車両装備仕様問題に対する解集合プログラミングの適用

竹内 頼人　田村 直之　番原 睦則

車両装備仕様とは，簡単に言うと，自動車のカタログに記載されているモデル／グレードと装備の組合せのことである．車両装備仕様問題は，与えられた装備仕様の個数，装備タイプの集合，装備オプションの集合などから，装備および燃費に関する制約を満たしつつ，予想販売台数を最大化する車両装備仕様を求める問題である．本論文では，企業平均燃費 (Corporate Average Fuel Efficiency; CAFE) 方式と呼ばれる燃費規制に基づく車両装備仕様問題に対して，解集合プログラミング (Answer Set Programing; ASP) を用いたソルバーの実装と評価について述べる．提案手法は，まず与えられた問題インスタンスを ASP のファクト形式に変換した後，それらファクトと車両装備仕様問題を解くための ASP 符号化と結合した上で，高速 ASP システムを用いて解を求める．ASP 符号化として，基本符号化と改良符号化の 2 種類を考案した．特に，改良符号化は，装備仕様の燃費や予想販売台数を算出するために必要な ASP のルール数を少なく抑えるよう工夫されており，大規模な問題への有効性が期待できる．また，製造ラインの削減や大量生産を促進することを狙いとして，予想販売台数の最大化に加えて，装備オプション数の最小化も行える拡張符号化についても触れる．最後に，提案手法の評価として，企業から提供を受けた実用規模の問題インスタンスを含む問題集を用いた実験結果について述べる．

1　はじめに

車両装備仕様とは，簡単に言うと，自動車のカタログに記載されているモデル／グレードと装備の組合せのことである．車両装備仕様を決めるには，販売される国や地域の法規や規制，地域や市場の特性，市場の嗜好や競合など十分に考慮する必要があり，現状では専門知識をもつ技術者の多大な労力が費やされている．そのため，車両装備仕様探索の自動化・効率化は自動車メーカーにとって重要な課題の一つである．

車両装備仕様問題は組合せ最適化問題の一種であり，主に装備タイプと装備オプションから構成される．**装備タイプ**はエンジンやトランスミッションなどの装備の種類を表す．**装備オプション**は 4 気筒エンジン，CVT などの具体的な装備を表す．車両装備仕様問題の目的は，与えられた装備仕様の個数，装備タイプの集合，装備オプションの集合から，装備および燃費に関する制約を満たしつつ，予想販売台数を最大化する車両装備仕様を求めることである．

本研究では，燃費に関する制約として，欧米で採用され日本でも 2020 年度から導入されている**企業平均燃費** (Corporate Average Fuel Efficiency; CAFE [5]) 方式を用いる．この CAFE 方式は自動車の燃費規制で，車種別ではなくメーカー全体での出荷台数を加味した平均燃費を算出し，規制をかける方式である．CAFE 方式の特長は，ある特定の車種では燃費基準を達成できなくても，他の車種の燃費を向上させることで基準を達成できることが可能な点である．本論文では，CAFE 方式に基づく車両装備仕様問題 (以降，**CAFE 問題**と呼ぶ) を対象とする．

解集合プログラミング (Answer Set Programing; ASP [1][3][4]) は，論理プログラミングから派生した宣言的プログラミングパラダイムである．ASP 言語

Applying Answer Set Programming to Vehicle Equipment Specification Problem

Raito Takeuchi, Mutsunori Banbara, 名古屋大学 大学院情報学研究科, Graduate School of Informatics, Nagoya University.

Naoyuki Tamura, 神戸大学 情報基盤センター, Information Science and Technology Center, Kobe University.

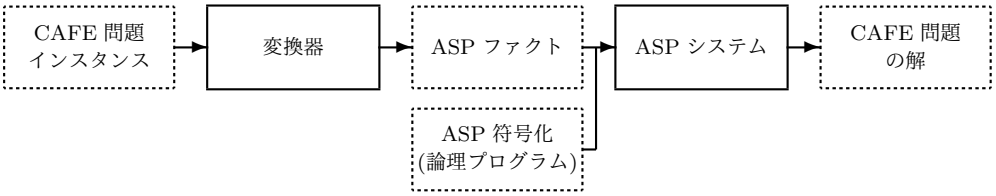

図 1　提案する CAFE 問題ソルバーの構成

は一階論理に基づく知識表現言語の一種である．論理プログラムは ASP のルールの有限集合である．ASP システムは論理プログラムから安定モデル意味論 [3] に基づく解集合を計算するシステムである．近年，SAT ソルバー技術を応用した高速 ASP システムが実現され，ロボット工学，システム生物学，システム検証，制約充足問題，プランニングなど様々な分野への実用的応用が急速に拡大している [2]．

本論文では，ASP を用いた CAFE 問題ソルバーの実装と評価について述べる．提案ソルバーは，まず与えられた問題インスタンスを ASP のファクト形式に変換した後，それらファクトと CAFE 問題を解くための ASP 符号化と結合した上で，高速 ASP システムを用いて解を求める (図 1 参照)．

ASP 符号化として，基本符号化と改良符号化の 2 種類を考案した．基本符号化は，CAFE 問題の制約を ASP のルール 17 個で簡潔に表現できる．改良符号化は，装備仕様の燃費や予想販売台数を算出するために必要な IWR (Inertial Working Rating) 値の総和の上下限を厳密に計算することにより，基礎化後のルール数を少なく抑えるよう工夫されている．これにより，改良符号化は大規模な CAFE 問題への有効性が期待できる．また，製造ラインの削減や大量生産を促進することを狙いとして，予想販売台数の最大化に加えて，装備オプション数の最小化も行える拡張符号化についても触れる．

提案手法の有効性を評価するために，企業から提供を受けた小規模・実用規模・大規模の CAFE 問題に対して実行実験を行った結果，基本符号化と改良符号化はどちらも小規模な問題の最適解を求めることができた．また，実用規模およびより大規模な問題に対しては，改良符号化が基本符号化より優れた結果を示し，その優位性が確認できた．

本論文の貢献は，CAFE 方式と呼ばれる新しい燃費規制に基づく車両装備仕様問題に対して，解集合プログラミング技術をはじめて適用し，実用規模の問題を用いた実験結果を通じて，その有用性を確認したことである．

2　CAFE 問題

CAFE 問題の入力は以下の通りである．以降，装備タイプをタイプ，装備オプションをオプションと簡単に書くことにする．

1. タイプの集合
2. オプションの集合
3. タイプとオプションの対応関係
4. 各タイプで選択可能なオプション数の上下限値
5. タイプ同士，オプション同士，および，タイプとオプション間の依存関係
6. 各オプションに付加された IWR 値
7. 求めたい装備仕様の個数
8. 各装備仕様とタイプ (あるいはオプション) 間の依存関係
9. 各装備仕様に含まれるオプションの IWR 値の総和と燃費との対応表
10. 各装備仕様に含まれるオプションの IWR 値の総和と予想販売台数との対応表
11. CAFE 基準値

入力 6 の IWR は Inertial Working Rating の略で，直観的には各オプションの重量を表す．入力 7 の個数は，求めたい派生車両の数と考えるとわかりやすい．CAFE 問題は，上記の入力から，装備および燃費に関する制約を満たしつつ，予想販売台数を最大化する車両装備仕様を求める問題である．

CAFE 問題の制約は以下の通りである．

範囲制約 ：各装備仕様について，各タイプで選択

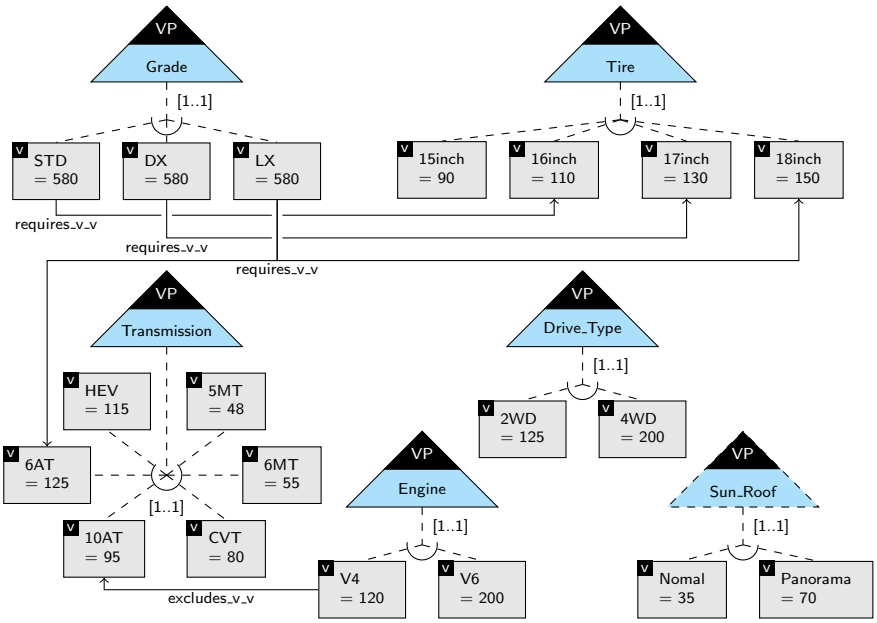

図 2　CAFE 問題の例

されるオプション数は，入力 4 で与えられた上下限値の範囲内でなければならない．

依存制約 ：各装備仕様について，入力 5 で与えられた依存関係を満たさなければならない．依存制約には，要求制約と排他制約の 2 つがある．

燃費制約 ：入力 11 の CAFE 基準値を t，入力 7 の装備仕様個数を n として，以下の CAFE 規制を満たさなければならない．

$$\frac{\sum_{i=1}^{n} FE_i \cdot SV_i}{\sum_{i=1}^{n} SV_i} \quad \geq \quad t$$

不等式の左辺は n 個の装備仕様の**平均燃費**を表している．FE_i と SV_i は，装備仕様 i の燃費と予想販売台数を表しており，それぞれ，入力 9 と 10 の対応表を元に計算される．

初期制約 ：入力 8 で与えられた依存関係を満たさなければならない．

CAFE 問題の例を図 2 に示す．この例は，ソフトウェアプロダクトライン開発の分野で用いられる**可変性モデル** (Orthogonal Variability Model; OVM [6]) によって記述されている．図 3 に，可変性モデルの基本的な表記法を示す．可変性モデルでは，仕様ごとに変わりうる項目を**可変点**と呼び，三角形で表す．可変点の具体的なインスタンスを**バリアント**と呼び，長方形で表す．可変点とバリアントの対応関係には**変異依存性**と**選択肢**があり，選択肢の場合は，その多重度も付記される．可変点同士，バリアント同士，および，可変点とバリアント間の依存関係は，**制約依存性**によって表される．制約依存性には，要求 (requires) と排除 (excludes) の 2 種類がある．

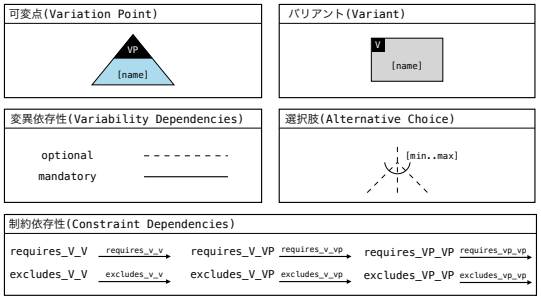

図 3　可変性モデルの表記法 [6]

表 1　CAFE 問題 (図 2) の解

装備仕様		1	2	3
装備	Grade	STD	DX	LX
	Drive_Type	2WD	2WD	2WD
	Engine	V4	V6	V6
	Tire	16inch	17inch	18inch
	Transmission	5MT	6MT	10AT
	Sun_Roof	-	Normal	-
IWR 値の総和		983	1,125	1,180
燃費 (km/L)		10.2	8.9	8.5
予想販売台数		745	1,988	1,171
平均燃費 (km/L)		9.0		
予想販売台数 (合計)		3,904		

可変性モデルで CAFE 問題を記述する場合，タイプは可変点，オプションとその IWR 値はバリアント，タイプとオプションの対応関係および選択可能なオプション数の上下限値は選択肢，タイプ同士，オプション同士，および，タイプとオプション間の依存関係は制約依存性によって表される．以上から，CAFE 問題の入力のうち，1〜6 は可変性モデルによって記述されることがわかる．

図 2 の問題例は，6 個のタイプ，19 個のオプション，5 個の依存制約から構成され，各タイプの選択可能なオプション数はすべて 1 である．本論文では，可変点で表されるタイプは，各装備仕様に対して必須とする．ただし，Sun_Roof のような選択可能なタイプ (必須ではないタイプ) については，破線の可変点で表すものとする．

図 2 の問題に対する解の例を表 1 に示す．この解は，CAFE 基準値に 9.0km/L，求めたい装備仕様の個数に 3 を与え，装備仕様とオプションの依存関係として，(装備仕様 1, STD), (装備仕様 2, DX), (装備仕様 3, LX) を要求して得られたものである．各装備仕様の燃費 (km/L) は，左から順に 10.2, 8.9, 8.5 と個々には CAFE 基準値を満たしていないが，3 台の平均燃費は 9.028km/L となり，CAFE 規制を満たしている．

3　解集合プログラミング

ASP の言語は，一般拡張選言プログラム (General Extended Disjunctive Program) をベースとしている [4]．本節では，説明の簡略化のため，そのサブクラスである標準論理プログラムについて説明する．以降，標準論理プログラムを単に論理プログラムと呼ぶ．

論理プログラムは，以下の形式の**ルール**の有限集合である．

$$a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \qquad (1)$$

このルールの直観的な意味は，「$a_1, \ldots, a_m$ がすべて成り立ち，$a_{m+1}, \ldots, a_n$ のそれぞれが成り立たないならば，a_0 が成り立つ」である．ここで，$0 \leq m \leq n$ であり，各 a_i はアトム，$\sim$ は**デフォルトの否定** †1，“,” は連言を表す．$\leftarrow$ の左側を**ヘッド**，右側を**ボディ**と呼ぶ．ボディが空のルール (すなわち $a_0 \leftarrow$) を**ファクト**と呼び，$\leftarrow$ を省略してよい．

ヘッドが空のルールを**一貫性制約**と呼び，以下のように表す．

$$\leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \qquad (2)$$

例えば，一貫性制約 “$\leftarrow a_1, a_2$” は，「a_1 と a_2 が両方同時に成り立つことはない」を意味し，“$\leftarrow a_1, \sim a_2$” は，「a_1 が成り立つならば，a_2 が成り立つ」を意味する．

ASP 言語には，組合せ問題を簡潔に記述するために，**アグリゲート** (aggregate) と呼ばれる拡張構文がいくつか用意されている．例えば，**選択子** “$\{a_1; \ldots; a_n\}$.” は，アトム集合 $\{a_1, \ldots, a_n\}$ の任意の部分集合を解集合に含めることを意味する．**個数制約**は選択子の両端に選択可能な個数の上下限を付けたものである．例えば，“$lb\ \{a_1; \ldots; a_n\}\ ub \leftarrow Body$” と書くと，「$Body$ が成り立つならば，$a_1, \ldots, a_n$ のうち，lb 個以上 ub 個以下が成り立つ」を意味する．**重み付き個数制約** “$t = \#sum\ \{w_1 : a_1; \ldots; w_n : a_n\}$.” は，$a_1, \ldots, a_n$ のうち真となるアトムの重み和が項 t に等しくなることを意味する．項 w_i は重みを表し，演算子としては “=” 以外にも “$\leq$”,“$\geq$” などを使用できる．さらに，重み付き個数制約の “$\#sum$” を，“$\#max$” や

†1　**失敗による否定**とも呼ばれる．述語論理で定義される否定 ($\neg$) とは意味が異なる．

表 2　論理プログラムとソースコードの対応

論理プログラム	←	,	;	∼	$\#sum$
ソースコード	:-	,	;	not	#sum

“$\#min$” に書き換えると，重み和ではなく，真となるアトムの重みの最大値や最小値を求めることができる．また，組合せ最適化問題を解くために，最小化関数 (`#minimize`)・最大化関数 (`#maximize`) 等も用意されている．

近年，*clingo* †2，*DLV* †3，*WASP* †4 など，SAT ソルバー技術を応用した高速な ASP システムが開発されている．なかでも *clingo* は，高性能かつ高機能な ASP システムとして世界中で広く使われている．これらの高速 ASP システムは，変数を含む論理プログラムを変数を含まない論理プログラムに変換 (**基礎化**) したのち，ASP ソルバーを用いて解集合を計算する．本論文で使用する ASP システム *clingo* は，基礎化のためのグラウンダー *gringo* と ASP ソルバー *clasp* をシームレスに結合したものである．

以降の節で示す論理プログラムのソースコードはすべて *gringo* 言語で書かれており，表記上の対応については表 2 の通りである．

4　ASP に基づく CAFE 問題ソルバー

提案する CAFE 問題ソルバーは，与えられた問題インスタンスを ASP のファクト形式に変換した後，それらファクトと CAFE 問題を解くための ASP 符号化 (論理プログラム) を結合した上で，高速 ASP システム *clingo* を用いて解を求める (図 1 参照)．本節では，CAFE 問題の入力と制約を論理プログラムとして表現する方法について述べる．本論文では，説明の簡単化のため，各タイプが選択可能なオプション数の上下限値を 1 とする．

4.1　ASP ファクト形式

CAFE 問題の入力は，CAFE 基準値 (入力 11) を除いて，ASP のファクトで表される．CAFE 基準値

```
vp_def("Drive_Type").
v_def("2WD", "Drive_Type", 125).
v_def("4WD", "Drive_Type", 200).

vp_def("Engine").
v_def("V4", "Engine", 120).
v_def("V6", "Engine", 200).

vp_def("Grade").
v_def("STD", "Grade", 580).
v_def("DX",  "Grade", 580).
v_def("LX",  "Grade", 580).

vp_def("Sun_Roof").
v_def("Nomal",    "Sun_Roof", 35).
v_def("Pnrorama", "Sun_Roof", 70).

vp_def("Tire").
v_def("15_inch_Tire", "Tire",  90).
v_def("16_inch_Tire", "Tire", 110).
v_def("17_inch_Tire", "Tire", 130).
v_def("18_inch_Tire", "Tire", 150).

vp_def("Transmission").
v_def("6AT",  "Transmission", 115).
v_def("10AT", "Transmission", 125).
v_def("HEV",  "Transmission",  95).
v_def("CVT",  "Transmission",  80).
v_def("5MT",  "Transmission",  48).
v_def("6MT",  "Transmission",  55).

require_v_v("STD", "16_inch_Tire").
require_v_v("DX",  "17_inch_Tire").
require_v_v("LX",  "18_inch_Tire").
require_v_v("LX",  "10AT").
exclude_v_v("V4",  "10AT").

require_vp("Drive_Type").
require_vp("Engine").
require_vp("Grade").
require_vp("Tire").
require_vp("Transmission").

group(1). group(2). group(3).

require_g_v(1, "STD").
require_g_v(2, "DX").
require_g_v(3, "LX").
```

コード 1　CAFE 問題の例 (図 2) のファクト表現．装備仕様の個数:3，装備仕様とオプションの依存関係: (装備仕様 1, STD), (装備仕様 2, DX), (装備仕様 3, LX)

†2 https://potassco.org/
†3 http://www.dlvsystem.com/dlv/
†4 https://www.mat.unical.it/ricca/wasp/

は ASP の定数 t で表すものとし，実行時に *clingo* のオプションから値を指定する．

CAFE 問題の例 (図 2) をファクトで表現したものをコード 1 に示す．この 2 つを見比べると，可変性モデルの可変点，バリアント，制約依存性 (要求)，制約依存性 (排除) が，それぞれファクト vp_def/1, v_def/3, require_v_v/2, exclude_v_v/2 に対応していることがわかる．例えば，バリアント V4 は，ファクト v_def("V4", "Engine", 120). に対応している．可変性モデルは CAFE 問題の入力 1〜6 を含んでいる．この他，require_vp/1 は必須タイプ，group/1 は装備仕様の識別子 (入力 7)，require_g_v/2 は各装備仕様とオプション間の依存関係を表している (入力 8)．例えば，require_g_v(1, "STD"). は，装備仕様 1 がオプション STD を要求することを表す．

IWR 値の総和と燃費の対応表 (入力 9) は，ファクト fe_map(S,FE) の有限集合で表される．ここで，S は IWR 値の総和，FE は燃費を表す．同様にして，IWR 値の総和と予想販売台数の対応表 (入力 10) は，ファクト sv_map(S,SV) の有限集合で表される．

4.2 CAFE 問題の ASP 符号化

CAFE 問題の各制約は，ASP の個数制約および一貫性制約を使って簡潔に表現される．CAFE 問題の制約と目的関数を表す論理プログラムをコード 2 に示す．装備仕様 G がタイプ VP を装備することを意味するアトム vp(VP,G) を導入する．また，装備仕様 G がオプション V を実装することを意味するアトム v(V,G) も併せて導入する．

1 行目のルールは，各装備仕様 G，各タイプ VP に対して，vp(VP,G) の候補を生成する．2 行目のルールは，各装備仕様 G に対して，VP が必須タイプならば，G は VP を装備しなければならないことを表す．

範囲制約は 5 行目のルールで表される．このルールは，タイプ VP を装備する装備仕様 G に対して，VP のオプションからちょうど 1 個を実装することを表す．

燃費制約は 8〜13 行目のルールで表される．8 行目の iwr(S,G) は，装備仕様 G の IWR 値の総和が S であることを表す．11〜12 行目の fe(FE,G) と sv(SV,G) は，それぞれ，装備仕様 G の燃費 FE と予想販売台数 SV を表す．13 行目のルールは，第 2 節で示した CAFE 規制の式を以下のように変形し，ASP の重み付き個数制約で表している．

$$\sum_{i=1}^{n}(FE_i - t) \cdot SV_i \geq 0$$

要求制約は 16〜19 行目のルールで表される．例えば，16 行目のルールは，装備仕様 G がオプション V1 を実装し，かつ，V1 が V2 を要求するならば，G は V2 を実装しなければならないことを表す．他の要求制約，排除制約，初期制約も同様に，一貫性制約を用いて簡潔に表現できる．

CAFE 問題の目的は，予想販売台数を最大化することである．この目的関数は，31 行目の最大化関数 #maximize によって表される．本節で述べた論理プログラム (コード 2) を基本符号化と呼ぶ．

```
1  { vp(VP,G) } :- vp_def(VP), group(G).
2  :- not vp(VP,G), require_vp(VP), group(G).
3
4  % 範囲制約
5  1 { v(V,G) : v_def(V,VP,_) } 1 :- vp(VP,G).
6
7  % 燃費制約
8  iwr(S,G) :-
9      S = #sum { IWR,V : v(V,G), v_def(V,_,IWR) },
10     group(G).
11 fe(FE,G) :- iwr(S,G), fe_map(S,FE).
12 sv(SV,G) :- iwr(S,G), sv_map(S,SV).
13 :- not 0 #sum { (FE-t)*SV,FE,SV,G : fe(FE,G), sv(SV,G) }.
14
15 % 要求制約
16 :- require_v_v(V1,V2), v(V1,G), not v(V2,G).
17 :- require_v_vp(V,VP), v(V,G), not vp(VP,G).
18 :- require_vp_v(VP,V), vp(VP,G), not v(V,G).
19 :- require_vp_vp(VP1,VP2), vp(VP1,G), not vp(VP2,G).
20
21 % 排除制約
22 :- exclude_v_v(V1,V2), v(V1,G), v(V2,G).
23 :- exclude_v_vp(V,VP), v(V,G), vp(VP,G).
24 :- exclude_vp_v(VP,V), vp(VP,G), v(V,G).
25 :- exclude_vp_vp(VP1,VP2), vp(VP1,G), vp(VP2,G).
26
27 % 初期制約
28 :- not v(V,G), require_g_v(G,V).
29
30 % 目的関数
31 #maximize { SV,G : sv(SV,G) }.
```

コード 2　基本符号化

```
ub_vp(UB,VP) :-
    UB = #max { IWR,V : v_def(V,VP,IWR) },
    vp_def(VP).
lb_vp(LB,VP) :-
    LB = #min { IWR,V : v_def(V,VP,IWR) },
    vp_def(VP).

ub_iwr(S) :-
    S = #sum { UB,VP : ub_vp(UB,VP) }.
lb_iwr(S) :-
    S = #sum { LB,VP : lb_vp(LB,VP), require_vp(VP) }.

iwr(S,G) :-
    S = #sum { IWR,V : v(V,G), v_def(V,_,IWR) },
    LB <= S, S <= UB, lb_iwr(LB), ub_iwr(UB),
    group(G).
```

コード 3　基本符号化 (コード 2 の 8〜10 行目) の改良

4.3　符号化の改良

前節で述べた基本符号化は CAFE 問題の制約を簡潔に表現できるが，改良できる点も残っている．

タイプの集合を VP，必須タイプの集合を VP^*，オプションの集合を V，タイプ $i \in VP$ が実装可能なオプションの集合を V_i，オプション $j \in V$ の IWR 値を w_j とする．このとき，基本符号化 (コード 2) の 8〜10 行目のルールで生成されるアトム `iwr(S,G)` について，IWR 値の総和を表す `S` の上下限は，

$$0 \leq \mathtt{S} \leq \sum_{j \in V} w_j$$

である．しかし，これは自明な上下限であり，各タイプが選択可能なオプション数の上下限値，および，必須かどうかの別を考慮することで，

$$\sum_{i \in VP^*} \min_{j \in V_i} w_j \leq \mathtt{S} \leq \sum_{i \in VP} \max_{j \in V_i} w_j$$

のように `S` の上下限をより厳密に計算することができる．下限値は，各必須タイプが選択可能なオプションの IWR 値の最小値の総和である．上限値は，各タイプが選択可能なオプションの IWR 値の最大値の総和である．このアイデアを元にした論理プログラムをコード 3 に示す．基本符号化 (コード 2) の 8〜10 行目を，コード 3 で置き換えた符号化を，改良符号化と呼ぶ．改良符号化は，基本符号化と比較して，`iwr(S,G)` に関する基礎化後のルール数を少なく抑えることができる．そのため，大規模な問題への有効性が期待できる．

```
1  % 予想販売台数の最大化
2  #maximize { SV@2,G : sv(SV,G) }.
3
4  % オプション数の最小化
5  used_v(V) :- v(V,G).
6  #minimize { 1@1,V : used_v(V) }.
```

コード 4　オプション数の最小化

4.4　符号化の拡張

製造ラインの削減や大量生産を促進することを狙いとして，予想販売台数の最大化に加えて，オプション数の最小化も可能なように拡張する．コード 4 に拡張のための論理プログラムを示す．

予想販売台数の最大化 (2 行目) ついては，これまでと同じである．5 行目の `used_v(V)` は，オプション `V` がいずれかの装備仕様において実装されたことを意味する．オプション数の最小化は，`used_v(V)` が成り立つ `V` の数を最小化することにより，容易に実現される (6 行目)．目的関数中の`@`の右側はその優先度を表しており，今回の拡張例では，予想販売台数の最大化 (優先度 2)，オプション数の最小化 (優先度 1) の順に最適化される．改良符号化の目的関数を，コード 4 で置き換えた符号化を拡張符号化と呼ぶ．

図 2 の CAFE 問題に対して，最適解の全列挙をした結果を表 3 に示す．CAFE 基準値が 8.5km/L のとき，予想販売台数が 5,525 台である最適解が，改良符号化では (解 1, 解 2, 解 3) の 3 つ存在した．一方で，オプション数の最小化を加えた拡張符号化では，最適解は (解 3) のただ 1 つだけとなる．このように，ASP に基づく CAFE 問題ソルバーでは，ユーザの選好にあわせて目的関数を柔軟に追加することにより，複数ある最適解に対して優劣をつけることができる．

5　実行実験

提案符号化の有効性を評価するために，開発した CAFE 問題ソルバーの実行実験を行った．ベンチマーク問題 (表 4 参照) には，企業から提供を受けた小規模 (small)・実用規模 (medium)・大規模 (big) の CAFE 問題 (3 問) に対して，5 種類の CAFE 基準値 (8.5, 9.0, 9.5, 10.0, 10.5km/L) を適用した問題イン

表 3　CAFE 問題 (図 2) の最適解全列挙．CAFE 基準値：8.5km/L

		解 1			解 2			**解 3**		
装備仕様		1	2	3	1	2	3	1	2	3
装備	Grade	STD	DX	LX	STD	DX	LX	STD	DX	LX
	Drive_Type	4WD	2WD	4WD	2WD	2WD	4WD	2WD	2WD	4WD
	Engine	V4	V6	V6	V6	V6	V6	V6	V6	V6
	Tire	16	17	18	16	17	18	16	17	18
	Transmission	5MT	HEV	10AT	CVT	HEV	10AT	6AT	HEV	10AT
	Sun_Roof	Panorama	-	-	Nomal	-	-	-	-	-
IWR 値の総和		1,128	1,130	1,255	1,130	1,130	1,255	1,130	1,130	1,255
燃費 (km/L)		8.9	8.8	8.0	8.8	8.8	8.0	8.8	8.8	8.0
予想販売台数		2,007	2,007	1,511	2,007	2,007	1,511	2,007	2,007	1,511
平均燃費 (km/L)		8.6			8.5			8.5		
予想販売台数 (合計)		5,525			5,525			5,525		
オプション数		14			13			12		

表 4　ベンチマーク問題

問題名	タイプ数	オプション数	要求制約数
small	8	21	4
medium	86	229	147
big	315	1,337	0

スタンス (合計 15 問) を使用した．求めたい装備仕様の個数は 3 とした．

ASP システムには，広く普及している高速 ASP システム *clingo*-5.4.0 を使用し，一問当たりの時間制限は 7,200 秒とした．実験環境は，Mac mini (3.2GHz Intel Core i7, 64GB メモリ) である．

表 5 に実験結果を示す．左から順に，問題名，CAFE 基準値，基本符号化と改良符号化の予想販売台数 (目的関数の値) となっている．記号 '∗' 付きの値は最適値であることを表す．各問題インスタンスごとに，より良い予想販売台数をボールド体で示している．また，UNSAT は制約を満たす実行可能解が存在しないことを意味し，UNKNOWN は制限時間内に実行可能解が一つも求められなかったことを意味する．

最適値・最良値を求めた問題数は，基本符号化が 6 問，改良符号化が 13 問であり，改良符号化がより多くの問題に対して優れた結果を示している．特に，大規模インスタンス big については，5 問すべてに対して，改良符号化が基本符号化よりも優れた解を得ることに成功している．これにより，改良符号化の大規模な問題への有効性が確認できた．小規模なインスタンス small については，基本符号化と改良符号化ともに 5 問すべてに対して最適解 (あるいは UNSAT) を求めることができた．

インスタンス small について，それぞれの符号化で最適解，あるいは UNSAT を得るまでに要した CPU 時間を表 6 に示す．各問題インスタンスごとに，より短い CPU 時間をボールド体で示している．改良符号化は，基本符号化と比較して 5 問中 4 問で優れた結果を示している．また，CPU 時間の平均からも改良符号化の方がより高速に解を求めていることがわかる．

次に，基本符号化と改良符号化とで，基礎化後のルール数を比較した結果を表 7 に示す．括弧内の数字は，基本符号化のルール数を 1 としたときの比率である．表 7 より，改良符号化は，基本符号化と比較して，基礎化後のルール数を少なく抑えられていることがわかる．特に，big では，70%以上のルール数の削減に成功している．このルール数の削減が，改良符号

表 5　実験結果

問題名	CAFE	予想販売台数	
	基準値	基本符号化	改良符号化
small	8.5	**6,021***	**6,021***
	9.0	**5,007***	**5,007***
	9.5	**2,688***	**2,688***
	10.0	**1,318***	**1,318***
	10.5	UNSAT	UNSAT
medium	8.5	6,010	**6,021**
	9.0	**5,595**	**5,595**
	9.5	**3,447**	3,430
	10.0	2,245	**2,250**
	10.5	1,690	**1,845**
big	8.5	UNKNOWN	**3,877**
	9.0	1,038	**4,623**
	9.5	688	**3,121**
	10.0	1,634	**2,064**
	10.5	538	**904**
最適値・最良値の数		6	13

表 6　解を得るまでに要した CPU 時間 (秒)

問題名	CAFE	CPU 時間	
	基準値	基本符号化	改良符号化
small	8.5	37.868	**23.318**
	9.0	48.965	**43.362**
	9.5	**95.110**	173.172
	10.0	99.954	**0.343**
	10.5	439.613	**0.080**
平均		144.302	**48.055**

化の大規模問題に対する優位性につながっていると考えられる．

さらに，インスタンス small を用いて，求める装備仕様の個数を増やした予備的な実験を行った．装備仕様の個数は $n = 3, 6, 12$ とし，CAFE 基準値は前述の実験と同じ 5 種類を用いた．各装備仕様とオプション間の依存関係については，同じ装備仕様が生じないように与えた．制限時間は 1 問あたり 1 時間とした．$n = 3$ では 5 問すべて，$n = 6$ では 5 問中 4 問について、最適解あるいは UNSAT を求めることができた．$n = 12$ では 5 問中 3 問は UNSAT であった．残り 2 問については実行可能解は得られたが，その最適性を証明するには至らなかった．

表 7　比較結果：基礎化後のルール数．CAFE 基準値：9.0km/L

問題名	基本符号化	改良符号化
small	83,520 (1.00)	32,855 (0.39)
medium	93,017 (1.00)	56,940 (0.61)
big	155,654 (1.00)	42,190 (0.27)

6　おわりに

本論文では，CAFE 方式と呼ばれる新しい燃費規制に基づく車両装備仕様問題に対して，解集合プログラミング技術をはじめて適用し，その記述性の高さ，柔軟な最適解探索などの利点を確認した．また，効率性についても，企業から提供された問題集を用いた実験結果から，実用規模の問題に対して適用可能であることが確認できた．

装備の組合せに関する制約は，まず企画部門で設定され，そのあと開発部門，生産部門，販売部門に受け渡され，各部門で制約が追加されながら徐々に成熟していく．これら様々な制約を調査・整理・実装することにより，CAFE 問題ソルバーを拡張することが今後の課題である．

参 考 文 献

[1] Baral, C.: *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press, 2003.

[2] Erdem, E., Gelfond, M., and Leone, N.: Applications of ASP, *AI Magazine*, Vol. 37, No. 3(2016), pp. 53–68.

[3] Gelfond, M. and Lifschitz, V.: The Stable Model Semantics for Logic Programming, *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, MIT Press, 1988, pp. 1070–1080.

[4] 井上克巳, 坂間千秋: 論理プログラミングから解集合プログラミングへ, コンピュータソフトウェア, Vol. 25, No. 3(2008), pp. 20–32.

[5] 経済産業省, 国土交通省: 乗用車燃費規制の現状と論点について, 第 5 回自動車燃費基準小委員会, 2018.

[6] Pohl, K., Böckle, G., and van der Linden, F.: *Software Product Line Engineering*, Springer, 2005.