hgc	825.6	180.2	94.6	103.2
並列 (Eg,Egc,Ehc)	38.5	14.4	13.6	7.7
並列 (Eg,Egc,Ehc,Bgc,Hgc)	38.5	14.4	13.1	7.6

ての文字列および文字列数が 450 のような文字列数が多い時は，Egc および Ehc が従来手法よりも実行時間が短いが，それ以外では従来手法が良かった．しかしながら，計算時間が長い問題が大きな影響を与える合計相対時間をみると，すべての文字列数について，Egc，Ehc が優れていた．この結果は，計算時間が短い問題については Eg の計算時間が他のプログラムよりも短いが，Eg の計算時間が長い問題を Egc，Ehc が効率的に解いていることを示している．これから，計算時間が長い問題については，Egc,Ehc は有効であることがわかった．

前節の結果から 2 進表現 (Bgc) やハイブリッド表現 (Hgc) はすべての文字列数について Eg よりも多くの問題を解いているのだが，文字列数が 400 の合計相対時間を除いて，Eg よりも長い計算時間がかかっている．しかし，後述する並列での実行結果から，これらの手法が最も短い計算時間で解を求める問題が存在するため，他の手法と共に用いることで計算時間を短縮できると考えられる．

Eg,Egc,Ehc,Bgc,Hgc に共通して解けている問題についての実行時間を図 3.14, 図 3.15, 図 3.16, 図 3.17 に示す．グラフにおいて，横軸は問題であり，縦軸は実行時間 (秒)(対数スケール) である．問題は Eg において実行時間の短いものから並べてある．これらのグラフから，文字列数が多い場合は，プログラム間の違いが計算時間に現れているように見える．つまり，Egc と Ehc のグラフが Eg の下にあり，Bgc と Hgc が上にある．しかし，文字列数が少なくなったときは，特に実行時間がかかる問題では手法間の違いが実行時間の違いとなって現れ

ていない．そこで，複数の手法を並列に実行することで一つの問題を解き，最も速く得られた解を利用する方法が考えられる．図 3.18 に最も良いと思われる Egc に対する 5 種類のプログラムの最短時間の相対時間を示す．これと表 3.6 に示した最短時間での平均相対時間および合計相対時間をみると，文字列数が少ない問題ほど並列に実行することによる実行時間の短縮が顕著であることがわかる．

表 3.6 に，Eg,Egc,Ehc の 3 種類で並列化した場合と，Bgc,Hgc を追加した 5 種類で並列化した場合の結果を示す．文字列長が 400 および 300 のときは，少しではあるが Bgc,Hgc を追加したほうが速く解を求められ，これらの手法を導入する意義は十分あると考えられる．特に，5 種類で並列化した場合，文字列数 300 のとき，従来手法に比べ，平均相対時間で 66.5% および合計相対時間で 7.6% と効率的に問題が解けることがわかった．

3.5.3.5.4 各提案手法に関する考察

実行時間に関する実験では，提案手法の中でクリーク順を用いた対称性除去が効率化に最も寄与することがわかった．この結果から，最小無矛盾 DFA を求める際に，クリークの大きさと最小無矛盾 DFA の状態数のギャップに関する対称性除去が重要であることがわかった．しかし，エッジ順の結果が良くないことは，状態の順番についてはどのような方法でも良いわけではないことを示している．この点から，本章の提案手法であるクリーク順を使った対称性除去は優れた手法であると言える．

色の表現方法について，本章では 2 進表現および排他表現と 2 進表現のハイブリッド表現を提案した．制限時間内に解くことができた問題について見ると，文字列数が 300 のデータに関して，Egc が解けなかった問題 4 題を Bgc は解くことができ，1 題を Hgc は解けていた．このことから考えると，実行時間については多くの問題で 2 進表現やハイブリッド表現は排他表現に劣っているものの，問題によっては有効な場合が存在する．ハイパーグラフ彩色問題制約については，制約時間内に解くことができた問題について見てみると，文字列数が 300 のデータに関して，Egc が解けなかった問題を Ehc は解くことが出来なかったが，Bgc が解けなかった 6 題を Bhc が解くことができ，Hgc が解けなかった 1 題を Hhc は解くことができた．

これより，2 進表現による色の表現およびハイパーグラフ彩色問題制約を導入することで解けるようになる問題が存在することがわかった．個々の問題に応じて効率的な手法を選ぶことは難しいと考えられるが，本章では並列でプログラムを実行した場合の評価を行った．その結果として，実行時間を短縮するためには，手法にバリエーションをもたせ，並列に実行することが重要であると考えられる．

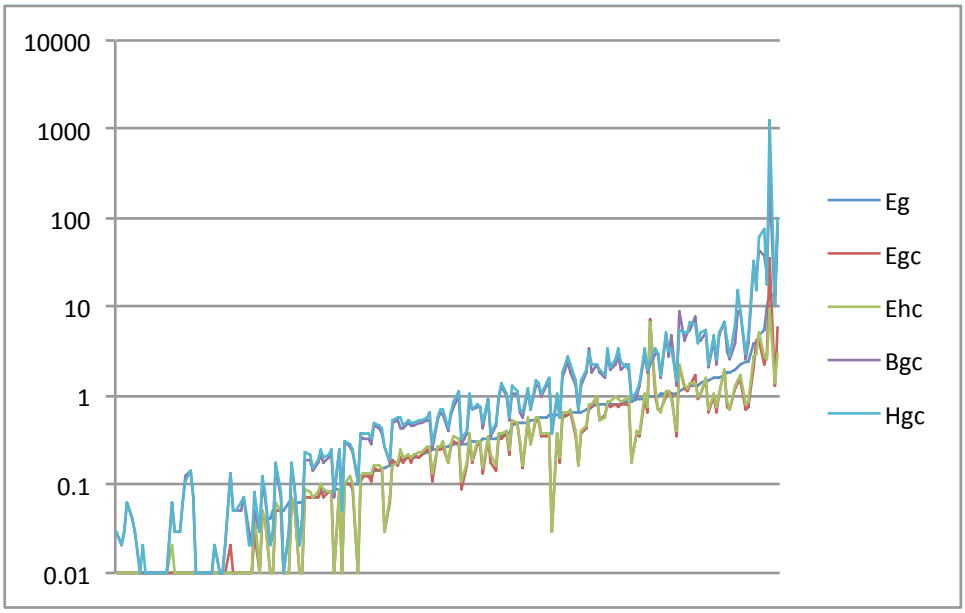


図 3.14: 実行時間 (すべての文字列)(横軸は問題，縦軸は時間 (秒)，Eg の実行時間はソート)

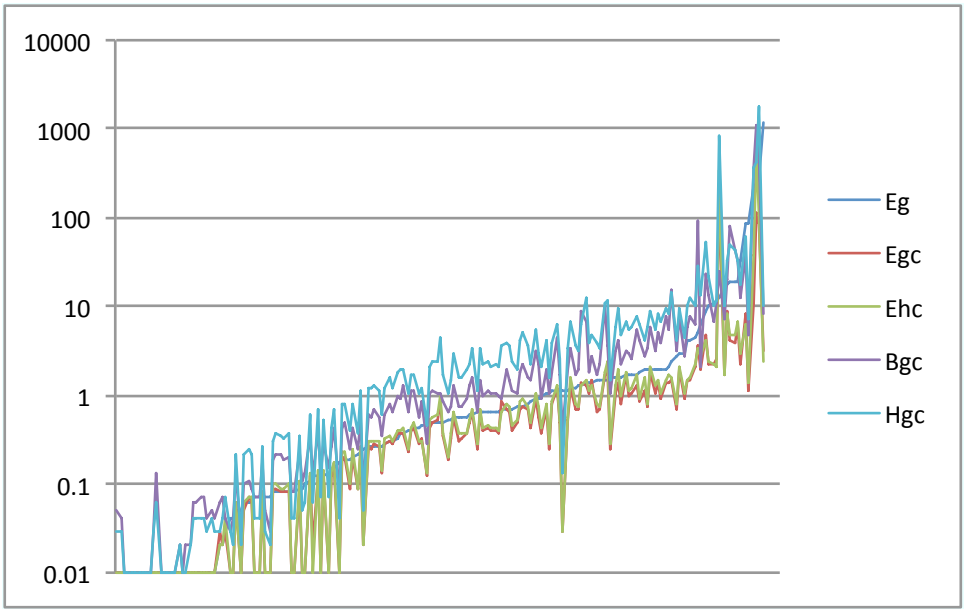


図 3.15: 実行時間 (文字列数：450)(横軸は問題，縦軸は時間 (秒)，Eg の実行時間はソート)

表 3.7: MiniSat によって解が得られた問題数 (総問題数 183)(括弧内は CLASP のみで解けた問題数と MiniSat のみで解けた問題数)

列数	すべて	450	400	300
Eg	183(-,-)	183(-,1)	181(1,9)	145(7,15)
Egc	183(-,-)	183(-,1)	182(1,4)	151(6,12)
Ehc	183(-,-)	183(-,1)	182(1,4)	148(6,13)
Bgc	183(-,-)	180(2,1)	178(2,6)	143(6,9)
Bhc	183(-,-)	180(2,0)	178(1,5)	141(6,7)
Hgc	183(-,-)	182(1,1)	180(2,8)	145(8,12)
Hhc	183(-,-)	183(-,2)	180(2,8)	143(7,12)
並列	183	183	183	157

表 3.8: CLASP と MiniSat の実行時間の相関係数

列数	すべて	450	400	300
Eg	1.00	0.14	0.27	0.36
Egc	0.63	0.15	0.22	0.45
Ehc	0.95	0.05	0.39	0.33
Bgc	0.98	0.55	0.67	0.51
Bhc	0.99	0.77	0.13	0.63
Hgc	0.97	0.72	0.17	0.05
Hhc	0.97	0.69	0.24	0.14

3.5.3.5.5 ソルバーの違いによる比較

コンペティションの結果を見ると，ＳＡＴソルバーの違いで解ける問題や解けない問題に違いが出ていくる．そのため，SAT ソルバーが異なると計算時間に違いが現れる可能性がある．本章では SAT ソルバーとして CLASP を用いて評価を行なってきたが，MiniSat2.2.0[70] を用いた実験も行った．利用したコンピュータは CLASP の場合と同じである．

3.5.3.5.3 で詳細に分析を行ったプログラム設定について，3.7 に 3.6 と同様の実験結果を載せた．括弧内は CLASP だけで解けた問題の数，MiniSat だけで解けた問題の数である．CLASP で解けた問題数を解が得られた問題数に加えれば，二つのソルバーで解けた問題となる．文字列数が 300 の場合を見ると，CLASP よりも MiniSat の方が問題が解けている．しかし，Egc が最もよく問題を解いているなど，プログラム設定と解けた問題数の関係は CLASP と MiniSat で等しいと考えている．3.8 は CLASP と MiniSat で解けた問題の実行時間を相関係数によって比較したものである．この表から文字列数が多い時は相関が高いことがわかる．文字列数が少ないときはかなりばらつきはあるが，概ね正の相関を示している．以上から，本実験で用いた最小無矛盾 DFA の解法について，CLASP と MiniSat の間には顕著な違いと判断した．

3.5.3.6　まとめ

本節では，SAT を用いた最小無矛盾 DFA 問題の解法に対して，２進表現を使った色の表現，クリークの状態に割り当てられない色に関する対称性除去，ハイパーグラフ彩色問題制約に関する新しい提案を行った．実験の結果，特にクリーク順を用いた対称性除去が効果的であり，他の手法も問題によっては効果的であることを示した．提案手法を組み合わせて，プログラムにバリエーションを作り，並列実行することで，従来手法に比べ，平均相対時間を 66.5%, 合計相対時間を 7.6% にすることができ，特に従来手法で実行時間の長い問題に有効であることを示した．

どのような事例に対して，どのような手法が効果的かを予測することは非常に難しい問題である．しかし，SAT ソルバーの効率化とともに，筆者らは SAT による最小無矛盾 DFA 問題へのアプローチはさらに効果的になるだろうと考えている．そのため，今後の課題として，さらに様々の手法を並列処理の枠組みで統合することがあげられる．

3.6　第 3 章のまとめ

本章では，最小無矛盾 DFA 問題を対象に MILP によるアプローチ，SAT によるアプローチを述べた．SAT によるアプローチでは，対称性除去のための最大クリーク問題の厳密解の利用，付加的な規則の導入による効率的な解法について述べた．MILP と SAT の解法を比べたところでは，現在の所，SAT による解法のほうが計算時間が短く，大きな問題を解ける．しかし，SAT を使った場合，問題によっては非常に時間がかかるため，付加的な規則の導入が重要であることがわかった．特に，SAT による解法では，対称性除去が非常に重要であることを示すことができた．

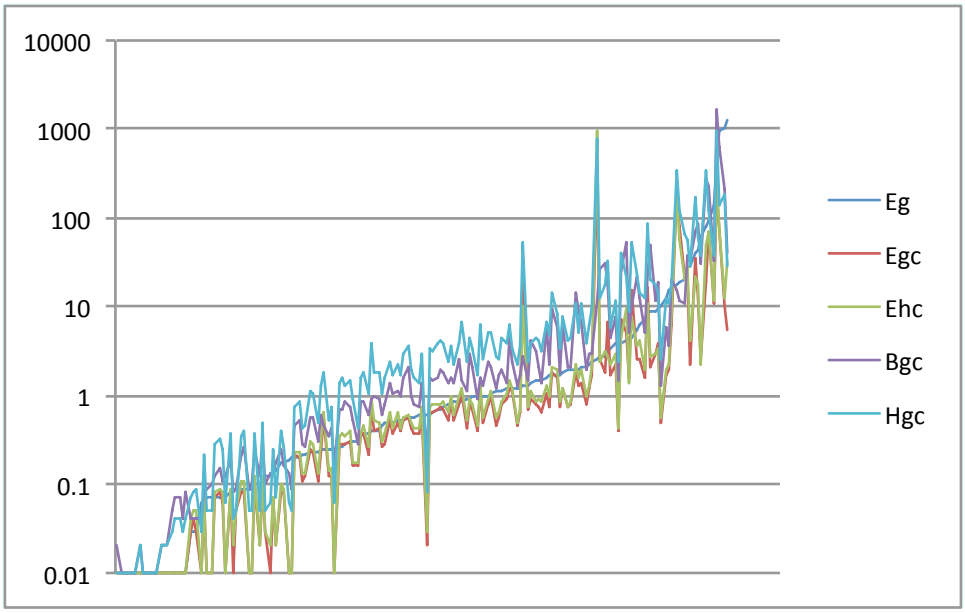


図 3.16: 実行時間 (文字列数：400)(横軸は問題，縦軸は時間 (秒)，Eg の実行時間はソート)

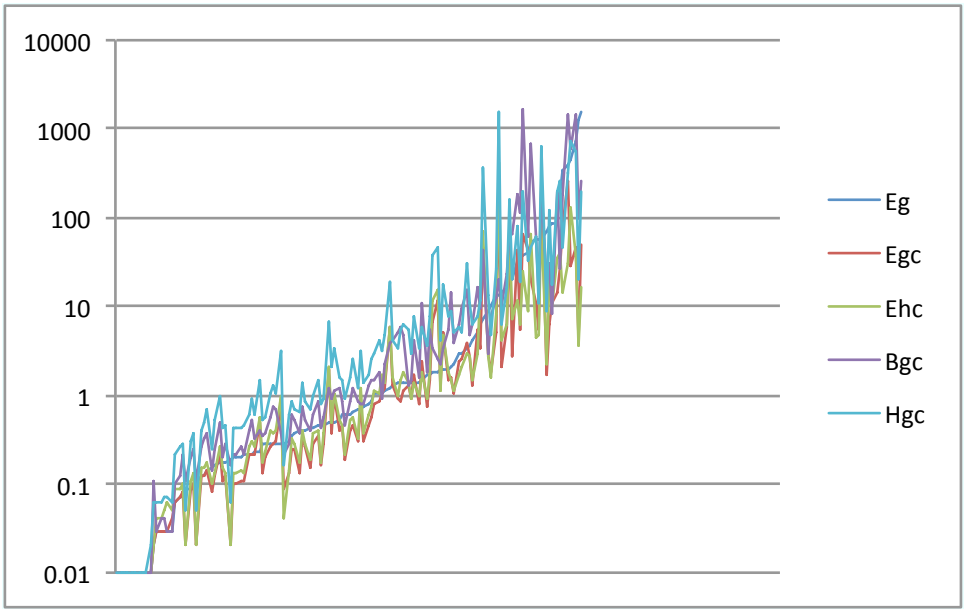


図 3.17: 実行時間 (文字列数：300)(横軸は問題，縦軸は時間 (秒)，Eg の実行時間はソート)

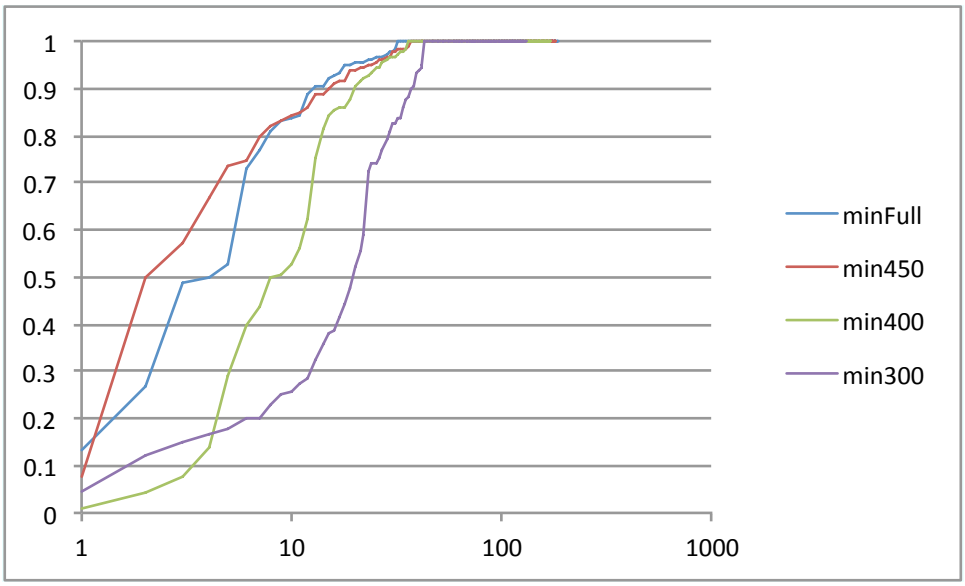


図 3.18: Egc に対する相対実行時間 (横軸は問題で，相対時間の短いものから順番に並べた)

第4章　走査型半導体露光装置における移動順最適化問題

4.1　はじめに

走査型半導体露光装置（スキャナー）における露光処理時間の短縮は，半導体の生産性を向上させる．そこで，本章はステージの移動順を最適化することでウェーハ一枚当たりの処理時間を短縮する．まずスキャナーの概要を図 4.1 に示す．これは，レチクルに描かれたパターンをウェーハの複数箇所に縮小投影露光する様子を示している．ウェーハステージは水平な平面を自由に移動するが，レチクルステージは水平な一方向のみ往復移動する．レチクルを通過した光は投影レンズを通してウェーハ面に縮小投影される．なお，露光はレチクルステージとウェーハステージが同期スキャンしているときに行われる．レチクルステージが装置正面に向かって移動するときを “上方向スキャン”，その逆を “下方向スキャン” とよぶ．このとき，1 回のスキャンで転写されたパターンを “ショット” とよぶ．

ウェーハ一枚を露光処理する工程は，ウェーハステージが所定の位置に移動して，ウェーハを取り付けるところから始まる．そして，前工程で転写されていたアライメントマークを顕微鏡で計測し，ウェーハの位置合わせ (アライメント) を行う．さらに，前工程で転写されたショットに対して重ね合わせ露光を行い，最後にウェーハを外す位置にウェーハステージを移動する．

本章における “移動順” とは，アライメントマークの計測順とショットの露光順およびそのスキャン方向を定めたものである．各移動にはその距離や位置関係およびスキャナーの性能によって定められた移動時間が生じる．本章では最短時間の移動順を見つける問題を移動順最適化問題 (MSOP: Movement Sequence Optimization Problem) とよぶ．

ところで，半導体露光装置には走査型以外にも一括型がある．一括型の露光装置はスキャン方向を考慮することなくショット間の移動時間を決定できるので，4.5 節で述べる巡回セールスマン問題 [72](TSP: Traveling Salesman Problem) に直接帰着でき，厳密な最短時間を求めることができた．一方，スキャナーついては進化型計算を使った手法 [95] が提案されているが，真に最短時間となる解を保証していない．本章は，モデル化の工夫により，MSOP を TSP に帰着し，TSP ソルバーを用いて真に最短時間となる移動順の解を得る．TSP は非常によく研究されている分野であり，多くの手法 [37] が考案されている．また，電子部品装着

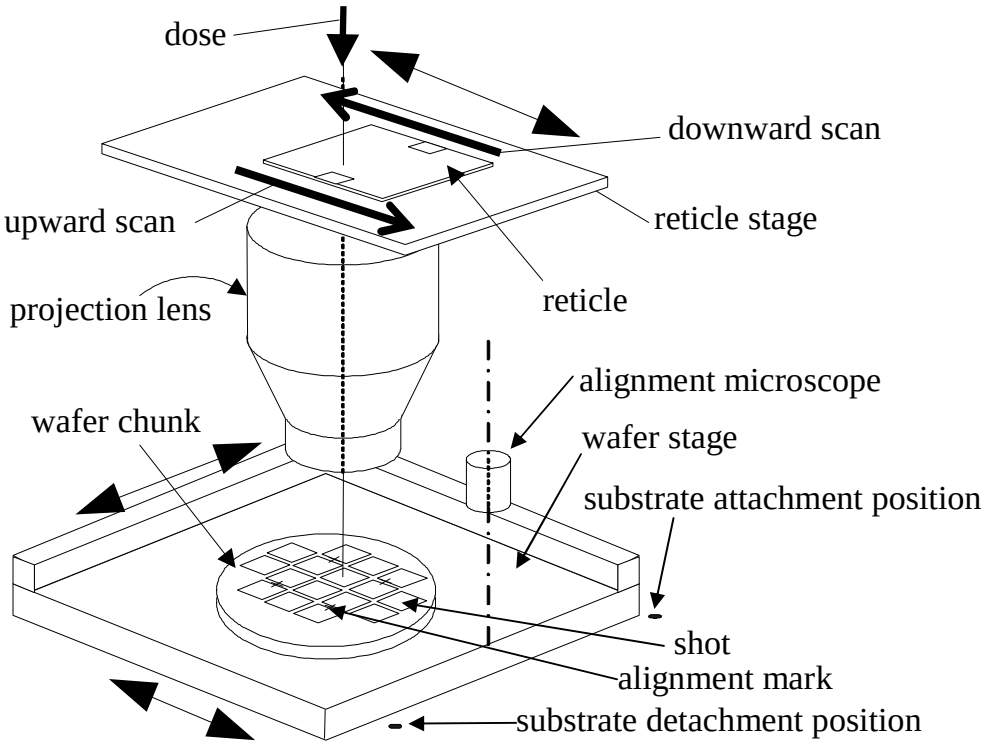


図 4.1: Scan-Typed Semiconductor Lithography

などの応用例 [28] も多く存在し，ソルバー [19] も公開されている．

本章では，4.2 節で MSOP を定義する．4.3 節では，MSOP を MILP によって記述し，この問題記述が解きにくいことについて述べる．4.5 節では TSP を紹介し，4.6 節では MSOP を TSP に帰着する方法を述べる．4.7 節では，提案手法に関する実験結果を示して考察する．4.8 節では，MSOP のパラメータの違いによるバリエーションについて述べ，幾つかの実行結果を示す．そして，本章の最後として，4.9 節で本章をまとめる．

4.2　移動順最適化問題 (MSOP)

移動順を記述するのに必要な位置は，ウェーハ取り付け位置，アライメント計測位置，露光位置，ウェーハ取り外し位置の 4 種類であるので，それぞれを，図 4.2 に示すように，開始頂点，アライメント頂点，露光頂点，終端頂点とする．移動順はこれらの頂点の列として表現される．このとき，露光頂点はスキャン方向を表現するため，いわゆる巡回セールスマン問題における頂点とは異なる．辺は頂点を結んだものであり，移動可能であることを示す．辺には頂点から頂点に移動する際の移動時間が与えられている．

開始頂点 b は移動順の先頭となる．アライメントは精度の粗いものから細かいものへと段階に分けて行われるので，アライメント頂点を G 個のグループ $(G \geq 1)$ に分け，$A_g, g \in$

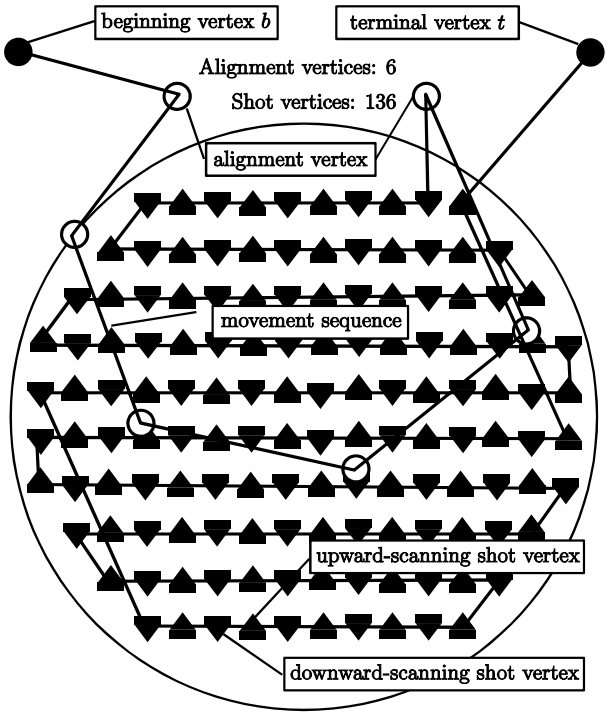


図 4.2: An Example of Optimal Solution for MSOP

$\{1,\ldots,G\}$ をアライメントグループと表記する．各アライメントグループは任意の数のアライメント頂点を含む．A_g 内のアライメント頂点を全て訪問した後にのみ，A_{g+1} 内のアライメント頂点を訪問することが許される．よって，開始頂点からは A_1 のアライメント頂点に対してのみ辺が定義される．そして，A_g に属するアライメント頂点からは A_g および A_{g+1} に対する辺のみが定義される．ただし，A_G 内のアライメント頂点については，全てのアライメント頂点を訪問した後，露光頂点へ移動することが認められる．

露光頂点集合を S で表現する．露光頂点はスキャン方向を持つ頂点であるため，露光頂点を訪問する場合は同時にスキャン方向を決定する必要がある．図 4.2 では，5 角形でスキャン方向を表している．終端頂点 t は移動順の末端となる．

まとめると，次の移動時間が定義される．

w_{bi}: b から $i \in A_1$.
w_{ij}: $i \in A_g$ から $j \in A_{g+1}$. $g = 1,\ldots,G-1$.
w_{ij}: $i \in A_g$ から $j \in A_g \setminus \{i\}$. $g = 1,\ldots,G$.
w^u_{ij}, w^d_{ij}: $i \in A_G$ から各スキャン方向についての $j \in S$.
$w^{uu}_{ij}, w^{dd}_{ij}, w^{ud}_{ij}, w^{du}_{ij}$: 各スキャン方向についての $i \in S$ から各スキャン方向についての $j \in S \setminus \{i\}$.
w^u_{it}, w^d_{it}: 各スキャン方向についての $i \in S$ から t.

ここで，下付添え字は移動頂点を示す．露光頂点についてはスキャン方向を表現するために，移動元・移動先頂点のスキャン方向を，上付き添え字，u(上方向)，d(下方向) により示す．

アライメント頂点間の移動ではウェーハステージだけが使われるので，$i \in A_g, j \in A_g \setminus \{i\}$ について，$w_{ij} = w_{ji}$ となる．これに対して，露光頂点が関係した移動時間については，レチクルステージの移動も移動時間に関係する．そのため，スキャン方向により移動時間が異なったり，逆方向と移動時間が等しくない場合がある．MSOP とは，開始頂点 b から $A_1 \dots A_G$ の順番でアライメント頂点を全て訪問し，さらに露光頂点を全て訪問してから, 最後に終端頂点 t を訪問する条件の下で，総移動時間を最小にする頂点列および露光頂点のスキャン方向を決定する最短経路問題として定義される．

4.3　MSOP の MILP による定式化

MSOP は MILP によって定式化できる．次にこの MILP を示す．ここで全ての変数 e は 01 整数変数である．

$$
\begin{aligned}
\text{minimize}\quad & \sum_{i\in A_1} w_{bi}e_{bi} \\
& +\sum_{g=1}^{G-1}\sum_{i\in A_g}\sum_{j\in A_{g+1}} w_{ij}e_{ij} \\
& +\sum_{g=1}^{G}\sum_{i\in A_g}\sum_{j\in A_g\setminus\{i\}} w_{ij}e_{ij} \\
& +\sum_{i\in A_G}\sum_{j\in S}(w^u_{ij}e^u_{ij}+w^d_{ij}e^d_{ij}) \\
& +\sum_{i\in S}\sum_{j\in S\setminus\{i\}}(w^{uu}_{ij}e^{uu}_{ij}+w^{dd}_{ij}e^{dd}_{ij}+w^{ud}_{ij}e^{ud}_{ij}+w^{du}_{ij}e^{du}_{ij}) \\
& +\sum_{i\in S}(w^u_{it}e^u_{it}+w^d_{it}e^d_{it}) \\
\text{s.t.}(1)\quad & \sum_{i\in A_1} e_{bi}=1 \\
(2)\quad & e_{bj}+\sum_{i\in A_1\setminus\{j\}} e_{ij}=1, \forall j\in A_1 \\
(3)\quad & \sum_{j\in A_g\setminus\{i\}} e_{ij}+\sum_{j\in A_{g+1}} e_{ij}=1, \forall g\in\{1,\dots,G-1\}, \forall i\in A_g \\
(4)\quad & \sum_{i\in A_g} e_{ij}+\sum_{i\in A_{g+1}\setminus\{j\}} e_{ij}=1, \forall g\in\{1,\dots,G-1\}, \forall j\in A_{g+1} \\
(5)\quad & \sum_{j\in A_G\setminus\{i\}} e_{ij}+\sum_{j\in S}(e^u_{ij}+e^d_{ij})=1, \forall i\in A_G \\
(6)\quad & \sum_{i\in A_G}(e^u_{ij}+e^d_{ij})+\sum_{i\in S\setminus\{j\}}(e^{uu}_{ij}+e^{dd}_{ij}+e^{ud}_{ij}+e^{du}_{ij})=1, j\in S \\
(7)\quad & \sum_{j\in S\setminus\{i\}}(e^{uu}_{ij}+e^{dd}_{ij}+e^{ud}_{ij}+e^{du}_{ij})+e^u_{it}+e^d_{it}=1, i\in S \\
(8)\quad & \sum_{i\in A_G} e^u_{ij}+\sum_{i\in S\setminus\{j\}}(e^{uu}_{ij}+e^{du}_{ij}-e^{uu}_{ji}-e^{ud}_{ji})-e^u_{jt}=0, j\in S \\
(9)\quad & \sum_{i\in A_G} e^d_{ij}+\sum_{i\in S\setminus\{j\}}(e^{ud}_{ij}+e^{ud}_{ij}-e^{dd}_{ji}-e^{du}_{ji})-e^d_{jt}=0, j\in S \\
(10)\quad & \forall g\in\{1,\dots,G\} \text{ に対するエッジ集合} \\
& E_g=\{(i,j)|e_{ij}=1, i\in\forall A_g, \forall j\in A_g\setminus\{i\}\} \text{ について,} \\
& E_g \text{が部分巡回路を含まない} \\
(11)\quad & E^{uu}_S=\{(i,j)|e^{uu}_{ij}=1, i\in\forall S, \forall j\in S\setminus\{i\}\} \\
& E^{dd}=\{(i,j)|e^{dd}_{ij}=1, i\in\forall S, \forall j\in S\setminus\{i\}\} \\
& E^{ud}=\{(i,j)|e^{ud}_{ij}=1, i\in\forall S, \forall j\in S\setminus\{i\}\} \\
& E^{dd}=\{(i,j)|e^{dd}_{ij}=1, i\in\forall S, \forall j\in S\setminus\{i\}\} \text{ について,} \\
& E^{dd}\cup E^{dd}\cup E^{ud}\cup E^{dd} \text{が部分巡回路を含まない}
\end{aligned}
\tag{4.1}
$$

この問題記述において，$e=1$ となる変数 e は経路として選択されるエッジを示している．

そのため，目的関数は選ばれたエッジの重みの総和であり，それを最小化する．(1) 式は開始頂点からアライメントグループ A_1 へのエッジが一つ選ばれることを表現する．(2) 式は，アライメントグループ A_1 の前の頂点は開始頂点か他の A_1 の頂点であることを表現する．(3) 式は，アライメントグループ A_g の次の頂点は A_g の他の頂点か A_{g+1} の頂点であることを表現する．(4) 式は，アライメントグループ A_{g+1} の前の頂点は A_g の頂点か A_{g+1} の他の頂点から一つ選ばれることを示す．(5) 式は，A_G の次の頂点は A_G の他の頂点か露光頂点であることを表現する．(6) 式は，露光頂点の前の頂点は A_G の頂点か別の露光頂点であることを示す．(7) 式は，露光頂点の前の頂点は別の露光頂点か終端頂点であることを示す．(1) 式によって，開始頂点から A_1 へのエッジが一つ選ばれるため，A_1 から A_2,A_{G-1} から A_G，A_G から露光頂点，露光頂点から終端頂点へのエッジが一つ選ばれるようになる．これによって，開始頂点から終端頂点への経路が生成されることになる．(8),(9) 式は各露光頂点でスキャン方向を一致させることを表現している．これにより，入ってくるエッジのスキャン方向と出ていくエッジのスキャン方向を一致させる．例えば，ある露光頂点 i について，$e_{ji}^{uu} = 1$ である頂点 j が存在するならば，任意の頂点 k について，e_{ik}^{du}, e_{ik}^{dd} は 0 にならなければならない．(10),(11) 式は，アライメントグループ内および露光頂点集合内で部分巡回路ができるのを防ぐ条件である．部分巡回路ができてしまうと，開始頂点から終端頂点への経路上にない頂点が存在することになる．

この問題は巡回セールスマン問題を解くために起きる問題と同じである．部分巡回路の除去については，様々な解法が考えられており，それを実装することで部分巡回路の問題を解決することができる．しかしながら，巡回セールスマン問題では優れたソルバーが開発されており，それを用いたほうが良いことが多いと考えられる．つまり，MSOP を巡回セールスマン問題に変換することができるならば，優れたアルゴリズムやデータ管理機構を利用することができる．そこで，次節では MSOP を巡回セールスマン問題に変換する方法について述べる．

4.4　MSOP の MaxSAT による定式化

4.3 節で述べた MILP による定式化と同様に partial weighted MaxSAT によっても MSOP を定式化することができる．partial weighted MaxSAT とは，SAT における選言節に 0 以上の重みを不可することができる SAT であり，選言節は必ず満たさなければならないハード条件と満たしたほうが良いソフト条件に分けられる．条件を満足しないソフト場である選言節の重みの総和を最小化するような論理変数への値の割当を行う．次に示す partial weighted MaxSAT による MSOP の記述では，$w : L$ という表記を使う．ここで，L は選言節であり，w はその重みとなる．重みを持たない選言節は，ハード条件である．

$$
\begin{aligned}
&(0)\quad w_{bi}:\neg e_{bi}, i\in A_1\\
&\quad w_{ij}:\neg e_{ij}, g\in\{1,\ldots,G-1\}, i\in A_g, j\in A_{g+1}\\
&\quad w_{ij}:\neg e_{ij}, g\in\{1,\ldots,G\}, i\in A_g, j\in A_g\setminus\{i\}\\
&\quad w^u_{ij}:\neg e^u_{ij}, w^d_{ij}:\neg e^d_{ij}, i\in A_G, j\in S\\
&\quad w^{uu}_{ij}:\neg e^{uu}_{ij}, w^{dd}_{ij}:\neg e^{dd}_{ij}, w^{ud}_{ij}:\neg e^{ud}_{ij}, w^{du}_{ij}:\neg e^{du}_{ij}, i\in S, j\in S\setminus\{i\}\\
&(0)\quad w^u_{it}:e^u_{it}, w^d_{it}:e^d_{it}, i\in S\\
&(1)\quad \bigvee_{i\in A_1} e_{bi}\\
&(2)\quad e_{bj}\vee\bigvee_{i\in A_1\setminus\{j\}} e_{ij}, \forall j\in A_1\\
&(3)\quad \bigvee_{j\in A_g\setminus\{i\}} e_{ij}\vee\bigvee_{j\in A_{g+1}} e_{ij}, \forall g\in\{1,\ldots,G-1\}, \forall i\in A_g\\
&(4)\quad \bigvee_{i\in A_g} e_{ij}\vee\bigvee_{i\in A_{g+1}\setminus\{j\}} e_{ij}, \forall g\in\{1,\ldots,G-1\}, \forall j\in A_{g+1}\\
&(5)\quad \bigvee_{j\in A_G\setminus\{i\}} e_{ij}\vee\bigvee_{j\in S}(e^u_{ij}\vee e^d_{ij}), \forall i\in A_G\\
&(6)\quad \bigvee_{i\in A_G}(e^u_{ij}\vee e^d_{ij})\vee\bigvee_{i\in S\setminus\{j\}}(e^{uu}_{ij}\vee e^{dd}_{ij}\vee e^{ud}_{ij}\vee e^{du}_{ij}), j\in S\\
&(7)\quad \bigvee_{j\in S\setminus\{i\}}(e^{uu}_{ij}\vee e^{dd}_{ij}\vee e^{ud}_{ij}\vee e^{du}_{ij})\vee e^u_{it}\vee e^d_{it}, i\in S\\
&(8)\quad \bigvee_{i\in A_G} e^u_{ij}+\bigvee_{i\in S\setminus\{j\}}(e^{uu}_{ij}\vee e^{du}_{ij}\vee\neg e^{uu}_{ji}\vee\neg e^{ud}_{ji})\vee\neg e^u_{jt}, j\in S\\
&\quad \bigvee_{i\in A_G}\neg e^u_{ij}+\bigvee_{i\in S\setminus\{j\}}(\neg e^{uu}_{ij}\vee\neg e^{du}_{ij}\vee e^{uu}_{ji}\vee e^{ud}_{ji})\vee e^u_{jt}, j\in S\\
&(9)\quad \bigvee_{i\in A_G} e^d_{ij}+\bigvee_{i\in S\setminus\{j\}}(e^{ud}_{ij}\vee e^{ud}_{ij}\vee\not{e}^{dd}_{ji}\vee\neg e^{du}_{ji})\vee\neg e^d_{jt}, j\in S\\
&\quad \bigvee_{i\in A_G}\neg e^d_{ij}+\bigvee_{i\in S\setminus\{j\}}(\neg e^{ud}_{ij}\vee\neg e^{ud}_{ij}\vee e^{dd}_{ji}\vee e^{du}_{ji})\vee e^d_{jt}, j\in S\\
&(10)\quad \forall g\in\{1,\ldots,G\}\text{ に対するエッジ集合}\\
&\quad E_g=\{(i,j)|e_{ij}=T, i\in\forall A_g, \forall j\in A_g\setminus\{i\}\}\text{ について,}\\
&\quad E_g\text{が部分巡回路を含まない}\\
&(11)\quad E^{uu}_S=\{(i,j)|e^{uu}_{ij}=T, i\in\forall S, \forall j\in S\setminus\{i\}\}\\
&\quad E^{dd}=\{(i,j)|e^{dd}_{ij}=T, i\in\forall S, \forall j\in S\setminus\{i\}\}\\
&\quad E^{ud}=\{(i,j)|e^{ud}_{ij}=T, i\in\forall S, \forall j\in S\setminus\{i\}\}\\
&\quad E^{dd}=\{(i,j)|e^{dd}_{ij}=T, i\in\forall S, \forall j\in S\setminus\{i\}\}\text{ について,}\\
&\quad E^{dd}\cup E^{dd}\cup E^{ud}\cup E^{dd}\text{が部分巡回路を含まない}
\end{aligned}
\tag{4.2}
$$

4.3 節で示した式と対応付けて示すため，細部の説明は省略する．この定式化に関連して予備実験を行った．予備実験では，TSPLIB[92] より非対称巡回セールスマン問題のベンチマー

表 4.1: ATSP の初期問題に対する MILP と partial weighted MaxSAT の計算時間の違い (秒)

ベンチマーク	都市数	目的関数値	Cplex 12.0	Clasp 2.0	Sat4j
br17	17	22	0.04	1234	216
ftv33	33	1217	0.07	≥3 時間	≥3 時間
ftv35	35	1420	0.09	≥3 時間	≥3 時間

ク問題を選び，部分巡回路の処理を場合の定式化について，MILP および partial weighted MaxSAT によって比較実験を行った．V を都市の集合，x_{ij} を都市 i から j へ移動を表す変数とし，d_{ij} を都市 i から j への移動距離とする．まず， 部分巡回除去を含まない巡回セールスマン問題の MILP による定式化は次のように与えられる．

$$\begin{array}{ll} \text{minimize} & \sum_{i\in V}\sum_{j\in V\setminus\{i\}} d_{ij}x_{ij} \\ (1) & \sum_{j\in V\setminus\{i\}} x_{ij} = 1, i \in V \\ (2) & \sum_{j\in V\setminus\{i\}} x_{ji} = 1, i \in V \\ (3) & x_{ij} + x_{ji} \leq 1, i \in V, j \in V \setminus \{i\} \\ (4) & x_{ij} \in \{0,1\} \end{array} \tag{4.3}$$

ここで，(1) は都市 i から出ていく道についての条件であり，(2) は都市 i に入る道についての条件である．(3) は二つの都市間で選ばれる道が高々一本であることを示している．この MILP による定式化に対する，partial weighted MaxSAT による定式化は次のように与えられる．

$$\begin{array}{ll} (0) & d_{ij} : \neg x_{ij}, i \in V, j \in V \setminus \{i\} \\ (1) & \bigvee_{j\in V\setminus\{i\}} x_{ij}, i \in V \\ (1) & \neg x_{ij} \vee \neg x_{ik}, i \in V, j \in V \setminus \{i\}, k \in V \setminus \{i,j\} \\ (2) & \bigvee_{j\in V\setminus\{i\}} x_{ji}, i \in V \\ (2) & \neg x_{ji} \vee \neg x_{ki}, i \in V, j \in V \setminus \{i\}, k \in V \setminus \{i,j\} \\ (3) & \neg x_{ij} \vee \neg x_{ji}, i \in V, j \in V \setminus \{i\} \\ (4) & x_{ij} \in \{T,F\} \end{array} \tag{4.4}$$

これら，MILP と partial weighted MaxSAT による定式化の効率性の違いを確認するため，TSPLIB[92] にある三つの問題 ATSP (Asymmetric TSP: 非対称 TSP, 都市間の移動距離が方向によって異なる場合を認める) で計算時間を測定した．ソルバーとして，MILP については CPLEX 12.0 を使い，partial weight MaxSAT については clasp 2.0 および Sat4j 2.3.1[89] を用いた．計算機は Quad Core Amd 2.7Gz 搭載の Linux マシンで行った．表 4.1 に結果を示す．

MSOP は ATSP の特殊な場合と考えられるので，ATSP を効率良く解けるかどうかがソル

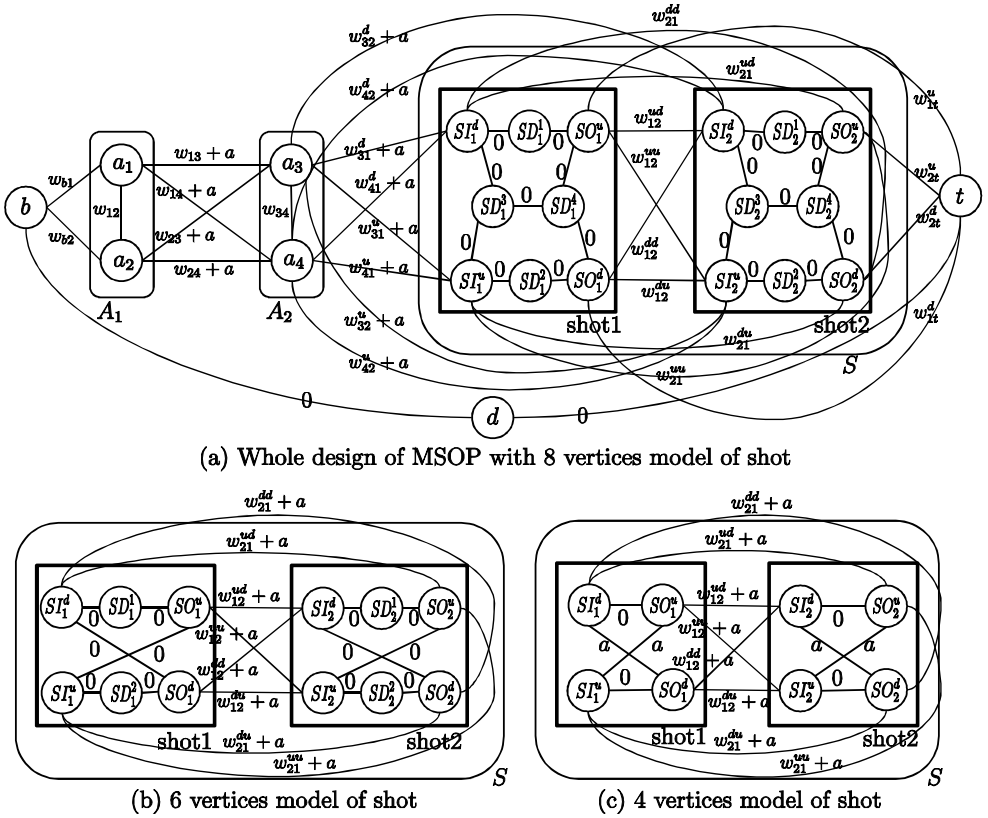
バーを選ぶ基準となる．巡回セールスマン問題を解く場合は，特に MILP によるアプローチでは分枝カット法の実装により，整数解が得られなくても，LP 解を見ることによって部分巡回路除去の制約式を入れることができるが，partial weighted MaxSAT によるアプローチでは，解を得た後でなければ，部分巡回路除去の制約条件を入れることができない．そのため，MILP で整数解を得るまでの時間と partial weighted MaxSAT によって解を得るまでの時間を比較することで，MSOP での実行時間を予測できると考えられる．表 4.1 の結果からは，partial weight MaxSAT による解法は，MILP による解法に比べ，効率が悪いことがわかる．このため，以降の節では MSOP に対する解法としては，MILP をベースとした手法だけを考えることにする．

4.5 巡回セールスマン問題

TSP は，複数の頂点と頂点間のコストが与えられたとき，全ての頂点をちょうど一度だけ訪問する巡回路の中で，総コストが最小の巡回路を求める問題である．距離の与え方の違いにより，TSP は対称型 TSP(STSP: Symmetric TSP) と非対称型 TSP(ATSP: Asymmetric TSP) に分類される．二つの頂点間の辺について，STSP ではコストが双方向で等しく，ATSP では異なる場合が存在する．しかし，これら二つのモデルは相互に変換可能であり，本質的な違いはない．TSP は NP 困難問題 [37] であるが，厳密解法および近似解法について多くの研究が行われている．厳密解法の多くは，LP ベース分枝限定法を使っており，整数計画法を使って MSOP を解く場合と本質的に等しい．近似解法としては Lin-Kernihan 法 [65] が有名である．これは実行可能解を局所探索によって最適化する手法であり，２本の同じ長さの部分経路をスワップすることによって探索の近傍を定義している．様々な効率的な実現方法が研究され [39]，様々なグラフにおいて良い近似解が得られている．さらに，STSP はソルバーが公開されており，MSOP を STSP に帰着させることで，最新の研究結果を利用できるという利点がある．これに対して，整数計画法を独自に使う場合は，定式化の工夫に加えて，新たにカット生成等のプログラムを書く必要がある．

4.6 MSOP の STSP による記述

本章では，図 4.3 の例を使って，MSOP を STSP に変換する方法を説明する．この例は，アライメントグループ $A_1 = \{1, 2\}, A_2 = \{3, 4\}$ および露光頂点集合 $S = \{1, 2\}$ の場合を示している．□は露光頂点を表現し，○は STSP での頂点を表現する．頂点 b は開始頂点，頂点 t は終端頂点，頂点 d はダミー頂点である．頂点 $a_i, i \in A_g, g \in \{1, \ldots, G\}$ は A_g に属するアライメント頂点を意味する．一つのアライメント頂点は一つの STSP での頂点に対応するが，一つの露光頂点は複数の STSP での頂点を使って表現される．露光頂点 $i \in S$ について，



(a) Whole design of MSOP with 8 vertices model of shot

(b) 6 vertices model of shot

(c) 4 vertices model of shot

図 4.3: STSP model for MSOP

頂点 SI_i^u, SI_i^d はそれぞれ上方向・下方向スキャンを行う際の入口となる頂点であり，頂点 SO_i^u, SO_i^d は出口となる頂点である．これによって，二つの露光頂点の移動方向およびスキャン方向の違いによる 4×2 通りの移動時間を表現する．頂点 $SD_i^1, SD_i^2, SD_i^3, SD_i^4$ は shot$i$ 内部に設けたダミー頂点である．本章では，露光頂点の表現について，8 頂点モデル,6 頂点モデル, 4 頂点モデルを提案する．辺には STSP を解くために使われるコストが付与されている．辺を持たない頂点間の移動はできない．なお，頂点 i, j 間の辺を $\{i, j\}$ と書く．

4.6.1 開始頂点と終端頂点

開始頂点と終端頂点は，STSP においてそれぞれ一つの頂点として扱う．開始頂点 b と終端頂点 t を結ぶことにより，MSOP の解は巡回路となる．STSP として記述した場合，必ずこれら二つの頂点を結ぶ辺が選ばれるようにすることが必要である．本章ではこれをダミー頂点 d を導入することで解決する．TSP は全頂点を訪問して元へ戻る道順を求める問題なので，その解では全頂点からちょうど 2 本の辺が出ている．d は b と t 以外とは辺で結ばれていないので，辺 $\{t, d\}$ と $\{d, b\}$ は巡回路上の辺となる．

4.6.2 アライメント頂点

図 4.3(a) のように，一つのアライメント頂点 i は a_i で表現する．前節より d と b は辺で結ばれているので，b から出るもう一本の辺は A_1 に属するアライメント頂点の一つと結ばれる．このため，開始頂点と A_1 に属する辺のコストには，後述するアライメントグループ間の辺についてのコストのように重みを加える必要はなく，コスト $w_{bi}, i \in A_1$ を割り付ける．アライメント頂点に関して，MSOP の実行可能解とは，例えば，A_1 のアライメント頂点を全て訪問してから A_2 のアライメント頂点を訪問する巡回路をさす．そのため，A_1 と A_2 の間の辺 $\{i,j\}, i \in A_1, j \in A_2$ (および，A_2 と露光頂点間の辺 $\{i,j\}, i \in A_2, j \in S$) は一本だけ選ばれなければならない．また，$A_2$ についても全てのアライメント頂点を訪問してからでなければ露光頂点を訪問できない．このような巡回路を生成するため，A_1 と A_2 間および A_2 と露光頂点間の辺のコストに重み a を加える．STSP は巡回路の総コストを最小化する問題であるので，正の重み a を加えられた辺は選びにくくなる．重み a の決定方法については 4.6.4 節で説明する．

4.6.3 露光頂点

8 頂点モデルでは，露光頂点を図 4.3(a) のように表現する．$SD_1^1, SD_1^2, SD_1^3, SD_1^4$ などの頂点はショット内部での経路を限定する機能を果たす．これらの頂点が存在することで，TSP の全ての頂点を訪問するという条件から，shot1 内を通る経路は，経路 $\{SI_1^d, SD_1^1, SO_1^u, SD_1^4, SD_1^3, SI_1^u, SD_1^2, SO_1^d\}$ と経路 $\{SI_1^u, SD_1^2, SO_1^d, SD_1^4, SD_1^3, SI_1^d, SD_1^1, SO_1^u\}$ の 2 通りとなる．最初の経路は下方向スキャン，2 番目の経路は上方向スキャンを表す．露光頂点に関して MSOP の実行可能解とはショットを必ずスキャンする巡回路である．一般に TSP を解く時間は頂点数の円順列に比例して増加すると考えられる．MSOP では露光頂点の数が他の頂点の数に比べて圧倒的に多いので，より少ない頂点数で露光頂点を表現できれば，STSP を解く時間が短縮される可能性がある．図 4.3(b),(c) に，一つの露光頂点を 6 頂点および 4 頂点で表現したモデルを示す．(b) の 6 頂点モデルでは，スキャンを表すために辺 $\{SI_1^d, SO_1^d\}$ および $\{SI_1^u, SO_1^u\}$ のどちらかが巡回路上の辺となる．必ずどちらかが選ばれるようにショットから外に出る 4 本の辺に重み a を加え，同時に 4 本選べないようにする．(c) の 4 頂点モデルでは，さらにスキャンを表すために $\{SI_1^d, SO_1^d\}$ と $\{SI_1^u, SO_1^u\}$ のどちらか一つだけが巡回路上の辺となる．これら 2 本の辺が同時には選ばれないように，$\{SI_1^d, SO_1^d\}$ および $\{SI_1^u, SO_1^u\}$ に重み a を加える．露光頂点におけるスキャン方向は，STSP において 3 頂点または 2 頂点で表現することも可能である．この場合，SI_1^d と SO_1^u および SI_1^u と SO_1^d が区別されないので，shot1 と shot2 の間の移動時間において，$w_{12}^{uu} = w_{21}^{dd}$,$w_{12}^{ud} = w_{21}^{ud}$, $w_{12}^{du} = w_{21}^{du}$,$w_{12}^{dd} = w_{21}^{uu}$ という制約が必要となり，適用可能な問題が限定されることになる．

4.6.4　重み a の決定

アライメント頂点および露光頂点を 6 頂点とするモデル，並びに露光頂点を 4 頂点とするモデルにおいては，重み a を導入している．このとき，STSP の解が MSOP の実行可能解となるためには，アライメントグループ内の全ての頂点は次のアライメントグループに移動する前に全て訪問することと，露光頂点はスキャンされることが必要である．前者についてはアライメントグループ間で一つの辺だけが選ばれればよい．後者については，一つのショットからは 2 本の辺が出て他のショットに結びつくこと（6・4 頂点モデル）とショット内部の上下を結ぶ辺がちょうど一本選ばれること（4 頂点モデル）が満たされればよい．アライメントグループ間については最低でも一本は選ばれること，ショットからは最低 2 本の辺が出ること（6・4 頂点モデル），ショットから出る辺と上下を結ぶ辺は最低 3 本である（4 頂点モデル）ことから，巡回路上の重みのついた辺の数を最小化することで STSP の解が MSOP の実行可能解を表すことになる．このため，STSP の最適解において重み a が与えられた辺の数が最小となるように a を決定する必要がある．重み a が与えられた辺の最小数はアライメントグループ数，露光頂点数より求めることができるので，これを利用して a を決定する．巡回路における重み a を持つ辺の数の最小数を M とする．具体的には，$M = G$(8 頂点モデル),$M = G + |S| - 1$(6 頂点モデル), $M = G + 2|S| - 1$(4 頂点モデル) である．今，STSP を解いたとき，重み a を持つ辺が N 本選ばれたとする．$N = M$ ならば，STSP の解が与える巡回路は MSOP の実行可能解である．最適な MSOP の移動時間を Y^*,$a = 0$ のときの STSP の最適解の目的関数値を $\overline{Y}$ とおく．このとき，次の定理が存在する．

定理　*MSOP* の実行可能解は *STSP* の実行可能解である．

証明　*MSOP* における，ある頂点の前後の移動時間の全組み合わせは，*STSP* においても同様に表現されていることから証明される．*(*証明終*)*

定理　$a > Y^* - \overline{Y}$ ならば，*STSP* の最適解は *MSOP* の実行可能解を与える．

証明　*STSP* の目的関数値を $Y + aN$ とする．任意の $N \geq M+1$ に対して,$Y^* + aM < Y + aN$ となるような $Y^* + aM$ を目的関数値に持つ *STSP* の解は *MSOP* の実行可能解を与える．Y の下限値は $\overline{Y}$，N の最小値は $M + 1$ であるから，$Y^* + aM < \overline{Y} + a(M + 1)$ が成り立つように a を決定すれば，*STSP* の最適解は *MSOP* の実行可能解を与える．*(*証明終*)*

4.6.5　MSOP のヒューリスティック解法

4.6.4 節の定理より，適切な a を与えた STSP の最適解から MSOP の最適解を得ることができる．しかし，事前に Y^* を求めることは不可能であり，$\overline{Y}$ も STSP の最適解を求めるこ

とになるため，事実上困難である．そこで重み a を高速なヒューリスティック解法により求められる MSOP の実行可能解を使うことにより決定する．本章では，ヒューリスティック解法により求めた MSOP の総移動時間 Y' を使い，$Y' > Y^* - \overline{Y}$ となることから，$a = Y'$ とした．

本章では次のヒューリスティック解法を用いた．まず開始頂点から A_1 の頂点の中で最短時間で移動可能なアライメント頂点を訪問する．以降，移動順の規則に沿って，順次，最短時間で訪問可能な頂点に移動することを繰り返し，MSOP の実行可能解を得る．これは，人手で行われる従来の方法に他ならない．

最後に，各露光頂点のモデルにおける STSP の最適解における目的関数値を示す．これは，各辺の重みが 0, 移動時間, 移動時間 $+a$ のいずれかであり，選ばれなければならない重みを持つ辺の数から求められる．

8 頂点モデル: $Y^* + aG$
6 頂点モデル: $Y^* + a(G + |S| - 1)$
4 頂点モデル: $Y^* + a(G + 2|S| - 1)$

4.7 数値実験

本章では，STSP を使った数値実験について述べる．STSP のソルバーとしては，最適解を求めるために Concorde[19] を，準最適解を求めるために Linkern[19] を用いた．実験に用いた PC は Intel Core 2 quad,2.8GHz を CPU として搭載し，メモリは 4GB である．本数値実験の目的は二つある．第一は，最適解が求められるまでの計算時間を評価することである．計算時間は，露光頂点に関する 3 種類の表現モデルごとに測定した．第二は，最適解と準最適解で求められる総移動時間を比較することにより，最適解の重要性を評価することである．このために，TSP の準最適解法である Linkern を使う．数値実験では 4 種類の頂点集合モデルを用いた．表 4.2 は各頂点集合の頂点数およびタイプを示している．これらは実際に用いられている中でも露光頂点数の多いものを選んでいる．表 4.3 と表 4.4 は，二つの露光頂点 shot$i$,shot$j$ の移動方向およびスキャン方向に関して，“=” は移動時間が等しいことを表し，“UK” は必ずしも等しくないことを表している．表 4.2 で示したタイプ “sym” は表 4.3 に，“asym” は表 4.4 に記す．4.6.3 節で述べたように，“sym” では露光頂点の STSP 表現で SI_1^d と SO_1^u および SI_1^u と SO_1^d を区別する必要がない．そのため，STSP においては実質的に露光頂点を少ない頂点数で表現できることになり，解が求めやすくなると考えられる．

まず，Concorde で解いた場合の MSOP の解と実行時間を表 4.5 に示す．各々の頂点集合モデルにおいて，上段は計算時間を，かっこ内は Concorde での処理タスク数（分枝限定法における分枝数）を示す．最適解に到達できなかった M4 のモデルを除けば，計算時間は 691 秒〜6864 秒を費やしている．中段は求められた MSOP の解（1 ウェーハあたりの露光処理

表 4.2: Data Specification (number of verteces)

Model	b	$\|A_1\|$	$\|A_2\|$	$\|S\|$	t	Type
M1	1	2	12	232	1	sym
M2	1	2	12	232	1	asym
M3	1	2	12	308	1	sym
M4	1	2	12	308	1	asym

表 4.3: Symmetry between Shots

		shot$i \to$ shot$j$			
		$u \to u$	$u \to d$	$d \to u$	$d \to d$
shot$j \to$ shot$i$	$u \to u$	$=$	UK	UK	$=$
	$u \to d$	UK	$=$	$=$	UK
	$d \to u$	UK	$=$	$=$	UK
	$d \to d$	$=$	UK	UK	$=$

表 4.4: Asymmetry between Shots

		shot$i \to$ shot$j$			
		$u \to u$	$u \to d$	$d \to u$	$d \to d$
shot$j \to$ shot$i$	$u \to u$	$=$	UK	UK	UK
	$u \to d$	UK	UK	$=$	UK
	$d \to u$	UK	$=$	UK	UK
	$d \to d$	UK	UK	UK	$=$

時間）を表し，下段は 4.6.5 節で述べたヒューリスティック解法で求まる初期解からの改善率を示す．この場合は最適解を求めているので，表現方法にかかわらず頂点集合モデルごとに等しくなっており，初期解に対する改善率は 0.25%〜4.66%となっている．なお，頂点集合モデル M4 についてはどの方法でも最適解は得られなかった．そのため，M4 に関しては最適解ではなく，丸 1 日計算した時点での実行可能解とその解が得られるまでの処理タスク数を示した．数値実験結果としては，頂点数が少ない M1 においては，露光頂点を 8 頂点で表現した方が有利であり，それ以外では，6 頂点および 4 頂点で表現した方が計算時間は短くなる傾向がみられた．このことから，単純に頂点数が少ないからといって計算時間が短くなるとは限らないことが分かる．初期解は人手で作成したものに相当することから，ウェーハ一枚当たりの処理時間を最大で 4.66% 早くできることが確認できた．大量生産を行うときに，この違いは大きいものと考えられる．作成された移動順である図 4.2 では，露光頂点一行すべてを訪問しないで，次の行に移る部分があるなど人手では作成困難な部分を持つものである．

表 4.6 に，Linkern を用いた準最適解の実験結果を示す．初期解に対する Linkern の改善率は 0.03% 〜2.95% に留まっているが，Linkern の実行時間は，全体的に非常に短く，頂点数が少ないほど高速である．ただし，頂点数は多いほど良い解が得られている．以上，MSOP

表 4.5: Optimal Solution of MSOP

model	init.sol.	comp.time(task) and sol.(sec.),ratio(%)		
	(sec.)	8 vertices	6 vertices	4 vertices
M1		691(174)	1028(192)	1008(194)
sol.	66.963	63.840	63.840	63.840
ratio		4.66	4.66	4.66
M2		5924(994)	6733(1049)	2506(539)
sol.	62.682	59.901	59.901	59.901
ratio		4.44	4.44	4.44
M3		6864(819)	5436(683)	5230(883)
sol.	72.444	71.731	71.731	71.731
ratio		0.98	0.98	0.98
M4		—- (2068)	—- (4452)	—- (8695)
sol.	67.325	67.159	66.348	66.28
ratio		0.25	0.25	0.25

の最適解，準最適解を求める数値実験について述べた．最適解を求めることについては，問題による計算時間の違いが大きいため，複数のモデルを同時に実行するなどにより，解を効率よく得ることが期待できる．ただし，頂点数が多い問題の最適解を得ることは今後の課題である．

4.8 MSOP のバリエーション

本節で述べた MSOP には幾つかのバリエーションがあり得る．このバリエーションは露光装置のチューニングやウェハに焼き付ける際に信頼性をあげるためなどの制約から発生する．特に露光頂点に関しては対称性が存在する場合も考えられ，その場合は頂点数を減らすことができる．対称性によって頂点数を減らすことが出来れば，同じ意味を持つエッジを削減できるので，TSP ソルバーの効率化に寄与すると考えられる．

表 4.2 節ではある露光頂点 i から別の露光頂点 j へ 4 種類の移動時間を考慮していた．MSOP の場合，表 4.4 のように i から j への移動時間と j から i への移動時間について 4 通りの組み合わせで等しい移動時間が最も二つの露光頂点間での移動時間に関する制約や緩い場合である．表 4.3 は移動時間に関する制約が厳しい場合であり，実用されている半導体露光装置では現実的な設定ではない．しかし，表 4.3 はひとつの露光頂点を 3 頂点あるいは 2 頂点で表現できる．表現方法を図 4.4 に示した．非対称の場合と異なり，(a),(b) 両方共，入力用の頂点と出力用の頂点を区別する必要がなく，共有できるので，頂点数を大幅に減らすことができる．

表 4.7 に 3 頂点モデルで表現した場合の Concorde による厳密解の実行時間，Linkern を用

表 4.6: Sub-Optimal Solution of MSOP

model	init.sol.	comp.time and sol. (sec.),ratio(%)		
	(sec.)	8 vertices	6 vertices	4 vertices
M1		0.749	0.493	0.341
sol.	66.963	65.029	65.039	65.189
ratio		2.53	2.15	2.05
M2		0.728	0.458	0.33
sol.	62.682	60.703	60.715	60.805
ratio		2.95	2.77	2.57
M3		1.117	0.709	0.484
sol.	72.444	72.184	72.155	72.293
ratio		0.16	0.35	0.03
M4		1.109	0.661	0.458
sol.	67.325	67.103	67.007	67.123
ratio		0.09	0.27	0.07

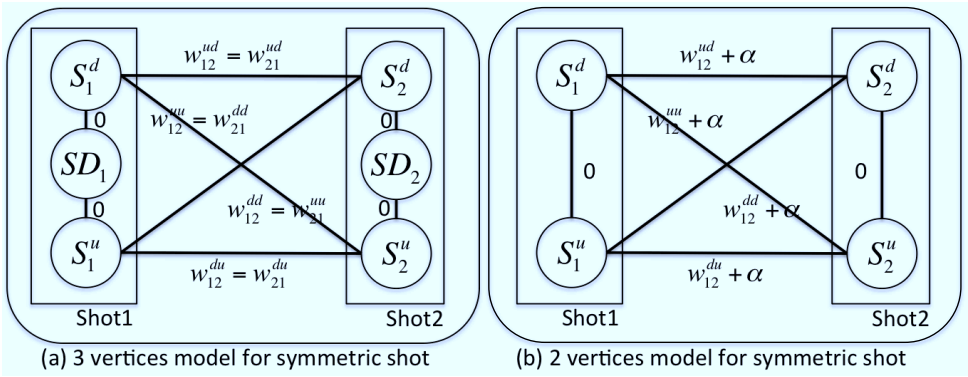


図 4.4: STSP model for MSOP

いた近似解の実行時間，4.3 節で述べた IP モデルによる解法を使った厳密解の実行時間を示した．IP モデルによるものは CPLEX の Callback 関数を使い，分枝時に部分巡回路を探し，部分巡回路を除去する制約条件を入れてある．問題は現実に即したパラメータを使い，アライメントグループは一つの場合で実行した．Linkern は近似解であるため，非常に高速に解を求めている．IP を使ったものは厳密解を得ているが，Concorde を使った場合より計算時間が非常にかかっている．これは Concorde が TSP を高速に解くためのカットを導入しているなど，単純に部分巡回路を除去するだけの IP よりも高性能であることが原因である．

図 4.5 に問題 2 において実際に作成されたウェハを示した．a) は専門家が最適だと考えて作成したもので，他のウェハに比べて規則的になっている．b) は Concorde によって最適解を求めたもので，Linkern は近似解であるが，近似解のほうが不規則的になっている．Concorde でも求めた MSOP のほうが Linkern のものより 0.2 秒程度速くなっているが，この程度の違

表 4.7: Comparing running-time

問題	アライメント数	露光頂点数	Concord	Linkern	IP
1	4	64	6 秒	0.1 秒	19 秒
2	12	232	34 秒	0.7 秒	29 分
3	48	232	34 秒	0.7 秒	281 分
4	12	136	46 秒	0.4 秒	17 分

いでも，ウェハの作成時間の短縮は有意であると考えられる．

図 4.6 には，アライメント頂点，露光頂点の位置が同じだが，異なる露光装置の仕様を与えた場合の経路を示す．ここで，露光装置の仕様は露光頂点間を移動するときの加速度などのパラメータで決定され，露光装置の性能を決める大きな要因となる．このようなパラメータは，ウェハを露光する時間や歩留まり率などを考慮して決定される．図 4.6 に示したように，パラメータが異なれば経路が変わってくる．このように MSOP の厳密解を求めるは，露光装置の最適なパラメータの決定に役立つと考えられる．

その他に，実際の MSOP の制約として用いられるのは，上端及び下端の露光頂点におけるスキャン方向である．大抵の場合，スキャンによる露光の信頼性の問題から，一方向だけしか使われないことが多い．このような一方向でしかスキャンされない露光頂点は 1 頂点で表現することが可能である．

4.9 第 4 章のまとめ

本章では，スキャナーにおけるウェーハ一枚当たりの露光処理時間を真に最短にするため，移動順最適化問題 (MSOP) を定義して STSP を利用した解法を研究した．本章が提案する手法により，MSOP の真の最適解および準最適解を得ることが可能となり，従来の方法に比べ，ウェーハ一枚あたりの処理時間を 0.03%〜4.66% 短縮できることを数値実験で確かめた．

MSOP は露光頂点においてスキャン方向を表現しなければならないため，一つの露光頂点をひとつの頂点とした TSP によって表現することはできないが，本章では一つの露光頂点を４，６，８頂点で表現する手法を提案した．これらの問題記述の違いは，MSOP の解を得るという意味では同値であるが，実行時間においてはどの方法が最適であるということはないが，最適解を得るまでにかなりばらつきがある．この点から，MSOP の解を得るためには，本章で提案した手法を同時に実行して，最も効率的だった解法から解を得るという方法が考えられる．

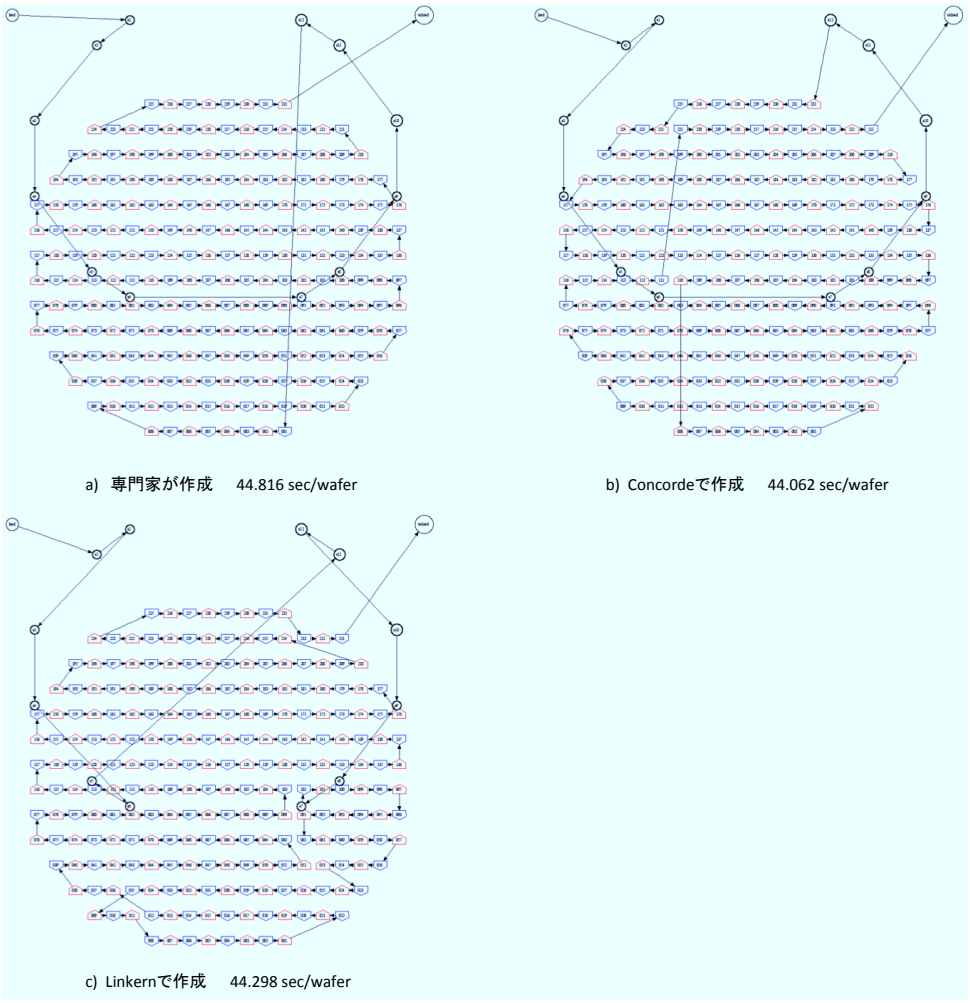


図 4.5: Comparing MSOP

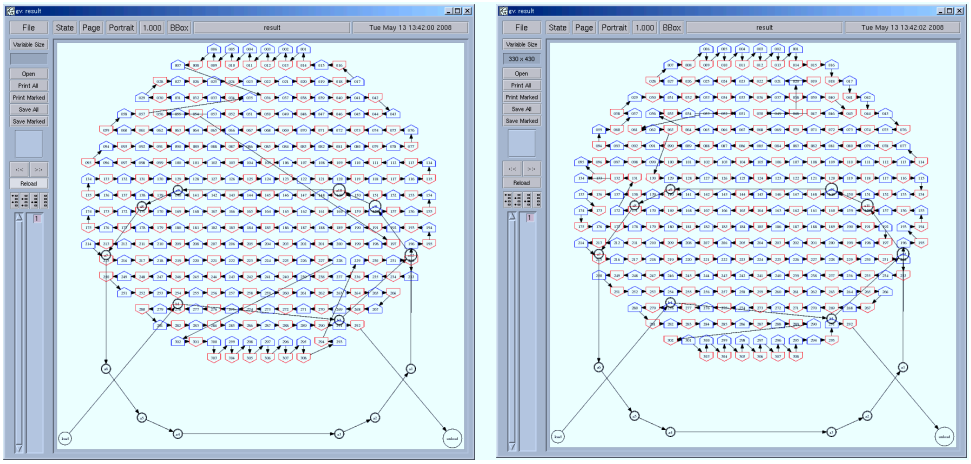


図 4.6: Same layout but Different Specification

第5章　ナーススケジューリング問題

5.1　はじめに

ナーススケジューリング問題 (Nurse Scheduling Problem: **NSP**) は病院における看護師の勤務シフトを決定する問題である (図 5.2 に勤務表例)．健全な看護師の勤務環境を保ちつつ，人命を守るという観点から質の高い勤務表を作成することは重要であり，問題の分析と解法について多くの研究が行われている [11]．近年の性能向上が著しい汎用ソルバーを用い，厳密解が求められていない NSP のベンチマーク問題に挑戦したので，本章ではその結果を報告する．

NSP は組合せ最適化問題の一つであり，なるべく多くの制約を満たすことを目的としている [76]．NSP は二つの側面より研究されている．一つ目は，NSP を特徴づける研究である [44, 94]．NSP で求められているものは実用的な勤務表の作成であり，聞き取り調査や過去の勤務表を参考に制約が決定される．勤務表は人間から見て満足するものでなければならないため，良い解を導く目的関数と制約の重要性についての議論が必要である [76]．二つ目は，NSP を解く研究である．NSP を解くことの難しさは，NSP が NP 困難問題の一つであることに由来する [11]．NSP を対象にした解法は，厳密解法 [5] や近似解法 [12, 43, 75, 94] の側面から非常に多くの研究が行われている．

本章は，二つ目の課題に関して，厳密解を求める研究である．NSP に関してはベンチマーク [10, 22] が公開されているので，これを解くことで解法を評価する．解法に関して，本章は他の多くの論文とは異なり，新たなアルゴリズムを提案するものではなく，既存のソルバーを使って NSP を解く．これにより，性能向上が著しいソルバーの NSP に対する有効性を評価する．

まず，NSP を混合整数線形計画問題 (Mixed-Integer Linear Programming: **MILP**) により記述する．MILP を使って解いた例としては 1997 年の論文 [91] が存在するが，最近では，”Just MIP it!”[27] という論文が出版されるほどに MILP ソルバーの性能向上は著しい．そのため，列生成法を使った解法 [5] のようにアルゴリズムの一部としてソルバーを使うのではなく，NSP を一つの MILP の問題として扱っても解ける可能性が高まっている．

次に，充足可能性判定問題 (Satisfiability Problem: **SAT**) により記述する．NSP に関しては，一般的な制約充足問題 (Constraint Satisfaction Problem: **CSP**) を使った研究が多く行われている [83]．SAT は CSP の一種であり，命題論理に特化した問題を扱う．毎年開かれる

SAT competition[87] を通して高速化に関する開発が進んでおり，OR の問題に適用する試みが行われている [55]．NSP に関しても SAT を使った研究は行われた [57]．この研究では，自前の SAT ソルバーに制約違反の最小化機能を実現しているが，本章ではブラックボックスとして SAT ソルバーを使うことで解を得る．そして，同じ厳密解法である MILP と比較する．本研究は，ソルバーを使って解いた時の NSP の困難さを理解する上で意味があると考える．

5.2　記号の説明

本章で扱う記号を説明する．N は看護師の集合，D はスケジュール期間中の日の集合であり，30 日間の場合は $D = \{1, 2, \ldots, 30\}$ となる．T は勤務シフトの集合を表す．H はスケジュール期間中における土曜日となる日の集合である．G は看護師グループへのインデックス集合を表し，インデックス $g \in G$ について，そのグループに属する看護師集合を N_g とする．看護師グループは，看護師の医療における専門分野，技術の習熟度，担当患者で決定される．$d \in D, n \in N, t \in T$ に対し $< d, n, t >$ は d 日における看護師 n の勤務シフトが t であることを示す．連続勤務パタンを $t_0 t_1 \ldots$ により参照する．例えば，夜勤の後に休暇である連続勤務パタンは〃夜休〃と表現され，$t_0 =$ 夜, $t_1 =$ 休 となる．d 日の勤務シフトが夜勤ならば $d+1$ 日の勤務シフトは休暇になる．

5.3　ベンチマークにおける NSP

本節では，ベンチマーク [22] で採用されている制約を示す．括弧内にはその制約によって決定される定数と集合を示す．これらは次節から説明する定式化で利用する．添字に関して，$d \in D, n \in N, t \in T, g \in G$ とする．

(a) 勤務シフトごとの最小・最大連続回数 (c_t^1, c_t^2)
(b) 勤務シフトごとの最小・最大間隔 (c_t^3, c_t^4)
(c) 禁止連続勤務パタン (P)
(d) 各日，グループ，勤務シフトの最小・最大人数 (c_{dgt}^5, c_{dgt}^6)
(e) 各看護師，勤務シフトの最小・最大回数 (c_{nt}^7, c_{nt}^8)
(f) 各看護師の土日休暇の最小・最大回数 (H, c_n^9, c_n^{10})
(g) 前月の勤務表 (各看護師の前月末勤務実績)
(h) 各看護師についての希望勤務 (L^+)
(i) 各看護師についての希望しない勤務 (L^-)

これらの中で，(d) はシフト拘束条件，(a)，(b),(c)，(e)，(f)，(h) ，(i) は看護師拘束条件と呼ばれている [43]．勤務シフトは日勤と夜勤からなる 2 シフト，日勤，準夜勤，深夜勤から

なる 3 シフトが代表的である．休暇およびその他 (例えば，セミナー) も勤務シフトとして扱われる．(d) ではこのグループを使って全ての日について必要な看護師の人数の下限 c^5_{dgt} と上限 c^6_{dgt} が規定される．(a) において勤務シフト t は c^1_t 日間以上 c^2_t 日間以下連続するという形式で連続日数の条件が記載される．(b) では，勤務シフト t は間を c^3_t 日以上 c^4_t 日以下空けるという条件が記載される．(c) では，夜勤の翌日の日勤（連続勤務パタンでの表現は” 夜日”）は禁止するなどの禁止される連続勤務パタン集合 P が規定される．スケジュール期間において，(e) は勤務シフト t の実施回数の下限 c^7_{nt} と上限 c^8_{nt}，(f) は土日連続休暇の取得回数の下限 c^9_n と上限 c^{10}_n を規定する．(g) の前月のスケジュールは月の上旬の勤務表を制約する条件となる．(h)，(i) は各看護師について特定の日について割り当てられる，あるいは割り当てられない勤務シフトを示しており，$< d, n, t > \in L^+, < d, n, t > \in L^-$ となる，L^+, L^- が規定される．

上記すべての条件を満たす勤務表が作成されることが望ましいが，NSP では作成可能であることは保証されない．ロバストに勤務表を作成するため，本章では，参考文献 [44] を参考にシフト拘束条件 (d) を緩和可能な制約とし，過不足を認める．これにより，本章では勤務表作成の目的関数を過不足人数の総和の最小化とした．本章では，現実性を考慮して，各日，各勤務シフト，各グループに割り当てられた人数について過不足 1 名まで認めることとした。

5.4 MILP と SAT による解法の比較

5.4.1 MILP による問題の定式化

MILP は組合せ最適化問題を扱う手法としては一般的であり，多くの研究事例がある．MILP においては，制約条件は線形 (不) 等式によって表現され，一部の変数は整数拘束される．5.1 式に 5.3 節で述べたベンチマークの MILP による表現を示す（この定式化は 0-1 整数計画問題となっているが，本章では MILP の意味を拡大解釈し，全ての変数が整数拘束または 0-1 拘束されるものも含むとして話を進める）．NSP の制約は複雑であり，直接問題を解く定式化においては制約の性質に応じて式の形を工夫することも効率に影響する．

変数 z_{dgt}(定義は (17) 式) は，d 日の看護師グループ g の勤務シフト t で過不足が発生する場合は 1 となる．目的関数 (1) 式は z_{dgt} の和を最小化する．この目的関数値のことを本章では**ペナルティ**と呼ぶ．

変数 x_{dnt} は，d 日の看護師 n の勤務シフトが t の場合は 1，そうでない場合は 0 となる 2 値整数変数（定義は (15) 式）である．d が 0 以下の場合は前月勤務表（制約 (g)）を参照し，値を決定する（例えば，0 日は前月の末日を表す）．

(2) 式は各看護師について各日の勤務で必ず一つの勤務シフトを決定することを表してい

る．制約 (a) に関して，(3) 式は勤務シフト t が c_t^1 日以上連続することを示す．d 日が勤務シフト t ではなく，$d-1$ 日が勤務シフト t のとき，$d-2$ 日から前に少なくとも c_t^1-1 日間勤務シフト t が連続することを表現しており，連続数の下限を守る．(4) 式は勤務シフト t が c_t^2 日を超えて連続しないことを示す．$d-(C_t^2+1),\ldots,d$ 日の全てが勤務シフト t ではないことで表現する．制約 (b) に関して，(5) 式は勤務シフト t の最小間隔が c_t^3 であることを示す．例えば，$c_t^3=3$ の場合は中2日および中1日で勤務シフト t を行うことはできない (連続での勤務は，最大連続勤務日数の条件を満たす限り認められる)．中2日での勤務シフト t の禁止は，$d-3$ 日と d 日が勤務シフト t ならば，$d-2$ 日あるいは $d-1$ 日に勤務シフト t を割りつけることで記述できる．また，中1日での勤務シフト t の禁止も，$d-2$ 日と d 日（ならびに $d-3$ 日と $d-1$ 日）が勤務シフト t ならば，$d-1$ 日（$d-2$ 日）に勤務シフト t を割りつけることで記述できる．(6) 式は最大間隔に関する制約であり，c_t^4 日間には必ず一回勤務シフト t の日があることを示し，c_t^4+1 日以上の間隔が空かない条件となる．制約 (c) に関し，(7) 式は，連続勤務禁止パタン $t_0\ldots t_k$ がどの連続した日をとっても出現しないようにする．制約 (d) に関し，(8) 式は，d 日におけるグループ g の勤務シフト t に対する最小人数 c_{dgt}^5 に関する制約および最大人数 c_{dgt}^6 に関する制約を表している．これらは，5.3 節で述べたようにロバストに勤務表を出力するため，一人までの過不足 z_{dgt} を認めることによって緩和している．

(9) 式は，スケジュール期間に関する勤務シフト t の勤務数を制約している（制約 (e)）．(10) 式〜(12) 式は土日に連続して休暇を取る回数を制約している（制約 (f)）．y_{dn}（d は土曜日）は，1 ならば土日に看護師 n が休暇を取り ((11) 式)，逆に土日連休を取るならば 1 となる ((12) 式)2 値整数変数である．y_{dn} は (16) 式で定義されている．制約 (h)，制約 (i) に関する (13) 式，(14) 式は，L^+ と L^- を使って特定の日の看護師の勤務シフトを制約する．

本章で扱うベンチマークは一ヶ月をスケジュール期間とし看護師の数も多いため，5.4.3 節で述べるように非常に多い制約条件を含む問題を解く必要がある．しかしながら，最近の MILP ソルバーの進歩はめざましく，変数・制約式の数は十分に取り扱えるものとなっていると考えている．

$$
\begin{aligned}
&\text{minimize (1)} && \sum_{d\in D}\sum_{g\in G}\sum_{t\in T} z_{dgt} \\
&\text{s.t.(2)} && \sum_{t\in T} x_{dnt} = 1, d \in D, n \in N \\
& && \sum_{i=2}^{c_t^1} x_{(d-i)nt} - (c_t^1 - 1)x_{(d-1)nt} \\
&(3) && \quad +(c_t^1 - 1)x_{dnt} \geq 0, d \in D, n \in N, t \in T \\
&(4) && \sum_{i=0}^{c_t^2} x_{(d-i)nt} \leq c_t^2, d \in D, n \in N, t \in T \\
& && x_{(d-c)nt} - \sum_{i=1}^{c-1} x_{(d-i)nt} + x_{dnt} \leq 1, \\
&(5) && \quad d \in D, n \in N, t \in T, c \in \{2, \ldots, c_t^3\} \\
&(6) && \sum_{i=0}^{c_t^4} x_{(d-i)nt} \geq 1, d \in D, n \in N, t \in T \\
&(7) && \sum_{i=0}^{k} x_{(d+i-k)nt_i} \leq k, t_0 \ldots t_k \in P, d \in D, n \in N \\
& && \sum_{n\in N_g} x_{dnt} + z_{dgt} \geq c_{dgt}^5, d \in D, g \in G, t \in T \\
&(8) && \sum_{n\in N_g} x_{dnt} - z_{dgt} \leq c_{dgt}^6, d \in D, g \in G, t \in T \\
& && \sum_{d\in D} x_{dnt} \geq c_{nt}^7, n \in N, t \in T \\
&(9) && \sum_{d\in D} x_{dnt} \leq c_{nt}^8, n \in N, t \in T \\
& && \sum_{d\in H} y_{dn} \geq c_n^9, n \in N \\
&(10) && \sum_{d\in H} y_{dn} \leq c_n^{10}, n \in N \\
&(11) && 2y_{dn} - x_{dn\,\text{休}} - x_{(d+1)n\,\text{休}} \leq 0, d \in H, n \in N \\
&(12) && y_{dn} - x_{dn\,\text{休}} - x_{(d+1)\,\text{休}} \geq -1, d \in H, n \in N \\
&(13) && x_{dnt} = 1, < d, n, t > \in L^+ \\
&(14) && x_{dnt} = 0, < d, n, t > \in L^- \\
&(15) && x_{dnt} \in \{0, 1\}, d \in D, n \in N, t \in T \\
&(16) && y_{dn} \in \{0, 1\}, d \in H, n \in N \\
&(17) && z_{dgt} \in \{0, 1\}, d \in D, g \in G, t \in T \\
& && (11),(12)\text{ で }t = \text{休 は休暇}
\end{aligned}
\tag{5.1}
$$

5.4.2 SAT による問題の定式化

$$
\begin{aligned}
&(1') \quad \sum_{d\in D}\sum_{g\in G}\sum_{t\in T} Z_{dgt} \le Penalty \\
&(2') \quad \bigvee_{t\in T} X_{dnt}, d\in D, n\in N \\
&\qquad \overline{X_{dnt_1}} \vee \overline{X_{dnt_2}}, \\
&\qquad\qquad d\in D, n\in N, t_1, t_2 \in T, t_1 \neq t_2 \\
&(3') \quad X_{(d-i)nt} \vee \overline{X_{(d-1)nt}} \vee X_{dnt}, \\
&\qquad\qquad d\in D, n\in N, t\in T, i\in\{2,\ldots,c_t^1\} \\
&(4') \quad \bigvee_{i=0}^{c_t^2} \overline{X_{(d-i)nt}}, d\in D, n\in N, t\in T \\
&(5') \quad \overline{X_{(d-i)nt}} \bigvee_{j=1}^{i-1} X_{(d-j)nt} \vee \overline{X_{dnt}}, d\in D, \\
&\qquad n\in N, t\in T, i\in\{2,\ldots,c_t^3\} \\
&(6') \quad \bigvee_{i=0}^{c_t^4} X_{(d-i)nt}, d\in D, n\in N, t\in T \\
&(7') \quad \bigvee_{i=0}^{k} \overline{X_{(d+i-k)nt_i}}, t_0\ldots t_k \in P, d\in D, n\in N \\
&(8') \quad c_{dgt}^5 - Z_{dgt} \le \sum_{n\in N_g} X_{dnt} \le c_{dgt}^6 + Z_{dgt}, \\
&\qquad d\in D, g\in G, t\in T \\
&(9') \quad c_{nt}^7 \le \sum_{d\in D} X_{dnt} \le c_{nt}^8, n\in N, t\in T \\
&(10') \quad c_n^9 \le \sum_{d\in H} Y_{dn} \le c_n^{10}, n\in N \\
&\qquad \overline{Y_{dn}} \vee X_{dn\,休}, d\in H, n\in N \\
&(11') \quad \overline{Y_{dn}} \vee X_{(d+1)n\,休}, d\in H, n\in N \\
&(12') \quad Y_{dn} \vee \overline{X_{dn\,休}} \vee \overline{X_{(d+1)n\,休}}, d\in H, n\in N \\
&(13') \quad X_{dnt}, < d, n, t > \in L^+ \\
&(14') \quad \overline{X_{dnt}}, < d, n, t > \in L^- \\
&\qquad (11'),(12') \text{ で } t = 休 \text{ は休暇}
\end{aligned}
\tag{5.2}
$$

$$
\begin{aligned}
&(18)\quad S_0^m \rightarrow S_0^{m-1} \wedge \overline{X_m}\\
&(19)\quad \overline{S_0^m} \rightarrow \overline{S_0^{m-1}} \vee X_m\\
&(20)\quad S_i^m \rightarrow (S_{i-1}^{m-1} \wedge X_m) \vee (S_i^{m-1} \wedge \overline{X_m}),\\
&\qquad i \in \{1, \ldots, m-1\}\\
&\qquad \overline{S_i^m} \rightarrow (\overline{S_{i-1}^{m-1}} \vee \overline{X_m}) \wedge (\overline{S_i^{m-1}} \vee X_m),\\
&(21)\qquad i \in \{1, \ldots, m-1\}\\
&(22)\quad S_m^m \rightarrow S_{m-1}^{m-1} \wedge X_m\\
&(23)\quad \overline{S_m^m} \rightarrow \overline{S_{m-1}^{m-1}} \vee \overline{X_m}\\
&(24)\quad \bigvee_{i=\max(c_{min}+m-M,0)}^{\min(c_{max},m)} S_i^m\\
&\qquad \overline{S_i^m}, i \in \{0, \ldots, \max(c_{min}+m-M,0)-1\}\\
&\qquad \overline{S_i^m}, i \in \{\min(c_{max},m)+1, \ldots, m\}\\
&\qquad \text{SAT における算術加算} \quad s_m \leftarrow s_{m-1} + X_m\\
&(24')\quad \bigvee_{i=\max(c_{dgt}^5+m-M,0)-1}^{\min(c_{dgt}^6,m)+1} S_i^m\\
&\qquad \overline{S_i^m}, i \in \{0, \ldots, \max(c_{dgt}^5+m-M,0)-2\}\\
&\qquad \overline{S_i^m}, i \in \{\min(c_{dgt}^6,m)+2, \ldots, m\}\\
&\qquad S^m_{\max(c_{dgt}^5+m-M,0)-1} \vee S^m_{\min(c_{dgt}^6,m)+1} \rightarrow Z_{dgt}\\
&\qquad \text{上下限値の制約違反の表現}
\end{aligned}
\tag{5.3}
\tag{5.4}
$$

SAT は命題論理に基づき，論理式が真か偽かを証明する．論理式が真となるような論理変数の真偽値が存在するなら充足可能と呼び，それ以外の場合を充足不可能と呼ぶ．5.4.1 節で述べた MILP による定式化において，2 値整数変数が論理的に用いられている制約は容易に SAT の論理式に変換可能である．これに対し，人数や勤務数の和に関する制約は直接 SAT に変換できない．従来研究 [57] では，和を求めるのではなく,「可能でない組合せを列挙する」方法を用いて表現した．これに対して，ペナルティに関する過不足数の扱いを容易にするため，本章では，真を 1，偽を 0 として，明示的に和を計算する．

図 5.2 に SAT による定式化を示す．図 5.1 に対応して式番号をつけ，論理和 ($\vee$)，論理積 ($\wedge$)，含意 ($\rightarrow$)，否定 ($\overline{X}$) を用いて論理式を示す．MILP で全ての制約式は満たされなければならないのと同様に，SAT では全ての論理式は真でなければならない．

MILP と異なり，SAT は最小化問題を直接解くことができない．そのため，NSP の SAT による解法では，ペナルティ (論理変数 Penalty) を $0, 1, \ldots$ と順に増やしていき，最初に充足可能な問題の解を出力する．2 値整数変数 x_{dnt}, y_{dn}, z_{dgt} に対し，真偽値を表す論理変数 X_{dnt}, Y_{dn}, Z_{dgt} を用いる．これを使うことで，例えば，(2) 式は (2') 式のように表現できる．(2') 式は，勤務シフトが少なくとも一つ選ばれる式と，任意の二つの勤務シフトの少なくと

表 5.1: NSP のベンチマーク（*SAT に関してはペナルティが 0 の時の数を示す）

	シフト	看護師	日数	勤務シフト	グループ	備考
2s	2	28	30	5	9	—
3s1	3	25	30	5	8	—
3s2						↑希望追加
3s3						↑希望追加
2sa	2	35	31	5	17	土日制約なし

	MILP 制約	MILP 変数	SAT 制約*	SAT 変数*
2s	16,318	6,472	607,157	172,852
3s1	14,463	5,770	515,166	149,485
3s2	14,490		515,193	
3s3	14,525		515,228	
2sa	26,231	9,641	1,123,946	322,214

もどちらかが偽である式から構成されている．上下限値を表現する (8') 式〜(10') 式および目的関数 (1') 式で算術和を求める．

SAT においては算術和を 2 進数全加算器によって表現することができるが，NSP のベンチマークにおいてその値がそれほど大きくないことから，各数字に対応する論理変数を導入する．これによって，上限値，下限値を考慮した算術和を容易に計算できる．今，m 個の論理変数の算術和を s_m とおくと，$s_m \in \{0, 1, \ldots, m\}$ となる．ここで，m 番目の論理変数 X_m が与えられたときに，$s_m \leftarrow s_{m-1} + X_m$ を計算しよう ($s_0 = 0$)．s_m が取り得る値に対応し論理変数 $S_0^m, S_1^m, \ldots, S_m^m$ を定義する．これらはちょうど一つだけ真となり，$s_m = i$ ならば S_i^m が真となる．この計算は 5.3 式のように与えられる．M を論理変数の数，$c_{\min}, c_{\max}$ を算術和の下限値，上限値とする．

(18) 式,(19) 式は 0 および 0 以外になる条件，(20) 式，(21) 式は i およびそれ以外になる条件，(22) 式，(23) 式は m および m 以外になる条件を表している．(24) 式は上下限値内に算術和を収めるための条件である．

(1') 式と (8') 式において使われる Z_{dgt} は，上下限値を一つ広げる，つまり，一人の過不足を表現する．これを表現するために (24) 式の代わりに式 5.4 の (24') 式を用いる．

M 個の論理変数の算術和では $O(M^2)$ 個の論理変数が必要であり，MILP に比べ，SAT では論理変数の数・論理式が非常に多くなる．実用的な規模の NSP に対して現在の SAT ソルバーが適用可能であるかを 5.4.3 節では検証する．

表 5.2: 3 シフト問題の暫定解発見の歴史

年 (発見者)	3s1	3s2	3s3
2003(池上)[43]	6	13	12
2009/9(Curtoise)[22]	5	9	11
2009/11(乾，前田，池上)[22]	2	4	3
本章 (SAT による厳密解)	2	3	3

5.4.3 数値実験

本節では ベンチマークを解き，計算時間を比較する．1 節で述べたように，最近の MILP ソルバーの性能向上は著しいため，困難なベンチマークが高速に解ける可能性がある．分枝限定法を使った MILP ソルバーでは，最小化問題の場合，下界値と暫定値が一致したとき最適解となる．下界値はそれより小さい値では実行可能解が存在しないことを表す．暫定値はそれまでに見つかった実行可能解の中での最小の目的関数値である．下界値と暫定値のギャップが 0 になり易いかどうかが実行時間に大きく影響する．一方，SAT は論理変数の真偽値だけで探索を行うため，下界値を定めることに関しては，整数変数が小数となる空間も探索してしまう MILP より効率的な可能性がある．

実験データとしては NSP のベンチマークである Ikegami_2shift(2s) および Ikegami_3shift(3s)[22, 43] および Ikegami_2shift_a(2sa) を用いる（表 5.1 参照）．これらは，日本の病院の調査から作成されたものであり，勤務実態がよく反映されている問題である．3s はさらに看護師の希望勤務の有無により 3 種類の問題に分かれる (3s1,3s2,3s3)．3s は 2009 年 10 月で最適解がわかっていなかった問題である．表 5.2 にこの問題に対するペナルティを示した．この表の結果が示すように，本章の手法により見つかった厳密解であるペナルティは近似解法から予想される比較的大きなペナルティの解とは大きく異なるものであった．2sa は 2 シフト問題で土日連続休暇に関する制約 (f) がない問題である．

実験では，MILP ソルバーとして CPLEX12.0 を使い，Intel QuadCore2.8GHz の PC によって実行する．SAT ソルバーとしては，CLASP1.3.0[17] を使い，Intel DualCore 2.4GHz の PC によって実行した．

2s および 2sa と 3s の最も大きな違いは，勤務シフトである．シフト数は両者ともに 5 種類であるが，2s が「休暇」,「日勤」,「夜勤 1」,「夜勤 2」,「その他」であるのに対し，3s は「休暇」,「日勤」,「準夜勤」,「深夜勤」,「その他」である．夜勤 2 は夜勤 1 の翌日に続く勤務であり，この二つで 2 交代制の一つの夜勤を表しているので，実質的に勤務シフトは 4 種類である．表 5.1 に示したベンチマークのデータに関して，変数の数，制約式の数がそのまま問題の難しさとなるわけではないが，SAT では算術和を求めるのに多くの論理変数・論理式を必要とする．そのため，MILP に比べて，これらの数が非常に多くなっている．

表 5.3: MILP による実行時間 (秒)(2時間まで実行)

	下限値		暫定値	
	ペナルティ	時間	ペナルティ	時間
2s	0	6	0	20
3s1	1	88	2	324
3s2	2	111	4	1109
3s3	2	127	3	661
2sa	19	6	19	20

表 5.4: SAT による実行時間（秒）（下線は充足可能）

ペナルティの上限	0	1	2	3	4
2s	41	45	16	27	18
3s1	6	14	32	39	65
3s2	4	7	16	77	62
3s3	3	8	16	100	42

表 5.3 に MILP に関して 2 時間以内に求められた最善の下界値と暫定値および実行時間を示す．下界値と暫定値が等しいときは最適解である．MILP は 2s および 2sa を解くのには有効であったが，3s を解くのには非力であった．次に述べる SAT の結果からわかる最適解から考えて，MILP による結果は 3s2 を除いて最適なペナルティを見つけているが，下界値はこれと等しくならなかった．

ペナルティは 0 未満にならないので，2s については暫定値 0 の解を見つけた時点で MILP の実行は終了する．これに対して，3s は暫定値以下の実行可能解がないことを証明する必要があり，これに計算時間の大半をかけている．2sa は最適解のペナルティが大きい問題であるが，最適解に対する下界値が短時間で求まっている．3s で土日休暇の制約を除いても実行時間に違いが見られないことから，MILP では実質的な勤務シフト数が実行時間に大きな影響を与えていると考えられる．

表 5.4 に SAT の計算時間を示す．SAT は 2s と 2sa で MILP よりも実行時間が長いものの，MILP では最適解が見つからない 3s の全ての問題に対して，最適解を出力することができた．図 5.1 に 2sa の SAT に対する結果を示す．2sa はペナルティが大きい問題であるため，SAT の適用回数が多い．

このように，SAT は MILP が解くことができなかった 3s を短時間で解くことができるが，MILP が短時間で解くことができた 2sa に対しては実行時間がかかることがわかった．2sa に対しても小さいペナルティの問題が充足不可能であることを短時間で判定できることから，SAT の適用ではペナルティの大きさが時間に大きな影響を与えると思われる．

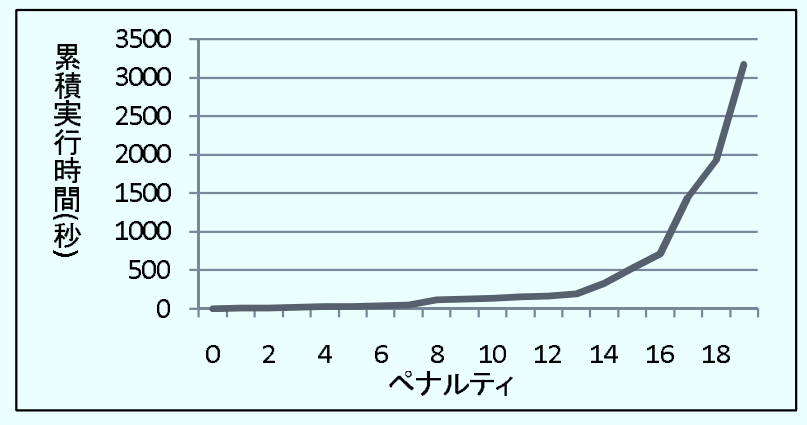


図 5.1: SAT による 2sa の実行時間

最後に 3s1 について SAT が生成した勤務表を図 5.2 に示した[1]．勤務表を人手で作成することは多大な労力を要する作業であることが理解されるだろう．本章は入手可能な汎用ソルバーを用いて現実的な時間内に NSP の厳密解が求められることを示したと考えている．

5.4.4 まとめ

本章では NSP のベンチマークを対象に，MILP および SAT による定式化および解法の評価を行った．その結果， MILP では解きにくい 3 シフト問題を SAT によって短い時間で解くことでできた．ただし，問題によっては MILP の方が優れている場合もあるため，現状では問題によって使い分けて利用することが最良である．これまで，近似解を求める研究は多くなされ，実用的に使われてきたが，本章の結果は，厳密解の利用が現実的になったことを示している．

5.5 SAT による解法の改良

本節では，SAT による NSP の表現方法を変えることによって，計算時間の短縮を試みる．2010 年にナーススケジューリングコンペティション (NRC: Nurse roestering competition)[77] が開催された．これは決められた時間内で最も良い解を探すことを目標としていた．筆者らは，SAT によって近似解を求めるアプローチを開発して参加したが，実行時間については満足する結果が得られなかった．

[1]看護師 1,2,3,4,5,6 がグループである g4 の 12 日，および看護師 14,15,16,17,18,25 がグループである g6 の 13 日に準夜勤の人数が 0 人になっている (必要人数は一人あるいは二人)．勤務表を各行で見た場合，各ナースのスケジュールに関する条件は全て満たしている．

日付

看護師	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	/	-	-	n	n	/	/	-	e	/	-	e	n	n	/	/	-	-	e	e	/	-	-	-	-	/	/	-	-	-
2	n	/	e	e	/	-	-	e	/	e	e	/	+	-	/	-	e	/	-	-	n	n	/	/	-	-	-	-	/	-
3	-	/	-	-	/	/	-	n	n	/	/	-	-	/	-	e	n	n	/	/	-	-	e	e	n	n	/	/	-	e
4	e	e	/	-	-	-	e	/	-	-	/	/	-	-	n	n	/	/	-	+	-	e	n	n	/	/	-	-	e	/
5	-	n	n	/	e	e	/	-	-	n	n	/	/	/	-	-	-	e	/	-	e	/	/	/	/	-	e	e	n	n
6	/	-	-	/	-	n	n	/	/	-	-	-	e	e	e	/	/	-	n	n	/	/	-	-	e	e	n	n	/	/
7	/	-	-	-	n	n	/	e	e	/	-	-	-	e	e	/	-	-	e	e	/	-	-	/	n	n	/	/	/	-
8	e	e	n	n	/	/	-	-	-	+	e	/	-	-	-	-	/	-	-	-	e	/	-	-	/	/	n	n	/	/
9	/	/	-	-	/	-	-	/	-	-	-	/	/	/	-	-	e	e	/	-	n	n	/	e	e	/	-	-	-	-
10	+	/	/	/	-	-	e	/	n	n	/	/	-	-	/	+	-	-	-	-	/	/	-	-	-	e	e	/	e	e
11	-	-	e	e	/	-	-	-	/	e	n	n	/	/	-	+	-	/	/	/	/	-	-	/	-	-	/	e	n	n
12	-	-	-	-	e	e	/	/	/	-	-	n	n	/	/	+	-	-	-	/	-	-	-	/	/	-	-	-	/	/
13	-	-	/	-	-	-	+	-	/	/	/	e	e	/	-	e	n	n	/	/	-	e	e	/	/	/	-	-	-	-
14	-	/	-	-	e	e	e	/	-	+	/	/	/	-	e	n	n	/	/	-	-	e	e	n	n	/	/	-	-	-
15	-	-	e	/	-	-	/	-	n	n	/	e	e	/	-	e	/	/	-	e	e	n	n	/	/	-	-	/	-	n
16	e	e	/	-	n	n	/	e	e	/	-	-	+	n	n	/	/	-	e	/	+	-	/	-	-	e	/	n	n	/
17	/	-	n	n	/	/	-	-	/	e	e	/	+	-	-	/	-	n	n	/	/	-	-	e	e	/	e	e	/	-
18	/	/	-	e	/	-	n	n	/	/	/	-	-	e	/	-	e	e	/	+	-	/	-	-	-	n	n	/	e	e
19	-	-	-	/	/	-	+	/	-	-	-	/	-	-	-	+	-	e	/	-	e	/	/	/	/	/	-	-	-	-
20	n	n	/	/	/	-	-	e	/	/	-	-	-	-	e	e	/	/	/	-	-	-	-	/	-	e	/	-	-	+
21	/	-	e	/	-	e	e	/	e	e	/	-	-	/	n	n	/	/	-	-	-	e	n	n	/	/	/	-	-	+
22	+	e	/	-	-	/	n	n	/	/	-	-	e	e	/	-	-	-	/	-	-	/	-	-	-	-	e	/	/	+
23	-	-	-	e	e	/	-	-	/	/	-	e	n	n	/	/	/	-	e	e	/	-	-	-	e	/	/	/	-	+
24	e	/	/	-	-	/	-	-	-	-	e	/	-	-	-	-	e	/	n	n	/	/	-	-	/	/	-	e	/	-
25	n	n	/	/	-	-	-	/	/	-	n	n	/	/	/	-	-	-	-	n	n	/	e	e	/	-	-	-	e	e
割当合計 休暇	7	7	8	8	9	7	6	9	11	9	8	10	5	9	8	6	8	8	10	6	8	9	5	9	10	11	9	7	8	5
日勤	9	11	10	10	9	11	10	9	7	7	10	8	10	9	10	8	10	10	8	10	9	9	13	9	8	7	9	11	10	9
準夜勤	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
深夜勤	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
その他	2	0	0	0	0	0	2	0	0	2	0	0	3	0	0	4	0	0	0	2	1	0	0	0	0	0	0	0	0	4

看護師	グループ(所属に1) g1	g2	g3	g4	g5	g6	g7	g8	勤務数 休暇	日勤	準夜勤	深夜勤	その他
1	1	1		1				1	9	13	4	4	0
2	1	1		1					9	11	6	3	1
3	1	1		1					10	10	4	6	0
4	1	1		1					9	11	5	4	1
5	1	1		1					10	8	6	6	0
6	1	1		1					10	9	5	6	0
7	1	1							9	11	6	4	0
8	1	1							9	12	4	4	1
9	1	1						1	10	14	4	2	0
10	1	1							10	11	5	2	2
11	1	1							10	11	4	4	1
12	1	1							10	15	2	2	1
13	1	1							10	12	5	2	1
14	1		1		1	1			9	10	6	4	1
15	1		1		1	1			9	10	6	5	0
16	1		1		1	1			9	7	6	6	2
17	1		1		1	1			10	9	6	4	1
18	1		1		1	1			10	9	6	4	1
19	1		1				1		10	16	2	0	2
20	1		1				1		10	13	4	2	1
21	1		1				1		10	9	6	4	1
22	1		1				1		9	13	4	2	2
23	1		1				1		10	11	6	2	1
24	1		1				1		10	14	4	2	0
25	1		1			1	1		9	11	4	6	0

図 5.2: ベンチマーク 3s1 で生成された勤務表 (/:休暇, -:日勤，　e:準夜勤, n:深夜勤,+:その他)

実行時間がかかった原因の一つには，前節のベンチマーク問題とは種類の違う問題であったことがあげられる．ベンチマーク問題では看護師拘束条件を必ず守らなければならない条件とし，シフト拘束条件を違反数を最小化する条件としていた．どちらを重視するかというのは実際問題として重要な問題である．看護師拘束条件を重視すれば，看護師たちの勤務条件の向上を優先することになり，非常勤の看護師などを人員補充に当てることになる．これに対して，シフト拘束条件を重視すれば，現状の人員の中でうまくやりくりをすることになる．この意味からは，長期的な人事を考える上では看護師拘束条件を重視すべきであり，短期的な病院の運用においてはシフト拘束条件を重視するなどの勤務表作成問題の利用方法が考えられる．これ以上，本章ではこの問題は考えないが，実際の実現方法としては，前者の表現方法では，各日・各シフト・各グループでの過不足数をカウントすればよかったが，後者の表現方法では，すべての看護師拘束条件の違反数を数える必要が出てくる．これは数えるべき要素数において非常な違いがあり，看護師拘束条件を重視する場合に比べて，シフト拘束条件を重視する方が非常に多くの要素を数える必要が出てくる．そのため，本節では数字の加算を表現する方法について更に考察する．

5.5.1　和の表現方法

NSP やその他，何らかの数を数える必要のある問題を SAT で表現する場合は和の表現を考える必要がある．MILP と異なり，SAT は論理式を扱うので，表現方法の中に和がないので，工夫が必要である．ここで，$I_i \in \{0,1\}, i = 1,2,\ldots,n$ となる論理変数において，真となるものの数を数えることを考える．数を数える場合は，次の 3 通りがあり得る．

1. ある数 m である
2. ある数 m 以上である
3. ある数 m 以下である

SAT で最も直接的にこれらの数を表現する方法は，そうならない論理変数の真偽値をすべて列挙して，否定することである．上記の 1 について言えば，m 未満と m を超える場合のすべての真偽値の組み合わせを列挙する．例えば，$n = 5$，$m = 2$ である場合，論理変数の組み $< I_1, I_2, I_3, I_4, I_5 >$ に対して，$< F, F, F, F, F >$，$< F, F, F, F, T >$，$< F, F, F, T, F >$，$< F, F, T, F, F >$，$\ldots$，$< T, T, T, T, F >$,$< T, T, T, T, T >$ のように T が二つにならない組み合わせを全て列挙する．この中で，例えば，$< F, F, T, F, F >$ となることは論理式 $\neg I_1 \wedge \neg I_2 \wedge I_3 \wedge \neg I_4 \wedge \neg I_5$ と表現できるため，その否定 $I_1 \vee I_2 \vee \neg I_3 \vee I_4 \vee I_5$ がこのような値の組みにならないことの表現となる．すべての真となる論理変数の数が m にならない場合を列挙し論理式で表現し，連言で接続すれば，1 の場合の和が表現できたことになる．同様に，2 については，m 個未満が真である場合を列挙すればよく，3 については，m 個を超える場合が真である場合を列挙すればよい．しかしながら，この表現の場合，連言接続される論理式の数が 1 の場合 $2^n - \binom{n}{m}$ となり，2 については $2^n - \sum_{k=0}^{m-1} \binom{n}{k}$ 個，3 については $2^n - \sum_{k=m+1}^{n} \binom{n}{k}$ の論理変数が必要となる．すなわち，論理変数の数が指数オーダーとなるため，実質的にこの表現は小さい問題でしか用いることができない．そのため，論理式の数が多項式オーダーで扱える和の表現方法を用いる必要がある．

前節で述べた数字の加算は，図で示すと図 5.3(a) による方法である．ここでは，論理変数で真の場合を 1，偽の場合を 0 として，$S = \sum_{i=1}^{n} I_i$ の計算を考える．$S \in \{0, 1, \ldots, n\}$ であるから，各値に対応して，論理変数 $< O_0, O_1, \ldots, O_n >$ を用意する．図は和が求められる過程を示している．この図で，表のセルは計算途中で必要となる論理変数を表しており，例えば，I_1 については 0 か 1 かを表す変数，I_2 までについては，和が 0,1,2 のいずれかであることを示す変数が必要であることを示している．和がある範囲の中であるならば，そうならない部分の計算については明示的に書く必要はないため，範囲が狭い場合，この和を実行するために必要となる論理変数と式は必要ない．しかし，一般的に全ての和の可能性を計算した場合，論理変数は $\frac{n(n+3)}{2}$ 個の論理変数が必要であり，$O(\frac{n(n+3)}{2})$ の長さの条件式が必要となる．

和を求める方法としてこのような論理回路の論理和の形以外の表現方法もあり得る．ここでは，n 個の変数 $I_1, I_2, \ldots, I_n$ から真となる変数の数を m 個にすることを考える．この場合を図示したものを図 5.3(b) に示す．$I_1, I_2, \ldots, I_n$ から真となる変数を m 個選ぶ場合には，m 個のノード $O_1, O_2, \ldots, O_m$ を用意しておき，O_i から I_j に一対一の写像が成り立てば良い．このとき，O_i に I_j が対応していることを表す論理変数を F_{ij} によって表現する．F_{ij} が真ならば，O_i は I_j に対応しているとともに，論理変数 I_j は真となる．すべての i について F_{ij} が偽ならば，I_j は偽とする．このような関係は，次のように書くことができる．

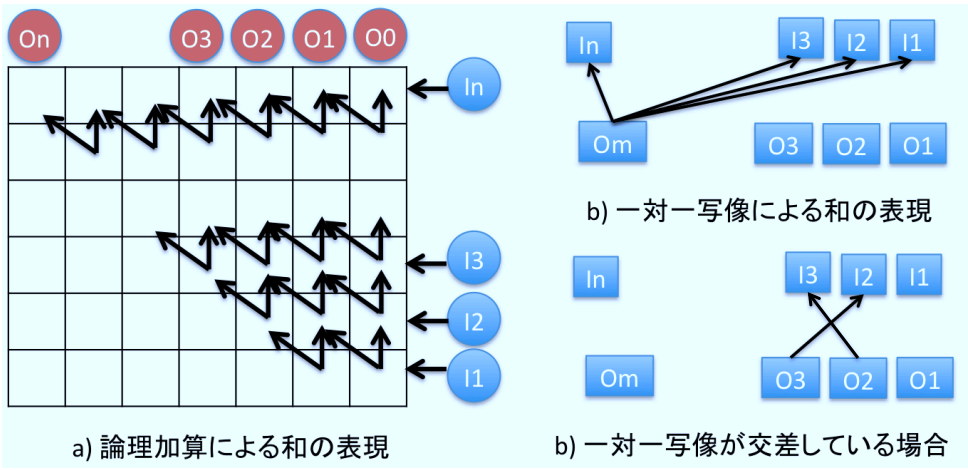


図 5.3: 論理式による和の表現方法

$$
\begin{aligned}
&F_{i,1} \vee \cdots \vee F_{i,n}, i \in \{1, \ldots, m\} \\
&\neg F_{i,j_1} \vee \neg F_{i,j_2}, 1 \in \{1, \ldots, m\}, 1 \leq j_1 < j_2 \leq n \\
&\neg F_{i,j} \vee I_j, i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\} \\
&F_{1,j} \vee \ldots F_{m,j} \vee \neg I_j, j \in \{1, \ldots, n\} \\
&\neg F_{i_1,j} \vee \neg F_{i_2,j}, j \in \{1, \ldots, n\}, 1 \leq i_1 < i_2 \leq m \\
&\neg F_{i_1,j} \vee F_{i_2,j+1} \vee \cdots \vee F_{i_2,n}, 1 \leq i_1 < i_2 \leq m, j \in \{1, \ldots, n-1\}
\end{aligned}
\tag{5.5}
$$

第 1 式は，各 O_i について少なくとも真となる I_j が存在することを表現している．．第 2 式は，各 O_i について高々真となる I_j が一つであることを表現している．第 3 式は，$F_{i,j}$ が真ならば I_j が真であること表現している．第 4 式は，I_j が真ならば，$F_{i,j}$ が真となる O_i が存在することを表現している．第 5 式では，各 I_j について，真となる $F_{i,j}$ が高々一つであることを示している．第 6 式は，後で述べる対称性除去のための条件である．

この場合，mn 個の論理変数が必要となり，$O(mn^2)$ の長さの条件式が必要となる．これは前に述べた論理回路の論理和に比べて論理変数が多いが，二つの値の論理和を階層的に繰り返して行う論理回路の論理和表現に比べて，よい直接的に和を表現したものになっているため，よりよい効率性が期待できる．そのかわり，論理和による表現方法では論理変数の I_j の値の組み合わせに対して，O_j の値の組み合わせは一意に決定されるが，一対一写像の場合は F_{ij} の値の組み合わせは複数あり得る．すなわり，I_j が真の場合は，$F_{1,j}, \ldots, F_{m,j}$ のどれかが真であればよいので，可能な組み合わせは $m!$ となってしまう．このような問題は対称性の問題と呼ばれ，このような対称性のない条件を書くことが，解を効率的に求める鍵の一つとなる．本節では，対称性のあるような論理変数の値の組み合わせを除去する条件を書くことによって，効率的に解を求めることを考える．第 6 式は，対称性を除去するための条件である．この式では，O_{i_1} が I_j に対応付けられているならば，O_{i_2} は I_{j+1} から I_n に対応付

けられなければならないことを表現している．これによって，F_{ij} と $F_{i'j'}$ について，$i < i'$ で $j > j'$ や $i > i'$ で $j < j'$ の場合など，図 5.3(c) のような一対一写像が交差することを避けることができ，ある O_i と I_j の対応付けで複数の割り当てが存在することを防ぐことができる．

以上は，n 個の論理変数のうちで m 個が真となる場合の条件であるが，同様に m 個以上や m 個以下の変数が真となる場合も記述できる．m 個以上を真とする場合，少なくとも m 個を真とすればよいので，次のように書くことができる．これは，前の式で，一対一写像が割り当てられなかった変数も真となれるようにすることで実現でき，具体的には，次のように前の式での第 4 式を省けば良い．

$$
\begin{aligned}
&F_{i,1} \vee \cdots \vee F_{i,n}, i \in \{1, \ldots, m\} \\
&\neg F_{i,j_1} \vee \neg F_{i,j_2}, \text{1} \in \{1, \ldots, m\}, 1 \leq j_1 < j_2 \leq n \\
&\neg F_{i,j} \vee I_j, i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\} \\
&\neg F_{i_1,j} \vee \neg F_{i_2,j}, j \in \{1, \ldots, n\}, 1 \leq i_1 < i_2 \leq m \\
&\neg F_{i_1,j} \vee F_{i_2,j+1} \vee \cdots \vee F_{i_2,n}, 1 \leq i_1 < i_2 \leq m, j \in \{1, \ldots, n-1\}
\end{aligned}
\tag{5.6}
$$

同様に，m 個以下を真とする場合，少なくとも (n-m) 個を偽とすればよいので，次のように書くことができる．この場合，一対一写像で割り当てられた変数を偽にしている．

$$
\begin{aligned}
&F_{i,1} \vee \cdots \vee F_{i,n}, i \in \{1, \ldots, n-m\} \\
&\neg F_{i,j_1} \vee \neg F_{i,j_2}, \text{1} \in \{1, \ldots, n-m\}, 1 \leq j_1 < j_2 \leq n \\
&\neg F_{i,j} \vee \neg I_j, i \in \{1, \ldots, n-m\}, j \in \{1, \ldots, n\} \\
&\neg F_{i_1,j} \vee \neg F_{i_2,j}, j \in \{1, \ldots, n\}, 1 \leq i_1 < i_2 \leq n-m \\
&\neg F_{i_1,j} \vee F_{i_2,j+1} \vee \cdots \vee F_{i_2,n}, 1 \leq i_1 < i_2 \leq n-m, j \in \{1, \ldots, n-1\}
\end{aligned}
\tag{5.7}
$$

本節で述べた和の表現に関して，変数の数は論理和による方法よりも多くなってしまうが，より階層的な方法によらず，直接的な表現となっている．これは，SAT ソルバーで解くときに少ない変数に対する値の割り当てで解が求まることにつながるので，効率化のために有利な特徴となっている．数値実験によってこれを評価する．

5.5.2 最適解の探索方法

前節で述べた手法によって NSP の厳密解を求める．本節では，ウィンドウ探索によるアルゴリズムについて述べる．ウィンドウサーチとは，ある一定の範囲内に解が存在するかどうかを繰り返し調べていく手法である．本節で述べる手法は，ある程度広い範囲で最適解の存在する領域を求め，最適解の証明を行う．このため，最初の最適解の存在範囲を求めるためにウィンドウサーチを求める．次に，最適解を探索するアルゴリズムを示す．

アルゴリズム: NSP の最適解を求める
入力: NSP の SAT による記述 (NSP)，ウィンドウサイズ (w>0)
出力: NSP の解
lb=0; ub=lb+w-1
while True:
　　solution=execSAT(NSP,lb,ub)
　　if solution が実行可能解: break
　　lb=ub+1; ub=lb+w-1
ub=(solution のペナルティ)-1
while lb<=ub:
　　solution1=execSAT(NSP,lb,ub)
　　if solution1 が実行可能解ではない: break
　　solution=solution1
　　ub=(solution のペナルティ)-1
solution を出力

上記のアルゴリズムは，二つの部分からなる．w はウィンドウのサイズであり，最初の while ループはウィンドウサーチを行い，最適解が存在する範囲を下限値 lb，上限値 ub によって求める．ここで，関数 execSAT(NSP,lb,ub) は SAT で記述された問題 NSP をペナルティが lb を下限値，ub を上限値とするウィンドウの中に入っているか調べる．この結果として，execSAT は解（論理変数に対する値の割り当て）を返す．割り当てられなかった場合は，実行可能解がなかった旨を返す．このため，execSAT 関数は NSP にペナルティの下限値，上限値の条件を付け加え，SAT ソルバーを呼び出すことによってこれを解く．２番目の while ループは，上限値をひとつづつ下げていき，実行可能解が存在しない領域を探す．これによって，厳密解を探索する．

5.5.3　数値実験

本節では，ナーススケジューリングコンペティション ([77]) のベンチマーク問題を使って論理和を使った和の表現と一対一表現を使った和の表現の計算時間を比較する．

ベンチマーク問題は，計算時間によって三つのカテゴリに分かれている．本実験では，比較的計算時間の短い二つのカテゴリ (sprint_ late, medium_ late) から５問づつを使って評価する．元々の問題は，シフト拘束条件は必ず守る条件となっており，看護師拘束条件は違反が認められている．ここの看護師拘束条件には重みが付けられており，違反した看護師拘束条件の重みの総和の最小化が目的関数となっている．しかしながら，これらのベンチマーク問題は元々近似解法のために作られており，厳密解を求めるのは難しいと考えられる．本節

での数値実験では，ベンチマーク問題の設定を使うが，目的関数を解を求められやすいものに変える．すなわち，看護師ごとに違反した看護師拘束条件の重みの総和を計算し，その最大値を最小化する．これは，看護師ごとに重みの総和の上限値を設定し，その上限値がもっとも小さい実行可能解が厳密解となる．

実験は Python2.5.2 によってプログラムを実装し，SAT ソルバーとして，plingeling[64] を使った．この SAT ソルバーは SAT-Race 2010[88] の並列トラックで優勝したソルバーである．

表 5.5 に実行結果を示した．S で始まる問題は sprint_ late の問題であり，M で始まる問題は medium_ late の問題である．最大ペナルティは目的関数値である．変数の数は，NSP を記述するの必要となった論理変数の数であり，条件数は論理式の数（連言接続される論理式の数）を示している．sprint_ late の方が問題としては小規模なのにかかわらず，最大ペナルティは大きな値となっている．前処理時間は，NSP の問題を作るのに必要となった時間であり，SAT 実行時間は SAT ソルバーの実行時間である．この実験結果では，ウィンドウサイズ w=5 としている．また，SAT ソルバーの計算時間は 1 時間以内とし，時間内に解が得られないことを”*” によって示した．

5.5.1 で述べたように，一対一表現は論理表現に比べて，変数の数，条件の数ともに多くなる傾向にある．そのため，表 5.5 にあるように，前処理である NSP の問題を生成する時間は多くなる．しかし，この時間は C 言語のような実行速度が早い言語を用いることで，大きな問題にはならないと考えられる．

SAT の実行時間に関しては，全体として，一対一表現は論理表現の実行時間を 70% 短縮している．この理由としては，和を表現する方法としては一対一表現は和を直接的に表現していることによるが，対称性除去も大きく貢献している．

5.5.4 まとめ

本節では，SAT における和の表現方法に関して，二つの表現方法を比べた．和の表現は様々な問題に現れると考えられるため，SAT を用いた解法ではその表現方法が重要となる．ナーススケジューリング問題において二つの表現方法で和を表現したとこと，論理和による表現よりも一対一による表現のほうが SAT の実行時間が短い結果が観測された．しかしながら，一対一による表現は和が大きくなるときに条件数が多くなるため，大きな数の表現はできないと考えられる．

5.6 第 5 章のまとめ

本章では，ナーススケジューリング問題の厳密解に対する効率的な解法について研究した．まず，MILP と SAT による解法を提案して，実験的に効率を比較した．MILP による NSP の

表 5.5: 論理和表現と一対一表現の実行時間の比較

問題		論理和表現			
	最大ペナルティ	変数数	条件数	前処理時間 (秒)	SAT 実行時間 (秒)
S01	3	21527	151119	12.9	44.3
S02	14	* 41965	466107	14.7	18.1
S03	4	24882	177947	10.4	61.5
S04	4	24982	178547	10.3	51.9
S05	3	21578	151260	12.0	69.1
M01	1	59369	434100	*	*
M02	1	49942	377532	71.6	1278.3
M03	1	39534	338666	40.2	229.6
M04	1	49656	374746	72.7	985.3
M05	1	60429	435796	*	*
問題		一対一表現			
	最大ペナルティ	変数数	条件数	前処理時間 (秒)	SAT 実行時間 (秒)
S01	3	21747	131097	12.8	15.4
S02	14	63571	2732669	97.7	40.8
S03	4	25022	147768	12.2	12.3
S04	4	25122	148148	12.0	11.7
S05	3	21798	131118	12.7	18.2
M01	1	51585	658620	813.2	634.4
M02	1	51482	675378	849.1	1051.7
M03	1	46142	656784	768.0	180.4
M04	1	51465	657568	817.8	315.4
M05	1	51881	693960	868.1	610.1

記述は，比較的直接的に行える．これは，MILP の記述性の高さによるが，SAT による記述では和の表現が大きな障害となる．そこで，非常に多くの論理変数を必要とするのだが，論理表現によって和を表現した．実験結果としては，MILP はシフト数の少ない問題については有効であるが，これは，MILP を解く際に実行される LP の解が小数になりにくいことが原因である．SAT の場合は，シフト数が多くても，効率的に問題を解けることがわかった．特に，ペナルティの数が少なく，シフト数が多い場合は，MILP では最適解の証明に困難があったが，SAT の場合は問題を短時間で解くことができた．

しかし，SAT の場合は，やはり，和の表現に問題があることがわかった．そこで，論理表現に対して，一対一表現を導入した．この結果，実行時間を短縮できることがわかった．これは，一対一表現が和を直接的に表現していることに理由があると考えられる．

ナーススケジューリング問題は，現実的に重要な問題である．勤務表は一度求められても，様々な理由により，何度も修正されることが予想される．このようなときに，近似解である勤務表よりも厳密解である勤務表のほうが説得力があるため，本章で述べたような厳密解法は意味がある．しかしながら，インタラクティブに勤務表を作成する場合は，より短時間でコンピュータは人間に解を提示する必要があると思われるため，さらなる効率化，例えば，並列処理の導入，などが実用的に必要となると考えられる．

第6章　おわりに

本論文では，三つの組合せ最適化問題について，厳密解を効率良く求める手法を研究した．本章では，各問題における結論をまとめると共に，組合せ最適化問題の厳密解法について考察する．

6.1　個々の問題についてのまとめ

組合せ最適化問題の厳密解は，一般的に対象の大きさが大きくなると急激に解くことが難しくなる問題であるが，個々の問題については次のような結論を得た．

1. 最小無矛盾 DFA 問題

   3 章では最小無矛盾 DFA 問題の厳密解法について研究した．厳密解法では MILP と SAT を用いた．MILP による方法は，最小化問題における下界値が上昇しない問題があった．これは，変数の整数条件を緩和した LP 緩和問題の目的関数が厳密解の状態数に比べて小さいことが原因である．つまり，整数変数が整数にならないことを示しており，そのようなケースは非常に多いことを意味している．この問題に対して，分枝カット法を使った実験を行った．その結果，下界値を上げる効果が観測されたが，解を得るまでには至らなかった．

   MILP と比べて，SAT による解法は解ける問題の規模が大きいが，これには対称性除去が大きく貢献している．特にクリークを用いた対称性除去は効果的である．これに対して，隣接グラフ密度が高い問題に関して，MILP によって最大クリークが短時間で得られることを示した．ヒューリスティックにより得られたクリークと最大クリークのサイズの差は大きいとは言えないが，最大クリークを用いて，対称性除去を行うことで，ベンチマーク問題を非常に効率的に解くことができた．

   隣接グラフの密度が低い問題は高い問題に比べて難しい問題である．この問題に対処すべく，動的な対称性除去の制約条件を追加したところ，効率よく解けることがわかった．また，ハイパーグラフ制約条件を追加したところ，平均的には計算時間が改善がみられなかったが，効率良く解ける問題が存在することを示すことができた．その他に，２進表現により状態を表現する方法を導入したところ，これも平均的には計

算時間の改善が見られなかったが，効率良く解ける問題が存在することを示すことができた．

以上の結果より，最小無矛盾 DFA 問題には SAT を使ったアプローチが効果的であり，対称性除去やハイパーグラフ制約条件のようなアドホックな条件を追加することが有効である．そして，２進表現など表現モデルにバリエーションを持たせることが効率化につながることがわかった．

2. 移動順最適化問題

4 章では半導体露光装置における移動順最適化問題について研究した．この問題については，MILP,TSP および MaxSAT によるアプローチを議論した．MILP によるアプローチは最も問題を直接的に表現しているのだが，計算時間は長くなってしまった．これは，部分巡回路の問題が原因である．

MILP と比べ，TSP ソルバーを使って解く手法が効果的であった．そこで，露光頂点を８，６，４頂点で表現するモデルを提案した．この中では，頂点数が多くなってしまうという欠点を持つものの，８頂点モデルが平均的には効率がよかったが，問題によっては６頂点モデル，４頂点モデルが良い場合があることを示すことができた．

以上の結果より，移動順最適化問題では TSP ソルバーのような問題に特化したソルバーを使うことが効果的であり，問題を表現するモデルにバリエーションを持たせることが効率化につながることがわかった．MaxSAT に関しては，部分巡回路除去を行わない場合の巡回セールスマン問題に対する予備実験を行った．この結果から，現状では巡回セールスマン問題を解くのに MaxSAT は適切ではないことを示した．

3. ナーススケジューリング問題

5 章ではナーススケジューリング問題について研究した．ここでは，MILP と SAT によるアプローチを議論した．

ナーススケジューリング問題ではペナルティや勤務数などをカウントする必要があるため，SAT においては和を計算するメカニズムを導入した．実験結果として，日勤・夜勤・深夜勤のあるような３シフトの問題について，MILP を用いる手法では下界値が上がらない問題から厳密解が得られにくいが，SAT による手法は効率よく厳密解を求めることができた．また，日勤・夜勤を扱う２シフトの問題について，ペナルティが大きい場合は SAT によって効率よく厳密解を求めることができなかったが，MILP は効率よく厳密解を求めることができた．

この結果は，３シフト問題のような選択肢が比較的多い問題の場合，MILP は効率的でないことを示している．また，ペナルティが大きい場合，SAT による解法は MILP による解法に比べて効率的でなかったが，これは SAT が和を扱うのが苦手であること

表 6.1: 各アプローチについての各問題のまとめ

問題	MILP	SAT
最小無矛盾 DFA	×初期下界値と最小状態数のギャップが大きい．下界値と厳密解が一致しないため適用困難.	◯すべての状態にラベルがあるベンチマーク問題については効率的に解が求まる．最大クリークと最小状態数のギャップが大きい時は適用困難.
移動順最適化	◯ショット数が少ない時は解が効率的に求まる．非対称の場合効率的ではない.	×巡回セールスマン問題の解を求めるが難しい.
ナーススケジューリング	◯２シフトの場合効率的に解を求められる．３シフトでは下界値と厳密解が一致しない.	◯３シフトの場合効率的に解を求められる．ペナルティが大きい場合効率的ではない.

が原因である．さらに，ナーススケジューリング問題については和の表現法として，論理加算による方法と一対一表現による方法を比較した．SAT による実行時間に関しては，論理変数・論理式共に一対一表現が多くなるが，計算時間に関しては一対一表現によるもののほうが効率がよかった．これは問題記述が直接的なものほど効率が良いことを示していると考えられる．

6.2 総括

本節では汎用ソルバーを使って組合せ最適化問題を効率的に求める手法を総括する．二つのソルバーに関しての各問題の傾向を表 6.1 にまとめる．二つのソルバーの比較に関しては，最小無矛盾 DFA が SAT によるアプローチが有効，移動順最適化問題は MILP によるアプローチが有効，ナーススケジューリング問題はインスタンスによって異なるという顕著な違いが得られた．

表 6.2 に書く問題のサイズを示す．問題のサイズは，変数の数と節数によって示しているが，問題ごとに式の形が異なっているので直接的な比較はできない．しかし，問題の比較の役にはたつと考えられる．特に節数と変数の数の比に着目すると，これらの問題はそれぞれ異なった特徴を持っている事がわかる．移動順最適化問題は特殊で，MILP と SAT での比に大きな違いがある．最小無矛盾 DFA は両者ともに比が大きい．これに対して，ナーススケジューリング問題は比が小さい問題である．比が小さいということは，変数の数に比べて節

表 6.2: 問題の大きさの比較

問題		MILP	節数/変数	SAT	節数/変数
最小無矛盾 DFA 問題	変数 節数	800–7,000 2,500–240,000	2–25	8,000–70,000 5,000–320,000	6–46
移動順最適化問題	変数 節数	57,000–76,000 1,000–1,300	0.2	57,000–76,000 605,000–1,141,000	11–15
ナーススケジューリング問題	変数 節数	6,000–10,000 15,000–27,000	0.2	150,000–330,000 500,000–1,124,000	0.3

表 6.3: 目的関数と問題の特徴

問題	目的関数	特徴
最小無矛盾 DFA 問題	状態数の和,20 変数程度	MILP においては MaxMin の形式
移動順最適化問題	アークの重み付き和，6000 変数前後	ほとんどの変数が目的関数に現れる
ナーススケジューリング問題	ペナルティ変数の和，1300 変数程度	SAT では和を特別に表現している

数が少ないということである．表と前節での各問題に対する結果を比べると，この比が小さい問題は MILP によって解きやすいという傾向が予想される．

表 6.3 に，目的関数の特徴について示した．最小無矛盾 DFA 問題は変数の数に比べて，目的関数に現れる変数の数が少ない問題であるといえる．このことから考えると，MILP によるアプローチでは，目的関数に多くの変数が現れる問題のほうが効率的に解けることが予想される．

次に本論文で扱った三つの問題をグラフ彩色問題として考えてみよう．図 6.1 に各々の問題を隣接グラフで表現したときの模式図を示す．各々の問題の実行可能解は，この隣接グラフのグラフ彩色問題の実行可能解となっている．a) の最小無矛盾 DFA 問題はそのままグラフ彩色問題として考えることができる．b) の移動順最適化問題は経路を隣接辺とした隣接グラフとしても考えることができる．この隣接グラフにおける辺彩色と TSP の実行可能解であるハミルトン閉路の関係はよく研究されている (例えば,[86])．c) のナーススケジューリング問題では，各々の看護師に各々の日に対してシフトを割り当て，例えば二人の看護師を同じシフトに割りつけないような制約条件を与える．この場合は図に示したような隣接グラフとして，同じ色に割り当たった割り当て方が実行可能解となる．このように三つの問題をグラフ彩色問題として見てみることは難しさを考える上で興味深い．

本論文では，ソルバーをブラックボックスとして扱い，問題の定式化方法によって効率的

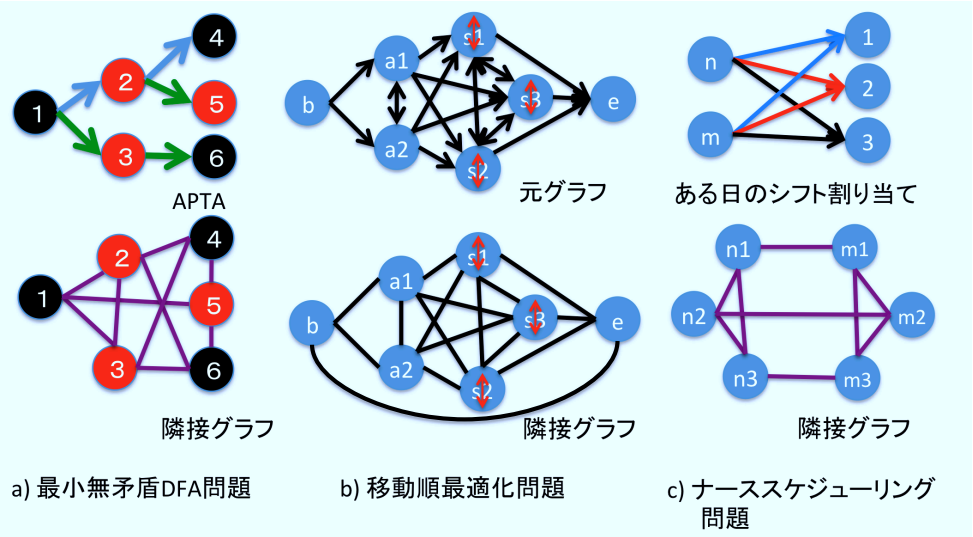


図 6.1: 三つの問題の隣接グラフによる表現

に解を求める手法について研究した．いずれの問題においても，問題の表現方法にバリエーションを持たせることが効率的に問題を解く鍵となることがわかった．近年，ソルバーの進歩には眼を見張るものがあるが，今後の課題としては，ソルバーの実行制御に対して問題依存の手法を取り入れることが考えられる．

6.3 まとめ

本論文では，組合せ最適化問題の厳密解を効率良く求める手法を確立するため，三つの事例研究を行った．その各々では，従来手法を上回る時間的効率性を得ることができた．

特に，MILP ソルバーと SAT ソルバーを中心に，解の効率性を比較した．これまで，このようなソルバーの違いによる効率性の違いの研究は多くない．問題によって，これらのソルバーの効率性に非常な違いが見られたことは興味深い．

最後に，組合せ最適化問題の厳密解法は近似解法に比べて，どのような問題でもある程度短時間で解けるというものではないが，今後のハードウェアや汎用ソルバーの性能向上により，解ける問題は着実に増えていくと思われる．新たな組合せ最適化問題の厳密解を得るためには，まず，汎用ソルバーの利用を検討するのがよいと思われるが，その際に本論文がその一助となれば幸いである．

謝辞

本論文は，著者が総合研究大学院大学複合科学研究科情報学専攻博士課程に在学中（平成１９年４月〜平成２４年３月）の研究成果をまとめたものです．

国立情報学研究所の相澤彰子教授には，学生としてはルーズな私を暖かい目で見てもらい，私をまともな方向に導く様々なご意見をいただきました．休学期間も長かったのに，研究活動を続けてこられたのは誠に先生のお陰であり，感謝します．

同研究所の速水謙教授には，指導教員として，面倒を見ていただきました．教務上の手続きでは迷惑をかけどうしで，恐縮するばかりです．

前東京農工大学工学府の品野勇治准教授 (現在，ZIB) には，もう１０年以上公私にわたってお世話をいただきました．著者が東京農工大を退職した後も自由に研究活動を続けてこられたのは先生のご助力のお陰で有り，感謝します．

成蹊大学理工学部の池上敦子教授には，研究に関して多くのご意見をいただきました．また，先生の学生さんたちと交流する機会を作って下さり，感謝します．

キヤノン株式会社宇都宮事業所の深川容三様には，研究において有益なご意見を頂いたので，感謝します．

東京農工大学工学府の小谷善行教授には，同大学在職中には一方ならぬご助力を頂いたので，感謝します．

国立情報学研究所の大山敬三教授，佐藤健教授，井上克己教授，宇野毅明准教授，定兼邦彦准教授には，博士論文審査委員として，本論文の執筆および口頭発表に対して，有益なご意見を頂いたので，感謝します．

相澤研究室の松林優一郎誌，原忠義氏，亀田尭宙氏，Minh-Quoc Nghiem 氏をはじめ，多くの皆さんには短い期間ではありましたが，研究室での楽しい時間を過ごさせていただいたことに感謝します．

家族には様々な面で負担をかけてしまい申し訳なく思っています．自分勝手な生活を送ってきたのに，比較的和やかな時間を過ごせたのは，妻とねの協力のおかげであり，いくら感謝をしてもしたりないと思っています．

最後に，私の先行きを心配しながら平成２３年１０月に亡くなった父瀧口千代吉の冥福を祈り，この論文を捧げます．

参考文献

[1] T. Achterberg: “Constraint intger programming”, PhD Thesis, Technische Universitȃt Berlin, 2007.

[2] A. V.Aho, R. Sethi and J. D. Ullman: “Compilers: Principles”, Techniques, and Tools, Addison-Wesley, 1986.

[3] D. Applegate, R. Bixby, V. Chvátal, W. Cook: “TSP Cuts Which Do Not Conform to the Template Paradigm”, Computational Combinatorial Optimization, Optimal or Provably Near-Optimal Solutions, volume 2241 of Lecture Notes In Computer Science,261–304, 2000.

[4] N. Barnier and P. Brisset: “Graph coloring for air traffic flow management”, Ann. Oper. Res. 130–163,2004.

[5] J. F. Bard, H. W. Purnomo: “Preference Scheduling for Nurses using Column Generation”, European Journal of Operational Research, **164**, **2**, 510-534, 2005.

[6] A. D. Belegundo and T. R. Chandruplatla: “Integer and dispcrete programming in Optimization concepts and applications in engineering (second edition)”, Cambridge University Press, 1999.

[7] D. Le Belle: “A brief introduction to practical SAT solving – What can I exprect from SAT today? –”, SAT workshop, 2011. (presentation material)

[8] A. Blumer, A. Ehrenfeucht, D. Haussler, M.K. Warmuth: “Occam’s razor”, Information Processing Letters, 24, 377-380, 1987.

[9] S. Boyd and L. Vandenberghe: “Localization and Cutting-plane Methods”,http://see.stanford.edu,05-localization_methods_notes.pdf, 2003.

[10] P. Brucker, E. Burke, T. Curtois, R. Qu, G. V. Berghe: “Adaptive Construction of Nurse Schedules: A Shift Sequence Based Approach”,NOTTCS-TR-2007-1, University of Nottingham, 2007.

[11] E. D. Burke, P. D. Causmaecker, G. V. Berghe, H. V. Landeghem: “The State of the art of Nurse Roestering”, J. of Scheduling, **7**, 441-499, 2004.

[12] E. Burke, J. Li, R. Qu: “A Pareto-Based Search Methodology for Multi-objective Nurse Scheduling”, Annals of Operations Research, Springer, 2008.

[13] R.Campelo, Y.Correa, Frora: “Colique, holes and the vertex coloring polytope”, Information Processing Letters, 89, 159–164,2004.

[14] G. Chaitin: “Register allocation and spilliting via graph coloring”, 20 Years ACMSIGPLAN Conference on Programming Language Design and Implementation: A selection 2003, 66–74, 2003.

[15] Y-F. Chen, A. Farzan, E. M. Clarke, Y-K. Tsay and B-Y. Wang: “Learning Minimal Separating DFA ’ s for Compositional Verification”,TACAS 2009, LNCS 5505, 31-45, 2009.

[16] O. Cicchello, S.C. Kremer: “Inducing Grammars from Sparse Data Sets: A Survey of Algorithms and Results”, J. of Machine Learning Research, 4, 603-632, 2003.

[17] http://www.cs.uni-potsdam.de/clasp/.

[18] P. Coll, J. Marenco, I. M. Díaz and P. Zabala: “Facets of the graph coloring polytope”, Annals of Operations Research, 116, 79–90, 2002.

[19] Cook, W.,“Concorde Home”,available from ⟨http://www.tsp.gatech.edu/concorde.html⟩.

[20] S. A. Cook: “The complexity of theorem proving procedures”, Third Annual ACM Symposium on the Theory of Computing, 151–158, 1971.

[21] F. Coste and J. Nicolas: “Regular Inference as a graph coloring problem”, Workshop on Grammatical Inference, Automata Induction and Language Acquisition (ICML’97), 1997.

[22] T. Curtois: “Staff Rostering Benchmark Data Sets”, http://www.cs.nott.ac.uk/˜tec/NRP.

[23] http://www-06.ibm.com/software/jp/websphere/ilog/optimization/core-products-technologies/cp1/.

[24] http://www-06.ibm.com/software/jp/websphere/ilog/optimization/core-products-technologies/cplex/.

[25] M.Dalal: “An Almost Quadratic Class of Satisfiability Problems”, 12th European Conference on Artificial Intelligence, 355–359, 1996.

[26] P. Dupont: “Regular Grammatical Inference from Positive and Negative Samples by Genetic Search: The GIG Method”, Proc.Grammatical Inference and Applications: Second Int’l Colloquium (ICGI-94), R.C. Carrasco and J. Oncina, eds., 236–245, 1994.

[27] M. Fischetti, A. Lodi, D. Salvagnin: “Just MIP it”, Matheuristics-Hybridizing Metaheuristics and Mathematical Programming-, 39-70, Springer, 2009.

[28] Fujimura, K., Fujiwaki, S., Tokutaka, H.: “Optimization of High Speed Chip-Mounter Using SOM-TSP Method”,The transactions of the Institute of Electronics, Information and Communication Engineers,Vol.J84-D-II, No.6, 1194-1202, 2001.

[29] M. R. Garey, D. S. Johnson: Computers and Intractability; A Guid to the Theory of NP-Completeness, W. H. Freeman & Co. New York, 1990.

[30] M. Gebser: “Answer Set Programming”, http://jcai-11.iiia.csic.es/files/proceedings/T04-asp.pdf,2004.

[31] M. Gebser, B. Kaufmann, A. Neumann and T. Schaub: “Conflict-driven answer set solving”, IJCAI2007, 386–392, 2007.

[32] A. V. Gelder: “Another look at graph coloring via propositional satisfiability”, Discrete Applied Mathematics, 156(2), 230–243, 2008.

[33] E. M. Gold: “Complexity of automaton identification from given data”, information and control, 3(37), 302-420, 1978.

[34] M. Grőtschel, M. W. Padberg: “On the symmetric travelling salesman problem I: lifting theorem and facets”, Mathemaical Programming, 16, 265–280, 1979.

[35] M. Grőtschel, M. W. Padberg: “On the symmetric travelling salesman problem II: lifting theorem and facets”, Mathemaical Programming, 16, 281–302, 1979.

[36] S. Gualandi and F. Malucelli: “Exact solution of graph coloring problems via constraint programming and column generation”, INFORMS journal on Computing(Online), 2011.

[37] Gutin, G. and Abraham, P. ed.: “The Traveling Salesman Problem and its Variation”, Kluwer Academic, 2002.

[38] P. Hausen, M. Labbe and D. Schindl: “Set covering and packing formulations of graph coloring: algorithms and first polyhedral results”,Discrete Optimization, 6(2), 135–147, 2009.

[39] K. Helsgaun: “An effective implementation of the Lin-Kernighan traveling salesman heuristic”, European Journal of Operational Research, 126, 106–130, 2000.

[40] F. Heras, J. Larrosa and A. Oliveras: “MiniMaxSat: a new weighted Max-SAT solver”, International Conference on Theory and Applications of Satisfiability Testing, 41–55, 2007.

[41] M.J.H. Heule,S.Verwer: “Exact DFA Identification Using SAT Solvers”, Grammatical Inference: Theoretical Results and Applications, Lecture Notes in Computer Science, 66-79, 2010.

[42] J. N. Hooker: “A principled approach to mixed integer/linear problem formulation”, Operations Research and Cyber-Infrastructure, Operations Research/Computer Science Interfaces Series, 47, II, part 1, 79–100, 2009.

[43] A. Ikegami, A. Niwa : “A Subproblem-centric Model and Approach to the Nurse Scheduling Problem”, Mathematical Programming, 97,517–541, 2003.

[44] 池上敦子: “ナース・スケジューリング-調査・モデリング・アルゴリズム-”, 数理統計,**53**,**2**,231–259, 2005.

[45] 井上 克己，田村 直之: SAT ソルバーの基礎．人工知能学会誌，25(1)，57–67, 2010.

[46] P. Kelsen, S. Mahajan and H. Ramesh: “Approximate hypergraph coloring”, Algorithm Theory - SWAT’96, LNCS 1097, 41–52, 1996.

[47] N.Inui, Y.Shinano: “Minimizing State Transition Model for Multiclassification by Mixed-Integer Programming”,Mexican International Conference on Artificial Intelligence(Springer LNAI 3789),473–482, 2006.

[48] Y.Shinano, N.Inui, Y.Fukagawa, N.Takakura: “An Application of Travelling-Salesman Models to Shot Sequence Generation for Scan Lithography”, 8th World Congress on Computational Mechanics, 5th European Congress on Computational Methods in Applied Sciences and Engineering, CD-ROM, 2008.

[49] 乾伸雄, 品野勇次, 深川容三, 高倉伸: “走査型半導体露光装置における移動準最適化”, 日本機械学会論文集,76, 770, 2578-2583, 2010.

[50] 乾伸雄, 池上敦子: “ナーススケジューリング問題における混合整数線形計画問題と充足可能性判定問題による厳密解法の比較”, オペレーションズ・リサーチ　55(11), 707-712, 2010.

[51] 乾伸雄, 池上敦子: “SAT を用いたナーススケジューリング問題の厳密解法の評価”, 日本オペレーションズ・リサーチ学会春季研究発表会,2-B-3,2010.

[52] N.Inui, A.Ikegami: “Using SAT solver for nurse scheduling problem”, International Symposium on Scheduling 2011, 2011(7).

[53] N. Inui, A. Aizawa: “A Hybrid Approach for Exact Solving of Minimum Consistent-DFA Generation Problem”, IPSJ SIG Notes, 2011-MPS-82(1), 1–6, 2011.

[54] 乾伸雄, 相澤彰子: “SAT ソルバーを用いた最小無矛盾 DFA の生成”, 人工知能学会論文誌,27(3), 2012.

[55] 越村三幸, 鍋島英和, 藤田博, 長谷川隆三: “SAT 変換による未解決ジョブショップスケジューリング問題への挑戦”, スケジューリング・シンポジウム 2009,209-213, 2009.

[56] 黒田陽之, 平山勝敏:“Multi-MaxSAT:ラグランジュ分解・調整法を用いた Max-SAT の解法”, 合同エージェントワークショップ＆シンポジウム２００７, 2007.

[57] S. Kundo, S. Achayya: “A SAT Approach for Solving the Nurse Scheduling Problem”, TENCON 2008, 1-6, 2008.

[58] B.Lambeau, C.Damas, P.Dupont: “State-Merging DFA Induction Algorithms with Mandatory Merge Constraints”, ICGI 2008, LNAI 5278, 139–153, 2008.

[59] K.J.Lang, B.A.Pearlmutter and R.A.Price: “Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm”, Grammatical Inference, LNCS 1433, 1–12, 1998.

[60] K.J.Lang: “Faster algorithms for finding minimal consistent DFAs”, Technical report, NEC Research Institute, 1999.

[61] J. K. Lenstra and A. H. Rinnooy Kan: “Some simple applications of the travelling salesman problem”, Operational Research Quarterly, 26, 4,717–733, 1975.

[62] M. Leucker: “Learning meets verification”, In Formal methods for components and object(F. S. de Boer, M.M.Bonsangue, S.Graf and W. P. de Roever (Eds), LNCS 4709, 127–151, 2006.

[63] V. Lifschitz: “What is answer set programming?”, AAAI’2008, 1594–1597, 2008.

[64] http://fmv.jku.at/lingeling/.

[65] S. Lin and B. W. Kernighan: “An effective heuristic algorithm for the traveling-salesman problem”, Operation Research, 21, 498–516, 1973.

[66] E. Malaguti and P. Toth: “A survey on vertex coloring problems”, International transactions in operational research, 17, 1–34, 2010.

[67] http://www.maxsat.udl.cat/.

[68] A. Mehrotra and M.Trick: “A column generation approach for graph coloring”, Informs Journal of Computing, 8(4), 344-353, 1996.

[69] I.Méndez-Díaz, P. Zabala: “A cutting plane algorithm for graph coloring”, Discrete Applied Mathematics, 156, 159–179, 2008.

[70] The MiniSat Homepage,http://minisat.se/.

[71] http://miplib.zib.de/miplib2003/index.php.

[72] T. Miyaji: “Projection Exposing Device”,Japanese Patent Dislosure H09-015872 (in Japanese), 1997.

[73] 宮代隆平: “ここまで解ける整数計画”, システム／制御／情報,50(9), 363–368, 2006.

[74] 鍋島 英知，宗 剛秀：高速 SAT ソルバーの原理，人工知能学会誌，25(1), 68–76, 2010.

[75] K. Nonobe, Ibaraki: “A tabu search approach to the constraint satisfaction problem as a general problem solver”, European Journal of Operational Research, 106, 599-623, 1998.

[76] 野々部宏司, 池上敦子:“ナーススケジューリング問題における解の評価”, スケジューリング・シンポジウム 2009,163-167, 2009.

[77] “Nurse Roestering Competition“: http://www.kuleuven-kortrijk.be/nrpcompetition.

[78] A.L.Oliveira, J.P.M.Silva: “Efficient Algorithms for the Inference of Minimum Size DFAs”, Machine Learning, 44(1–2), 93–119, 2001.

[79] G. Palubeckis: “On the graph coloring polytope”, Information technology and control, 37(1), 7–11, 2008.

[80] P. M. Pardalos, J. Rappe, Mauricio and M.G.C. Resende: “An exact parallel algorithm for the maximum clique problem”, In High Performance and Software in Nonlieanear Optimization, 279–300, Kluwer Academic, 1998.

[81] S. Porat and J. A. Feldman : “Learning automata from ordered examples”, Machine Learning, 7, 109-138, 1991.

[82] M. R. Prasad, A. Biere and A. Gupta: “A Survey of Recent Advances in SAT-Based Formal Verification”, INTERNATIONAL JOURNAL ON SOFTWARE TOOLS FOR TECHNOLOGY TRANSFER (STTT), 7(2), 156–173, 2005.

[83] R. Qu, F. He: “A Hybrid Constraint Programming Approach for Nurse Rostering Problems”, AI’08, 211-224, 2008.

[84] A. Ramani, I. L. Markov, K. A. Sakallah and F. A. Aloul: “Breaking Instance-Independent Symmetries in Exact Graph Coloring”, J. of Artificial Intelligence Research, 26, 289–322, 2006.

[85] J.-c. Régin: “Solving the maximum clique problem with constraint programming”, CPAIOR’03,2003.

[86] 猿渡康文: “辺彩色グラフにおける交互性について”, 情報処理学会数理モデル化と問題解決研究報告，MPS-97-113,25–30,1997.

[87] “The international SAT Competition web page”, http://www.satcompetition.org.

[88] http://baldur.iti.uka.de/sat-race-2010/.

[89] http://www.sat4j.org.

[90] A. Sharieh, W. Al_Rawagepfeh, M. H. Mahafzah and A. Al Dahamsheh: “An algorithm for finding maximimum independent set in a graph”, European Journal of Scientific Research, 23(4), 586–596, 2008.

[91] J. R. Thornton, A. Sattar: “Nurse Rostering and Integer Programming Revisited”, ICCIMA’97, 49-58, 1997.

[92] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

[93] O. Unold: “Regular Language Induction With Grammar-based Classifier System”, in Engineering the Computer Science and IT (S.Soomro ed.), 13-22, InTech, 2009.

[94] M. Vanhoucke, B. Maenhout: “On the Characterization and Generation of Nurse Scheduling Problem Instances”, European Journal of Operational Research, **196**, **2**, 457-467, 2009.

[95] Yoshida, K.,“Method for Deciding Moving Sequence”,*Japanese Patent Disclosure* H10-303126 (in Japanese), 1998.

[96] L. Zambito: “The traveling salesman problem: A comprehensive survey”, Submitted as a project for CSE4080, 2006.

本論文に関係する業績

ジャーナル論文

[1] 乾伸雄, 品野勇次, 深川容三, 高倉伸: “走査型半導体露光装置における移動準最適化”, 日本機械学会論文集,76, 770, 2578-2583, 2010. (4.1,4.2,4.3,4.5,4.6, 各節に関連)

[2] 乾伸雄, 池上敦子: “ナーススケジューリング問題における混合整数線形計画問題と充足可能性判定問題による厳密解法の比較”, オペレーションズ・リサーチ　55(11), 707-712, 2010. (5.1,5.2,5.3,5.4, 各節に関連)

[3] 乾伸雄, 相澤彰子: “SAT ソルバーを用いた最小無矛盾 DFA の生成”, 人工知能学会論文誌,27(3), 2012.(3.1,3.2,3.3,3.5.1,3.5.3, 各節に関連)

査読付き国際会議

[1] N.Inui, Y.Shinano: “Minimizing State Transition Model for Multiclassification by Mixed-Integer Programming”,Mexican International Conference on Artificial Intelligence(Springer LNAI 3789),473–482, 2006. (3.4 節に関連)

[2] Y.Shinano, N.Inui, Y.Fukagawa, N.Takakura: “An Application of Travelling-Salesman Models to Shot Sequence Generation for Scan Lithography”, 8th World Congress on Computational Mechanics, 5th European Congress on Computational Methods in Applied Sciences and Engineering, CD-ROM, 2008. (4.7 節に関連)

[3] N.Inui, A.Ikegami: “Using SAT solver for nurse scheduling problem”, International Symposium on Scheduling 2011, 2011. (5.6 節に関連)

査読なし研究会

[1] 乾伸雄, 池上敦子: “SAT を用いたナーススケジューリング問題の厳密解法の評価”, 日本オペレーションズ・リサーチ学会春季研究発表会,2-B-3,2010.

[2] N. Inui, A. Aizawa: “A Hybrid Approach for Exact Solving of Minimum Consistent-DFA Generation Problem”, IPSJ SIG Notes, 2011-MPS-82(1), 1–6, 2011. (3.5.2 節に関連)