大学の時間割編成問題を記述可能なDSLと時間割作成システムの提案

松田 陸斗[1]　乃村 能成[2]

概要：現在，少子化による大学再編の必要性やCOVID-19の感染拡大防止策による授業のオンライン化によって，大学の時間割を再編成したいという要求が増している．このため，大学の時間割作成システムの需要が高まっている．しかし，大学によって，時間割の制約が異なると考えられているため，時間割編成問題の表現方法の汎化が大きな課題である．そこで，本研究では，時間割編成問題を記述可能なDSLを提案し，これを用いた時間割作成システムを提案する．DSLの評価として，実際の時間割編成問題を提案DSLで記述した際の字句数を評価し，実用に足ると判断した．また，時間割作成システムは対話的に利用される場合が考えられるため，時間割編成問題の求解時間が用途に耐えるか調査を行った．結果として，SATソルバを用いて26.6 msと短時間で求解できた．

キーワード：時間割編成問題，ドメイン特化言語，SATソルバ

1. はじめに

大学における時間割作成は，未だに手作業で行われることが多く，多大な労力を要している．このため，大学の時間割作成システムに関する研究は行われてきた[1][2]．しかし，これらの中で広く普及しているものはない．また，現在の情勢では，少子化の影響により大学の統合や再編成が進んでおり[3]，さらに，COVID-19の感染拡大防止策として，多くの大学の授業がオンライン化している．このような情勢に伴い，大学の時間割再編成の機会が増えたため，今まで以上に大学の時間割作成システムが求められている．しかし，大学の時間割モデルは大学毎に異なるため，時間割編成問題の一般化は困難である．また，これまでに提案されている大学の時間割作成システムは，特定の大学の時間割モデルに特化したものであり，これを他の大学の時間割作成に用いるのは難しい．

そこで，大学の時間割編成問題の構造を調査し，大学の時間割編成問題を記述可能なドメイン特化言語（DSL; Domain-Specific Language）**AUK**を定義し，これを用いた時間割作成システムを提案する．AUKの文法は，時間割編成問題に関する国際競技会であるInternational Timetabling Competition 2019（ITC2019）[4]の提案を参考にしている．

AUKを提案することによる貢献の一つは，時間割の微調整が容易になることである．時間割作成の際には，教員からの要望等による時間割の変更要求が発生する．しかし，制約が複雑に絡み合った時間割を修正するのは容易でない．ここで，DSLを用いて制約を記述していた場合，入力する制約を変更するだけで時間割の修正が可能になる．したがって，時間割作成システムを用いて対話的に時間割作成を行う場合が考えられる．この場合，時間割作成システムは短時間で時間割作成可能でなければならない．そこで，本稿では，時間割編成問題の求解アルゴリズムにSATソルバを用いて，実際の時間割作成にかかる時間を調査する．SATソルバとは，充足可能性問題を高速に解くプログラムであり，SATソルバを用いることで時間割編成問題の高速な求解が期待される．

以降では，2章で時間割編成問題の構造を述べる．3章でAUKについて述べる．4章でSATソルバによる時間割編成問題の求解時間を調査する．5章で時間割作成システムの全体像を述べる．6章で関連研究について述べる．

2. 時間割編成問題

2.1 時間割編成問題とは

時間割編成とは，時間割を構成する要素（以下，**時間割資源**と呼ぶ）の組合せを決定する問題である．時間割資源には，時間枠，学生，教室，教員，授業等が考えられる．時間割編成の際には，通常，時間割資源の組合せを制限するような制約（以下，**時間割制約**と呼ぶ）が課せられる．

[1] 岡山大学大学院自然科学研究科
[2] 岡山大学学術研究院自然科学学域

©2021 Information Processing Society of Japan

したがって，時間割編成問題は，制約を満たすような時間割資源の組合せを決定する問題といえる．

ここで，一般的に時間割という単語は，2 つのものを指す．1 つは，時間割資源のスケジュールのことであり，もう 1 つは，これを表にしたものである．時間割編成の際に，作成する時間割は前者であり，後者は時間割の利用者ごとに存在する．時間割の利用者には，学生，教室，教員等が考えられる．以降，時間割資源のスケジュールを時間割と呼び，時間割資源のスケジュールの表を**時間割表**と呼ぶ．

学生が利用する時間割表は，小中高校では学級ごとに存在するが，大学では学生ごとに存在する．そして，この時間割表の観点から，時間割の変更要求が発生し，これが時間割制約になる．このため，大学の時間割編成では時間割制約が膨大になる．この問題を避けるために，大学の時間割編成においては，学生を時間割の利用者から除外する，もしくは，架空の学生を見立て，適切な要求を発生させることで，妥当な時間割を作成している．

2.2 時間割資源の種類

本稿では，大学における時間割編成問題の時間割資源は教室，教員，授業，および時間枠の 4 つとする．時間割資源に学生を考慮しない理由としては，日本の大学では，学生を時間割資源として考えることは少なく，提案する時間割作成システムを実状に即したものにするためである．

2.3 時間割制約の分類

2.3.1 ドメイン制約

時間割資源の組合せの総数は，各時間割資源数の積で求められる．しかし，資源どうしの組合せには，無効な組合せも考えられるため，これを制約によって定める．ここで，時間割資源の組合せのうち，有効なものの集合を**ドメイン**と呼ぶ．また，ドメインの範囲を定めるような制約を**ドメイン制約**と呼ぶ．ドメインとドメイン制約の具体例を次に示す．

ドメイン制約: 授業「化学」は教室「実験室」でのみ開講できる．

ドメイン: 「化学」が「実験室」で開講される．

ドメイン外: 「化学」が「講義室」で開講される．

このように，ドメイン制約は，ある時間割資源の特性から発生する制約といえる．

2.3.2 競合制約

ドメイン制約に対して，時間割資源間の競合に関する制約を**競合制約**と呼ぶ．競合とは，複数の時間割資源が他の時間割資源によって制限を受けることである．たとえば，授業が時間枠から競合制約を受ける場合，「授業 A，授業 B は同じ時間枠に開講できない」という例がある．また，競合制約には，時間割の利用者の要求から発生する制約が含まれる．

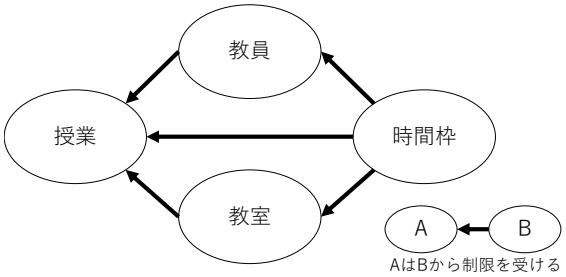

図 1 時間割資源間の制限関係図

2.4 時間割制約の制限関係

ドメイン制約は，時間割資源間に課される．これによって，**図 1** に示す従属関係が生じる．図 1 は以下のことを表す．

(1) 授業は教員，教室，および時間枠から制限を受ける

(2) 教員と教室は時間枠から制限を受ける

(3) 時間枠は他の資源から制限を受けない

これらの分析に基づいて 3 章で AUK の文法を提案する．

3. AUK

3.1 AUK の概要

既存の時間割編成問題の記述言語として ITC2019 で提案された XML フォーマットがある（以下，**ITC フォーマット**と呼ぶ）．ITC フォーマットは学生の履修希望を満足する時間割を編成する問題を扱うことを意図して設計された．一方，日本の大学では，通常，学生の履修希望は時間割に反映されない．このため，ITC フォーマットは日本の時間割編成問題に即しているとは言えない．また，ITC フォーマットは，人による記述を想定していないため，冗長な記法が多い．したがって，ITC フォーマットを時間割編成問題の DSL に流用するのは難しい．

ここで，ITC フォーマットの問題点を解消した時間割編成問題の DSL である AUK を提案する．すなわち，AUK とは，2 章の分析結果より，ITC フォーマットを日本の大学の時間割編成問題に適合し，かつ，ITC フォーマットから冗長性が削減され，かつ，人が記述することに即したものである．

AUK は，時間割作成者と時間割作成システムを仲介するフォーマットであるため，入力フォーマットと出力フォーマットがある．入力フォーマットでは，時間割資源，時間割制約を記述できる．一方，出力フォーマットでは，求解した結果である時間割資源の割当てが表される．

©2021 Information Processing Society of Japan

3.2 冗長性の削減

ITC フォーマットから冗長性を削減するための工夫として，以下の 2 つを導入する．

(1) 時間割資源の集合の導入

ドメイン制約や競合制約を記述する際，共通する属性を持った複数の時間割資源を集合として指定したいという要求がある．たとえば，少人数で行われる授業は，定員の少ない教室の集合「小講義室」を指定したい．この集合は，同一のものが複数回現れることが多い．ここで，複数の時間割資源の集合記法を導入することで冗長性を排除できると考えられる．

(2) 二次元表を指向した時間枠指定方法の最適化

時間枠とは，授業時間の範囲のことである．ITC フォーマットにおいて，時間枠の指定は，時間枠の列挙によって記述できる．しかし，時間枠数は，(1 週間の授業日数) × (1 日の授業数) であるため，単純な時間枠の列挙は冗長になりやすい．ここで，時間枠が曜日と時限の二次元表で表せることに着目し，時間枠指定の記述方法をスプレッドシートのセル参照記法に基づいて定義する．これによって，より簡潔な記述が期待できる．

3.3 AUK の入力フォーマット

AUK は Ruby の内部 DSL として実装されるため，AUK の構文は Ruby の構文に準ずる．AUK での予約語は，時間割資源，競合制約を表す**ブロック**と，各ブロックについてのドメイン制約等を表す**プロパティ**に分けられる．AUK で定義している予約語を**表 1** に示す．表 1 のブロック `period`，`term` のプロパティ名は，助数詞を受け付ける．また，表 1 の（競合制約名）には，**表 2** に示す ITC2019 で提案された競合制約名が入るものとする．競合制約は，実際の利用シーンから発生するものであり，現時点では十分に議論できていない．そのため，これら競合制約の種類については，今後議論が必要である．

また，以下に各時間割資源の記述例を示す．

時間枠の記述: 時間枠は，1 週間の授業日数（`nr_days_a_week`）と 1 日に行われる最大授業数（`nr_periods`）と作成する時間割の学期数（`nr_terms`）によって決まる．また，時間枠（`period`）と時間（`time`）の対応付けと学期（`term`）と日付（`date`）が必要である．時間枠の記述文法を以下に示す．

時間枠の記述例

```
initialize do
  nr_days_a_week 5
  nr_periods 8
  nr_terms 4
end

period do
  first start_time: "8:40",
        end_time: "9:40"
  second start_time: "9:50",
         end_time: "10:50"
  ...
end

term do
  first start_date: "2020/4/1",
        end_date: "2020/6/10"
  second start_date: "2020/6/11",
         end_date: "2020/8/10"
  ...
end
```

この例では，`initialize` 中で週 5 日，1 日 8 時限，4 学期分の時間割作成を行うことを明示している．また，`period`，`term` では，それぞれ時間枠と時間，学期と期間の対応付けを行っている．これによって，授業が行われる日時を明確にしている．

教室の記述: ここでは，各教室について，教室名（`room_name`），ドメイン制約（`unavailable`），および所属グループ（`belongs_to`）を記述する．教室の記述文法を以下に示す．

教室の記述例

```
room "第11講義室" do
  unavailable start_time: "2020/4/11 10:00",
              end_time: "2020/4/11 15:00"
  belongs_to "小講義室"
end
```

この例では，`unavailable` で教室「第 11 講義室」は 2020 年 4 月 11 日の 10 時から 15 時まで使用できないというドメイン制約を記述している．また，`belongs_to` で教室「第 11 講義室」が教室グループ「小講義室」に属するということを記述している．

教員の記述: ここでは，各教員について，教員名（`ins_name`），ドメイン制約（`unavailable`），および所属グループ（`belongs_to`）を記述する．教員の記述文法を以下に示す．

教員の記述例

```
instructor "岡大太郎" do
  unavailable start_time: "2020/5/15 12:00",
              end_time: "2020/5/16 12:00"
  belongs_to "CS"
end
```

授業の記述: ここでは，各授業について，授業名（`lec_name`），ドメイン制約（`rooms`，`instructors`，`period`，`term`），および所属グループ（`belongs_to`）

©2021 Information Processing Society of Japan

表 1　AUK で定義している予約語

ブロック	プロパティ	プロパティの引数	説明
initialize	nr_days_a_week	整数	1 週間のうち授業がある日数
	nr_periods	整数	1 日の時間枠の数
	nr_terms	整数	作成する時間割の学期数
period	first	開始時刻，終了時刻	第 1 時限目に対応する時間
	second	開始時刻，終了時刻	第 2 時限目に対応する時間
	...		
term	first	開始日，終了日	第 1 学期に対応する期間
	second	開始日，終了日	第 2 学期に対応する期間
	...		
room	unavailable	開始時刻，終了時刻	教室が利用できない時間
	belongs_to	文字列	教室が所属する教室グループ
instructor	unavailable	開始時刻，終了時刻	教員が授業できない時間
	belongs_to	文字列	教員が所属する教員グループ
lecture	rooms	教室名｜教室グループ名	授業が開講可能な教室
	instructors	教員名｜教員グループ名	授業を担当可能な教員
	period	時間枠	授業が開講可能な時間枠
	term	整数	授業が開講可能な学期
	belongs_to	文字列	授業が所属する授業グループ
(競合制約名)	lectures	授業名—授業グループ名	（競合制約名）が課せられる授業

表 2　ITC2019 で提案された競合制約

制約名	説明
SameStart	同じ時刻に開始する．
SameTime	同じ時間内に行われる．
DifferentTime	異なる時間に行われる．
SameDays	同じ曜日に行われる．
DifferentDays	異なる曜日に行われる．
SameWeeks	同じ週に行われる．
DifferentWeeks	異なる週に行われる．
SameRoom	同じ教室で行われる．
DifferentRoom	異なる教室で行われる．
Overlap	時間的に重複する．
NotOverlap	時間的に重複しない．
SameAttendees	時間的に重複しない．かつ出席者は次の授業の教室に移動できる．
Precedence	時間的順序が守られる．
WorkDay(S)	同じ週，同じ曜日に行われる授業のうち，最初の授業の開始時刻から最後の授業の終了時刻までの間に S 時間を超えてはならない．
MinGap(G)	同じ週，同じ曜日に行われる授業のうち，ある授業の終了時刻から，他の授業の開始時刻までに少なくとも G 時間空けなくてはならない．
MaxDays(D)	週に関係なく，D 日を超えて開講できない．
MaxDayLoad(S)	1 日あたり S 時間を超えてはならない．
MaxBreaks(R,S)	S 時間を超える休憩時間を 1 日あたり R 回を超えてはならない．
MaxBlocks(M,S)	S 時間以下の休憩時間の場合，同じブロックであるとみなされ，ブロックの長さが M 時間を超えてはならない．

を記述する．授業の記述文法を以下に示す．

授業の記述例

```
lecture "アルゴリズム" do
  rooms "小講義室"
  instructors "岡大太郎"
  period "Mon1:Mon8", "Wed1:Wed8"
  term 1
  belongs_to "必修科目"
end
```

この例では，rooms で授業「アルゴリズム」は，教室グループ「小講義室」で開講可能であるというドメイン制約を記述している．また，instructors で授業「アルゴリズム」は，教員「岡大太郎」が担当可能であるというドメイン制約を記述している．また，period，term で授業「アルゴリズム」は，第 1 学期の月曜 1〜8 限，および水曜 1〜8 限に開講可能であるというドメイン制約を記述している．また，belongs_to で授業「アルゴリズム」は，授業グループ「必修科目」に属するということを記述している．

競合制約の記述: 競合制約は，制約名と制約が課せられる資源（lectures）を記述する必要がある．制約名には，あらかじめ登録されている競合制約名が入る．この競合制約の例として，ITC2019 で提案された 19 個の競合制約がある．これらの競合制約は，2 つ以上の授業の集合に対して，時間枠や教室から制限を受ける制約として定義されている．ITC2019 の競合制約の例として，「複数の授業が同じ時間に開講されない（NotOverlap）」や「複数の授業が同じ教室で開講される（SameRoom）」，「ある 2 つの授業は少なくとも G 時間枠の間隔が必要である（MinGap(G)）」等がある．また，ITC2019 で提案された競合制約以外の例として，

©2021 Information Processing Society of Japan

実際のユースケースから「複数の授業が連続して開講される（NextTime）」が考えられる．競合制約の記述文法を以下に示す．

競合制約の記述例

```
NotOverlap do
  lectures "必修科目"
end
```

この例では，授業グループ「必修科目」に属する授業は同じ時間に開講されない（NotOverlap）という競合制約を記述している．

3.4 AUK の出力フォーマット

時間割の最終的な時間割資源の割り当ては，1 つの授業につき 1 つ発生する．したがって，出力フォーマットは，授業の記述に倣って，以下のようになる．

出力フォーマットの例

```
lecture "アルゴリズム" do
  rooms "第11講義室"
  instructors "岡大太郎"
  period "Mon1"
  term 1
end
```

この例では，授業「アルゴリズム」は，第 1 学期月曜 1 限目に教室「第 11 講義室」で教員「岡大太郎」によって行われることを表している．

3.5 AUK の評価

3.5.1 評価方針

本節では，同じ時間割編成問題を ITC フォーマットで記述した場合の記述量と AUK で記述した場合の記述量を比較評価する．評価に使用する時間割編成問題は，ITC2019 の例題 wbg-fal10 とする．また，岡山大学工学部情報系学科の時間割を AUK で記述した場合の記述量を調べる．ここで，記述量とは，比較対象の各言語に対して字句解析を行い，時間割編成問題の情報を持たない字句（タグ括弧等）を除去した状態の合計字句数である．

3.5.2 ITC2019 の例題を用いた比較評価

ここでは，ITC2019 で提示された時間割編成問題の例題である wbg-fal10 を AUK で記述し，ITC フォーマットと AUK の記述量を比較する．ITC フォーマットでは，学生を時間割資源として考慮している．一方で，AUK では学生を時間割資源として考慮していない．したがって，両者の全文比較は正当な評価にならないため，共通する項目の記述量を比較する．**表 3** に比較結果を示す．表 3 の各言語の合計字句数の比較結果により，AUK は ITC フォーマットに対して記述量の合計を 96% 削減できていることがわかる．この要因としては，3.2 節 (2) の二次元表を指向した時間枠指定方法の最適化よって，授業に関する記述量が大きく削減できたことが挙げられる．

表 3 wbg-fal10 を ITC フォーマットおよび AUK で記述した際の字句数の比較

	教室	授業	競合制約	合計
ITC フォーマット	184	73,920	1,522	**75,626**
AUK	235	2,334	6,16	**3,185**

表 4 wbg-fal10 を AUK で記述した際の時間割資源あたりの記述量

	教室	授業	競合制約
時間割資源数	7	150	82
AUK での記述量	235	2,334	616
時間割資源あたりの記述量（小数点以下切り捨て）	33	**15**	**7**

また，wbg-fal10 の時間割資源あたりの記述量を**表 4** に示す．表 4 の時間割資源あたりの記述量より，特に資源数が多くなる傾向にある授業，および競合制約あたりの記述量は大きく抑えられており，AUK は人が直接記述する DSL としても使用可能であるといえる．

3.5.3 実際の時間割を用いた記述量の調査

ここでは，2020 年度の岡山大学工学部情報系学科の 1 年間の時間割から時間割資源，および時間割制約を抽出し，時間割編成問題として AUK で記述する．このときの記述量，および記述行数を調査する．

2020 年度の岡山大学工学部情報系学科の時間割資源の種類とその数を**表 5** に示す．また，考慮する制約を以下に示す．

(1) ドメイン制約

(a) 授業における教員指定
授業は指定された教員によって行われる．

(b) 授業における教室指定
授業は指定された教室で行われる．

(c) 授業における時間枠指定
授業は指定された時間枠で行われる．

(2) 競合制約

(a) 教員における NotOverlap 制約
同じ教員が担当する授業は時間的に重複しない．

(b) 教室における NotOverlap 制約
同じ教室で開講される授業は時間的に重複しない．

(c) 対象学生が同じ授業における NotOverlap 制約
対象学生が同じ授業は時間的に重複しない．

(d) NextTime 制約
指定した 2 つの授業は連続して開講される．

(e) MinGap(G) 制約
指定した 2 つの授業は少なくとも G 時間枠の間隔を空けて開講される．

これらの条件のもとで，岡山大学工学部情報学科の時間割編成問題を AUK で記述した．結果として，記述量は 899 字句，記述行数は 439 行であった．

©2021 Information Processing Society of Japan

曜日	時限	1 (8:40~9:40)			2 (9:50~10:50)			3 (11:00~12:00)			4 (12:50~13:50)			5 (14:00~15:00)			6 (15:10~16:10)			7 (16:20~17:20)			8 (17:30~18:30)		
	学年	授業科目	担当教員	教室	授業科目	担当教員	教室	授業科目	担当教員	教室	授業科目	担当教員	教室	授業科目	担当教員	教室	授業科目	担当教員	教室	授業科目	担当教員	教室	授業科目	担当教員	教室
月	1																								
	2							プログラミング演習1	後藤祐介,他	プロ	プログラミング演習1	後藤祐介,他	プロ	プログラミング演習1	後藤祐介,他	プロ	プログラミング演習1	後藤祐介,他	プロ						
	3	知識工学	竹内孔一	11	知識工学	竹内孔一	11							プログラミング技法	乃村能成	10,11	プログラミング技法	乃村能成	10,11						
	4																								
火	1																								
	2	データ構造とアルゴリズム	山内利宏,他	11,14	データ構造とアルゴリズム	山内利宏,他	11,14																		
	3	情報工学実験A	相田敏明,他	プロ	情報工学実験A	相田敏明,他	プロ	情報工学実験A	相田敏明,他	プロ	情報工学実験A	相田敏明,他	プロ	人工知能	諸岡健一	11	人工知能	諸岡健一	11						
	4																								
水	1																								
	2							グラフ理論	神保秀司,他	11,14	グラフ理論	神保秀司,他	11,14												
	3	知識工学	竹内孔一	11	知識工学	竹内孔一	11	非手続き型言語	高橋規一,他	プロ	非手続き型言語	高橋規一,他	プロ												
	4																								
木	1																								
	2	プログラミング言語論	神保秀司	11	プログラミング言語論	神保秀司	11	データ構造とアルゴリズム	山内利宏,他	11,14	データ構造とアルゴリズム	山内利宏,他	11,14												
	3	情報工学実験A	相田敏明,他	プロ	情報工学実験A	相田敏明,他	プロ	情報工学実験A	相田敏明,他	プロ	情報工学実験A	相田敏明,他	プロ	プログラミング技法	乃村能成	10,11	プログラミング技法	乃村能成	10,11						
	4																								
金	1													コンピュータ科学基礎1	門田暁人,他	11,プロ	コンピュータ科学基礎1	門田暁人,他	11,プロ						
	2							グラフ理論	神保秀司,他	11,14	グラフ理論	神保秀司,他	11,14												
	3	人工知能	諸岡健一	11	人工知能	諸岡健一	11	非手続き型言語	高橋規一,他	プロ	非手続き型言語	高橋規一,他	プロ												
	4																								

図 2 作成した時間割（全 4 学期中第 1 学期）

表 5 岡山大学工学部情報系学科の時間割資源の種類とその数

	数	備考
時間枠	80	(学期 : 4) × (曜日 : 5) × (時限 : 4)
授業	43	(必修科目 : 29) + (選択科目 : 14)
教室	4	(講義室 : 3) + (実習室 : 1)
教員	23	

4. SAT ソルバによる求解時間

4.1 調査方針

提案する時間割作成システムでは，AUK によって記述された制約を満たす時間割を作成する手段として SAT ソルバを用いる．1 章で述べたように，システムのユースケースとして対話的な時間割作成が考えられるため，時間割作成にかかる時間はシステムの性能評価に大きく影響する．このため，本章では，実際の時間割編成問題の SAT ソルバによる求解時間を調査する．通常，SAT ソルバの入力は DIMACS フォーマット [5] で与えられる．DIMACS フォーマットとは，SAT Competition で提案された標準的な SAT ソルバの入出力フォーマットである．現在も多くの SAT ソルバが DIMACS フォーマットを入出力インタフェースに採用している．また，今回は時間割編成問題の例として，2020 年度の岡山大学工学部情報系学科の 1 年間の時間割を作成する．

4.2 充足可能性問題

充足可能性問題（Satisfiability problem; SAT）[6] とは，与えられた命題論理式を真にする変数割当てが存在するかを判定する問題である．命題論理式の全体を真にする割当てが存在するとき，その命題論理式は充足可能である．また，命題論理式の全体を真にする割当てが存在しないとき，その命題論理式は充足不能である．SAT は，通常，命題論理式を連言標準形（Conjunctive Normal Form; CNF）で扱う．CNF は，1 つ以上の節の連言で表される．節は，1 つ以上のリテラルの選言で表される．リテラルは，1 つの命題変数，または，その否定である．

SAT ソルバとは，SAT を高速に解くためのプログラムのことである．SAT ソルバは，与えられた命題論理式が充足可能か充足不能かを判定し，充足可能であれば，その変数割当てを出力する．

4.3 SAT への定式化

時間割編成問題を SAT ソルバを用いて解くためには，時間割編成問題を SAT へ変換する必要がある．ここで，各授業 l，各時間枠 t，各教室 r，各教員 i に対して，命題変数 P_{ltri} を導入する．命題変数 P_{ltri} は P_{ltri} が真のときに限って，授業 l は時間枠 t に教室 r で教員 i によって行われることを表す．

導入された命題変数のうち，ドメイン制約によって定められた無効な命題変数を除去し，競合制約を論理式に変換

©2021 Information Processing Society of Japan

表 6　実験環境

CPU	Intel(R) Xeon(R) Gold 6142 @ 2.60GHz
GPU	Quadro P4000
メモリ	48GB
SAT ソルバ	MiniSat[7]

する．

4.4　求解時間

3.5.3 項で示した条件のもと，時間割編成問題を CNF-SAT に定式化した．生成された命題変数の数は，最大で 316,480 個（時間割資源の組合せの総数; $80 \times 43 \times 4 \times 23$）であるが，最終的には，ドメイン制約によって，24,640 個となった．また，節の数は，19,866 個であった．定式化した CNF を**表 6** に示した実行結果で求解したところ，26.6 ms で解を得られた．SAT は NP 完全であるため計算量が爆発的に増加する可能性があるが，岡山大学工学部情報系学科の時間割編成問題においては，SAT ソルバを用いて短時間で求解可能であることが分かった．したがって，時間割編成問題の求解アプローチとして SAT ソルバを用いることで，高速な求解が期待できる．

最後に，得られた解を基にして作成した学生視点の時間割表を**図 2** に示す．図 2 の時間割表では，同じ時間枠内に 2 年次の必修授業と 3 年次の必修授業が同時に開講されている．たとえば，木曜日 1〜4 限に 2 年次の授業として開講される「プログラミング言語論」と「データ構造とアルゴリズム」，同時間に 3 年次の授業として開講される「情報工学実験 A」はいずれも必修授業である．このとき，2 年次の必修授業の単位を取得できなかった場合に 3 年次で再履修できないという問題が発生する．このため，必修授業は同じ時間に開講されないという制約（`NotOverlap` 制約）を考慮する制約に追加したいという要求が発生する．考慮する制約に，「必修授業における `NotOverlap` 制約」を追加して求解した結果，約 300 ms でこれを充足する解は存在しないことが分かった．つまり，岡山大学工学部情報系学科の時間割では，学年が異なる必修授業が同じ時間に開講されないような時間割を編成できないことが分かった．

このように短時間での求解が可能になると，制約を変更しながらインタラクティブな時間割作成が可能になる．

5.　時間割作成システム

本稿で提案する時間割作成システムは，AUK で記述された時間割資源，および時間割制約を DIMACS フォーマットに変換し，SAT ソルバを用いて求解することで時間割資源の自動割当てを行うシステムである．このシステムは，ユーザが必要な入力情報を AUK で記述することを想定している．

提案する時間割作成システムの構成要素とデータ流れを**図 3** に示す．図 3 を中心に，実際の時間割作成の処理流れを以下で説明する．

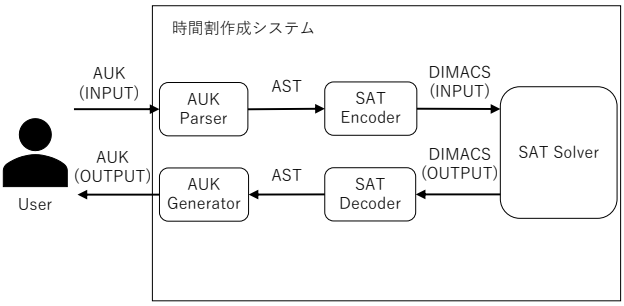

図 3　時間割作成システムの構成要素とデータ流れ

(1) ユーザは時間割資源，および時間割制約を AUK で記述し，時間割作成システムに入力する．

(2) 時間割作成システム（AUK Parser）は，AUK を解析し，AST（Abstract Syntax Tree）を生成する．ここで，AST とは，AUK と DIMACS フォーマットの中間表現である．

(3) 時間割作成システム（SAT Encoder）は，AST から SAT 符号化を行い，DIMACS フォーマットを生成する．

(4) SAT ソルバは，与えられた SAT の充足可能性判定を行い，結果を DIMACS フォーマットで生成する．

(5) 時間割作成システム（SAT Decoder）は，DIMACS フォーマットを解析し，AST を生成する．

(6) 時間割作成システム（AUK Generator）は，AST から AUK を生成する．

これまでに提案されている時間割作成システムと本システムの違いは，主に以下の 2 つである．

(1) 汎用性を指向した DSL を用いている点
本システムは，特定の大学の時間割に特化した入力ではなく，汎用性を指向した AUK を用いた入力を想定している．したがって，他のシステムと比較して，より汎用的であることが期待できる．

(2) 時間割編成問題を DIMACS フォーマットに変換している点
本システムが持つ能力は，AUK で記述された時間割編成問題を DIMACS フォーマット（CNF-SAT）に変換することである．本稿では，MiniSat を用いて時間割編成問題の求解を試みたが，システムそのものは特定の求解アルゴリズムに依存しない．

©2021 Information Processing Society of Japan

6. 関連研究

時間割作成システムの提案として，汎用プロジェクトスケジューラ SEES を大学の時間割作成に適用した例がある [1][2]．文献 [1] では，SEES を用いて約 0.37 秒で制約を満たす許容解を得られたことを示すとともに，時間割作成システムのインタフェースの重要性，困難性について述べられている．また，文献 [2] では，時間割作成システムを用いて作成した時間割が実際の時間割作成に利用された例を紹介している．

時間割編成問題の求解アルゴリズムとしては，最適化問題のメタヒューリスティクスアルゴリズムが適用される例がある [8]．文献 [8] では，時間割編成問題を例に組合せ最適化問題のメタヒューリスティクスの比較を行い，著者らが開発した大学時間割編成問題に特化したアルゴリズムである MAX-MIN Ant System が最も良い性能を示した．ITC2019 の議論においても，複雑な時間割作成にメタヒューリスティクスを用いることが有用であることが分かっている．

7. おわりに

本稿では，大学の時間割作成システムを提案し，システム実現のために 2 つの課題解決に取り組んだ．1 つ目は，時間割編成問題を記述可能な DSL である AUK を提案した．評価の結果，AUK は人が直接記述する用途として十分に耐えうることが分かった．2 つ目は，SAT ソルバを用いた時間割編成問題の求解時間の調査を行った．結果として，実際の時間割を 26.6 ms で求解できた．また，充足する解が存在しないような制約を追加した場合でも，300 ms で結果が求まった．

本稿では，時間割編成問題を決定問題として定式化したが，時間割編成問題は，しばしば最適化問題として定式化されている [9][10]．これは，決定問題では必ず満たさなければならない制約しか表現できないが，最適化問題では，これに加えてできるだけ満たしたい制約を表現できるからである．また，時間割編成問題を決定問題として扱う場合も，制約条件の緩和 [11] 等の工夫が必要である．そこで，今後の課題として，SAT ソルバの代わりに MaxSAT ソルバを用いることを検討する．MaxSAT とは，制約に重みをつけることで SAT を最適化問題に拡張したものであり，MaxSAT ソルバの多くは SAT ソルバと同様に DIMACS を入出力フォーマットとしている．このため，MaxSAT ソルバは，SAT ソルバと容易に差換え可能である．

参考文献

[1] 堀尾正典：汎用プロジェクトスケジューラの大学時間割問題への適用，名古屋学芸大学教養・研究紀要， Vol. 4, pp. 61–74 (2008).

[2] 堀尾正典：実用的な大学時間割作成システムの開発，日本経営工学会論文誌，Vol. 62, No. 1, pp. 1–11 (2011).

[3] 中央教育審議会：2040 年に向けた高等教育のグランドデザイン (答申)，文部科学省 (2018).

[4] Müller, T., Rudová, H. and Müllerová, Z.: ITC2019: International Timetabling Competition 2019, (online), available from ⟨https://www.itc2019.org⟩ (accessed 2021-01-13).

[5] committee, S. O.: SAT Competitions, SAT Competition (online), available from ⟨http://www.satcompetition.org/⟩ (accessed 2021-07-09).

[6] 宋　剛秀，番原睦則，田村直之，鍋島英知：SAT ソルバーの最新動向と利用技術，コンピュータソフトウェア，Vol. 35, No. 4, pp. 72–92 (2018).

[7] Niklas Een, N. S.: MiniSat Page, Chalmers University (online), available from ⟨minisat.se⟩ (accessed 2020-08-27).

[8] Socha, K., Knowles, J. and Sampels, M.: A MAX-MIN Ant System for the University Course Timetabling Problem, pp. 1–13 (2002).

[9] 池上敦子，呉　偉：学校時間割作成，日本オペレーションズ・リサーチ学会 (2020).

[10] 佐古田淳史，宋　剛秀，番原睦則，田村直之：登録後コース時間割問題の基数制約を用いた制約モデルと SAT ソルバーを用いた解法，人工知能学会 (2014).

[11] 柿本陽平，高橋弘毅，島川陽一：制約条件が厳しい時間割編成問題における制約条件の緩和とその評価 (2016).

©2021 Information Processing Society of Japan