解集合プログラミングを用いたハミルトン閉路問題の解法に関する考察

平手 貴大　宋 剛秀　田村 直之　番原 睦則

本稿では，解集合プログラミング (ASP) を用いたハミルトン閉路問題，最短ハミルトン閉路問題，コスト制約付きハミルトン閉路問題の解法について述べる．ハミルトン閉路問題を解く ASP 符号化として，undirected，directed，acyclicity の 3 つを考案した．undirected は，ハミルトン閉路問題を次数制約と連結制約で簡潔に表現した符号化である．directed は，与えられた無向グラフの各辺に対して，2 つの弧を対応させることで有向グラフ化して解く符号化である．acyclicity は，directed 符号化をベースに，連結制約に代わる部分閉路禁止制約を，ASP の組込み非閉路制約で表現した符号化である．最短ハミルトン閉路問題とコスト制約付きハミルトン閉路問題については，考案した 3 つの符号化に目的関数とコスト制約をそれぞれ追加することで自然に拡張できる．考案した符号化の有効性を評価するために，Flinders Hamiltonian Cycle Project で公開されているハミルトン閉路問題のインスタンス (全 1001 問) を用いて実験を行った．その結果，directed 符号化が，他の符号化と比較して，より多くの問題を高速に解くことができ，その優位性が確認できた．

1　はじめに

ハミルトン閉路問題 (Hamiltonian Cycle Problem; HCP) は，与えられたグラフの全頂点をちょうど一度ずつ通る閉路が存在するかどうかを判定する問題である [6]．ハミルトン閉路問題は代表的な NP 完全問題である．**ハミルトン路問題** (Hamiltonian Path Problem; HPP) は，ハミルトン閉路問題から始点と終点が一致するという閉路の条件を取り除いたものである．これらの問題は，重要な工学的応用が数多く存在するため，古くから盛んに研究されている．例えば，数理最適化の分野で有名な巡回セールスマン問題は，グラフの辺に距離が付随しているとき，最短距離のハミルトン閉路を求める**最短ハミルトン閉路問題**と考えることができる．また，ごく最近では，距離の総和が所与の閾値以下 (または以上) であることを制約条件として付加した**コスト制約付きハミルトン閉路問題**の解を高速に全列挙するアルゴリズムも提案されている [7]．本研究では，これらの (無向グラフ上の) ハミルトン閉路問題およびその関連問題を対象とする．

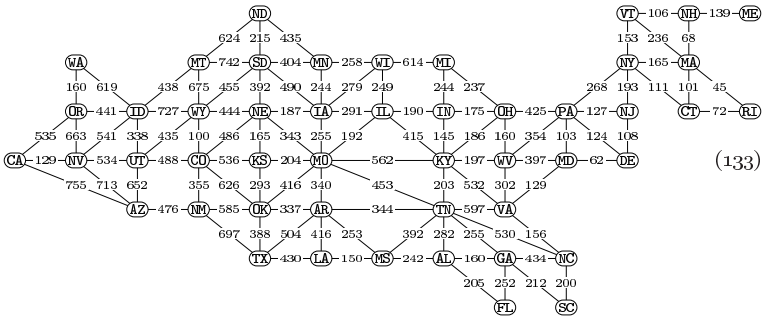
解集合プログラミング (Answer Set Programming; ASP [1] [4] [8] [11]) は，論理プログラミングから派生した宣言的問題解法のためのプログラミングパラダイムである．ASP 言語は，一階論理に基づく知識表現言語の一種である．論理プログラムは ASP のルールの有限集合である．ASP システムは論理プログラムから安定モデル意味論に基づく解集合を計算するシステムである．近年，SAT 技術を応用した高速 ASP システムが開発され，スケジューリング，プランニング，システム生物学，システム検証など様々な分野への実用的応用が急速に拡大している．ハミルトン閉路問題およびその関連問題に対して ASP を用いる利点としては，ASP 言語の高い表現力，高速な解列挙，組込み非閉路制約，充足不能コアに基づく最適化，インクリメンタル ASP 解法などが挙げられる．

本論文では，解集合プログラミング (ASP) を用い

A Study on Solving Hamiltonian Cycle Problem with Answer Set Programming

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

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



(133)

図 1　D. E. Knuth の教科書にある米国本土 48 州の隣接関係を表すグラフ

たハミルトン閉路問題，最短ハミルトン閉路問題，コスト制約付きハミルトン閉路問題の解法について述べる．ハミルトン閉路問題を解く ASP 符号化として，undirected，directed，acyclicity の 3 つを考案した．undirected は，ハミルトン閉路問題を次数制約と連結制約で簡潔に表現した符号化である．directed は，与えられた無向グラフの各辺 $u - v$ に対して，2 つの弧 $u \to v$ と $v \to u$ を対応させることで有向グラフ化して解く符号化である．acyclicity は，directed 符号化をベースに，連結制約に代わる部分閉路禁止制約を，ASP の組込み非閉路制約 [2] で表現した符号化である．最短ハミルトン閉路問題とコスト制約付きハミルトン閉路問題については，考案した 3 つの符号化に目的関数とコスト制約をそれぞれ追加することで自然に拡張できる．

考案した符号化の有効性を評価するために，Flinders Hamiltonian Cycle Project (FHCP) で公開されている HCP インスタンス (全 1001 問) [5] を用いて実験を行った．この問題集は FHCP Challenge と呼ばれる国際競技会で使用された頂点数が 66 個から 9,528 個の HCP インスタンスから構成されている．実験の結果，directed 符号化が 875 問と最も多くの問題を解き，他の符号化と比較してその優位性が確認できた．

2　ハミルトン閉路問題とその関連問題

グラフの全頂点を一度ずつ通る路は**ハミルトン路** (Hamiltonian Path) と呼ばれ，閉路は**ハミルトン閉路** (Hamiltonian Cycle) と呼ばれる．以下に，本稿で対象とする問題群について述べる．

- **ハミルトン閉路問題**は，与えられたグラフの全頂点をちょうど一度ずつ通る閉路が存在するかどうかを判定する問題である．
- **最短ハミルトン閉路問題**は，グラフの辺に距離が付随しているとき，最短距離のハミルトン閉路を求める最適化問題である．
- **コスト制約付きハミルトン閉路問題**は，ハミルトン閉路問題に，距離の総和が所与の閾値以下 (または以上) であることを制約条件として付加した問題である [7]．

これらの問題から，始点と終点が一致するという閉路の条件を取り除いた問題を，それぞれ，ハミルトン路問題，最短ハミルトン路問題，コスト制約付きハミルトン路問題と呼ぶ．

図 1 は，D. E .Knuth の The Art of Computer Programming [10] に記載されている米国本土 48 州の隣接関係を表すグラフである．この隣接グラフの頂点数は 48，辺の数は 105 である．各辺に付与されている数字は，州都の間の距離 (マイル) を表している．西海岸ワシントン州 (WA) から東海岸メーン州 (ME) までのハミルトン路は 6,876,928 通りあり，最短ハミルトン路は 11,698 マイルである．コスト制約付きハミルトン路問題は，WA から ME までの距離の総和がある閾値以下であるハミルトン路を求める問題と考えることができる．本稿では，コスト制約付きハミルトン路問題の解を全列挙する問題を対象とする．例えば，コスト制約を最短距離の 10%増 (12,868 マイル) 以内とした場合，解の総数は 16,180 個である．

3 解集合プログラミング

解集合プログラミング (ASP; [1] [4] [8] [11]) の言語は，一般拡張選言プログラムに基づいている．本稿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 (以降，論理プログラムと呼ぶ) について説明する．以降，論理プログラムのソースコードはすべて ASP システム *clingo* †1 の言語構文に従う．

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．

$$a_0 \texttt{ :- } a_1\texttt{,}\ldots\texttt{,}a_m\texttt{,not } a_{m+1}\texttt{,}\ldots\texttt{,not } a_n\texttt{.}$$

ここで，$0 \leq m \leq n$ であり，各 a_i はアトム，`not` は**デフォルトの否定** †2，カンマ (“,”) は連言を表す．ピリオド (“.”) はルールの終わりを表す終端記号である．“`:-`” の左側を**ヘッド**，右側を**ボディ**と呼ぶ．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．

ボディが空のルール (すなわち a_0 `:-`.) を**ファクト**と呼び，“`:-`” を省略してもよい．また，以下のようにヘッドが空のルールを**一貫性制約**と呼ぶ．

$$\texttt{:- } a_1\texttt{,}\ldots\texttt{,}a_m\texttt{,not } a_{m+1}\texttt{,}\ldots\texttt{,not } a_n\texttt{.}$$

例えば，`:- not` a_1,a_2. は，「a_2 が成り立つならば，a_1 が成り立つ」を意味する．

ASP 言語には，組合せ問題を解くための便利な拡張構文として，**選択子**と**個数制約**が用意されている．例えば，選択子{a_1; ...;a_n} をファクトとして書くと，「アトム集合 $\{a_1, \ldots, a_n\}$ の任意の部分集合が成り立つ」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，`:- not` lb {a_1; ...;a_n} ub,b. と書くと，「b 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．

また，組合せ最適化問題を解くために，最小化関数 (`#minimize`) と最大化関数 (`#maximize`) 等も用意されている．さらに，最近では，グラフ問題を解くための便利な拡張構文として，有向グラフの非閉路性を簡潔に記述できる`#edge` 宣言が追加されている [2]．例えば，`#edge(X,Y): arc(X,Y).` は，`arc(X,Y)` を満たす有向辺 `X` → `Y` を辺集合としてもつグラフが，閉路をもたないことを保証する．

†1 https://potassco.org/clingo/

†2 **失敗による否定**とも呼ばれる．述語論理の否定 (¬) とは意味が異なる．

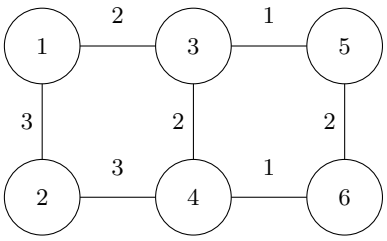


図 2　重み付き無向グラフの例

```
% node
node(1). node(2). node(3). node(4). node(5). node(6).
% edge
edge(1,2). edge(1,3). edge(2,4). edge(3,4).
edge(3,5). edge(4,6). edge(5,6).
% distance
cost(1,2,3). cost(1,3,2). cost(2,4,3). cost(3,4,2).
cost(3,5,1). cost(4,6,1). cost(5,6,2).
```

コード 1　グラフ (図 2) の ASP ファクト表現

4 ハミルトン閉路問題の ASP 符号化

図 3 に，ASP を用いたハミルトン閉路問題の解法を示す．これは，与えられた問題インスタンスを ASP ファクト形式に変換し，ハミルトン閉路問題を解く論理プログラム (ASP 符号化) と結合した後，ASP システムを用いて解を計算する一般的な解法である．本節では，ハミルトン閉路問題のインスタンスを記述するための ASP ファクト形式を説明した後，ハミルトン閉路問題を解くための 3 つの ASP 符号化 undirected, directed, acyclicity について述べる．

4.1 ASP ファクト形式

最短ハミルトン閉路問題を例にとって，入力となる重み付き無向グラフ (図 2) の ASP ファクト形式をコード 1 に示す．アトム `node/1` は頂点，`edge/2` は辺，`cost/3` は距離を表す．例えば，`edge(1,2)` は，辺 1 − 2 を表し，`cost(1,2,3)` は，この辺の重みが

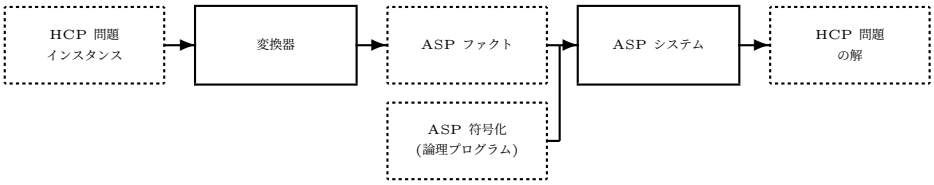


図 3　ASP を用いたハミルトン閉路問題 (HCP) の解法

```
1 { in(X,Y) } :- edge(X,Y).
2 :- not 2 { in(X,_) ; in(_,X) } 2, node(X).
3
4 reached(s).
5 reached(Y) :- reached(X), in(X,Y).
6 reached(Y) :- reached(X), in(Y,X).
7 :- not reached(X), node(X).
```

コード 2　undirected 符号化

```
1 { in(X,Y) ; in(Y,X) } 1 :- edge(X,Y).
2 :- not 1 { in(X,_) } 1, node(X).
3 :- not 1 { in(_,X) } 1, node(X).
4
5 reached(s).
6 reached(Y) :- reached(X), in(X,Y).
7 :- not reached(X), node(X).
8
9 :- not X < Y, in(s,X), in(Y,s).
```

コード 3　directed 符号化

3，すなわち，頂点 1 と頂点 2 の距離が 3 であることを示している．

4.2　undirected 符号化

グラフ $G = (V, E)$ にハミルトン閉路が存在する必要十分条件は，以下の 2 つの制約を満たす部分グラフ $G' = (V, E')$ が存在することである．

- G' の各頂点の次数が 2 (次数制約)
- G' が連結である (連結制約)

ハミルトン路の場合，次数制約は「始点と終点の次数が 1，他の頂点の次数が 2」となる．

undirected 符号化は，ハミルトン閉路問題の次数制約と連結制約を，ASP の一貫性制約と個数制約を使って表した基本的な符号化である．コード 2 に undirected 符号化を示す．コード中の s は始点の頂点番号を表し，実際の値は実行時に与えられる．

- 1 行目のルールは，各辺 `edge(X,Y)` に対して，その辺がハミルトン閉路に含まれるかどうかを意味するアトム `in(X,Y)` を選択子を使って導入している．
- 次数制約は 2 行目のルールで表される．このルールは，各頂点 `node(X)` に対して，その次数が 2 に等しくなることを個数制約を使って表している．
- 連結制約は 4–7 行目のルールで表される．ある頂点 `X` が始点 `s` から到達可能であることを意味する補助アトム `reached(X)` を導入する．4 行目のファクトは，始点 `s` が到達可能あることを表している．5 行目のルールは，各辺 `X – Y` に対して，その辺がハミルトン閉路に含まれ (`in(X,Y)`)，かつ，頂点 `X` が始点から到達可能であれば (`reached(X)`)，`Y` も到達可能であることを表している．6 行目は 5 行目と同様であるが，辺を逆向きに辿った場合を表している．7 行目のルールは，すべての頂点が始点から到達可能でなければならないことを一貫性制約を使って表している．

このように，undirected 符号化は，ハミルトン閉路問題の次数制約と連結制約を，ASP のルール 6 個で簡潔に表現している．特に，4–7 行目のように連結制約を簡潔に記述できる点は，ASP の大きな特長の一つである．

4.3　directed 符号化

directed 符号化は，undirected 符号化をベースに，与えられた無向グラフの各辺 $u - v$ に対して，2 つの弧 $u \to v$ と $v \to u$ を対応させることで有向グラフ化して解く符号化である．コード 3 に，directed 符号化を示す．

- 1 行目では，無向グラフの有向グラフ化を行う．

```
1 { in(X,Y) ; in(Y,X) } 1 :- edge(X,Y).
2 :- not 1 { in(X,_) } 1, node(X).
3 :- not 1 { in(_,X) } 1, node(X).
4
5 #edge (X,Y): in(X,Y), X != s, Y != s.
6
7 :- not X < Y, in(s,X), in(Y,s).
```

コード 4　acyclicity 符号化

与えられた無向グラフの各辺 edge(X,Y) に対して，in(X,Y) と in(Y,X) を導入し，二つの弧 X→Y と Y→X のうち高々1 個がハミルトン閉路に含まれることを個数制約を使って表している．

- 次数制約は 2–3 行目のルールで表される．2 行目のルールは，各頂点 node(X) の出次数が 1 に等しいことを表している．同様に，3 行目のルールは入次数が 1 に等しいことを表す．
- 連結制約は，directed 符号化と同様に，補助アトム reached/1 を使って表される (5–7 行目)．
- 有向グラフ化した場合，無向グラフ上の各ハミルトン閉路に対して，右回りと左回りの 2 つの対称なハミルトン閉路が存在する．9 行目のルールは，この対称解を除去する制約である．

4.4　acyclicity 符号化

acyclicity 符号化は，directed 符号化をベースに，連結制約を部分閉路を禁止する制約に置き換えたものである．コード 4 に，acyclicity 符号化を示す．directed 符号化との違いは，5 行目だけである．

- 5 行目のルールは，始点 s を含まない弧 in(X,Y) の集合をもつ有向グラフが閉路を含まないことを，*clingo* の#edge 宣言を使って簡潔に表している．

4.5　最短ハミルトン閉路問題

最短ハミルトン閉路問題の ASP 符号化は，これまで説明した各符号化に，目的関数を追加することで実現できる．undirected 符号化の場合は，コード 5 のルールを追加すればよい．この#minimize 宣言は，ハミルトン閉路に含まれる各辺 X – Y の距離 C の総和を最小化することを意味する．directed と acyclicity 符号化については，無向グラフの各辺 edge(X,Y) に対して，2 つの弧 in(X,Y) と in(Y,X) を導入している．そのため，#minimize 宣言に加えて，コード 6 の補助ルールを追加する必要がある．

```
#minimize { C,X,Y: in(X,Y), cost(X,Y,C) }.
```

コード 5　目的関数

```
cost(Y,X,C) :- cost(X,Y,C).
```

コード 6　補助ルール

```
:- not #sum { C,X,Y: in(X,Y), cost(X,Y,C) } c.
```

コード 7　コスト制約

4.6　コスト制約付きハミルトン閉路問題

コスト制約付きハミルトン閉路問題は，ハミルトン閉路問題に，距離の総和が所与の閾値以下であることを制約条件として付加した問題である．この問題の ASP 符号化は，ハミルトン閉路問題の undirected, directed, acyclicity 符号化に，コード 7 のルールを追加することで実現できる．コード中の c は閾値を表し，実際の値は実行時に与えられる．また，最短ハミルトン閉路問題の場合と同じ理由で，directed と acyclicity 符号化については，アトム cost/3 に関する補助ルール (コード 6) も追加する必要がある．

5　実行実験

考案した 3 つの符号化の性能を評価するために実行実験を行った．実験に使用したベンチマーク問題は以下の通りである．

- **ハミルトン閉路問題 (1,001 問)**
  Flinders Hamiltonian Cycle Project (FHCP) で公開されている HCP インスタンス (全 1,001 問) を使用した [5]．この問題集は FHCP Challenge と呼ばれる国際競技会で使用された頂点数が 66 個から 9,528 個の HCP インスタンスで構成されている．すべてのインスタンスはハミルトン閉路

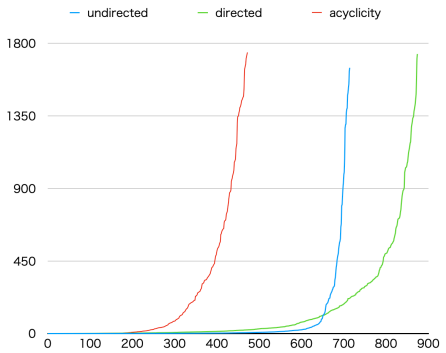


図 4　ハミルトン閉路問題: カクタスプロット

が存在する SAT インスタンスである．

- **最短ハミルトン路問題 (7 問)**
  図 1 に示した米国本土 48 州の隣接グラフに基づく問題インスタンス (1 問) と，$2N+1$ 次の正方グリッドグラフに基づく問題インスタンス (6 問，$3 \leq N \leq 8$) を使用した．グリッドグラフの辺の重みはランダムに生成した．
- **コスト制約付きハミルトン路問題 (全列挙)**
  図 1 に示した米国本土 48 州の隣接グラフに基づく問題インスタンスを使用した．コード 7 に示したコスト制約の閾値 c として，最短距離 (11,698 マイル) の N 倍となる 10 個の値，$N = 1.00, 1.01, 1.02, 1.05, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6$ を使用した．

ASP システムには，*clingo* のバージョン 5.5.0 を使用し，実験環境は Mac OS (Intel Core i7 3.2GHz 64GB メモリ) である．

5.1　ハミルトン閉路問題

表 1 に，ハミルトン閉路問題の解けた問題数を示す．左から，HCP インスタンスの頂点数の範囲，問題数，各符号化で解けた問題数を示している．各範囲において，最もよい結果をボールド体で示している．解けた問題数は，undirected 符号化が 715 問，directed 符号化が 875 問，acyclicity 符号化が 473 問であり，directed 符号化がもっとも多くの問題を解いた．directed 符号化は，どの頂点数の範囲においても他の符号化と同じかより多くの問題を解いており，その優位性が確認できる．

図 4 に，カクタスプロットを示す．縦軸は問題を解くのに要した CPU 時間 (秒)，横軸は解けた問題数を表す．グラフが右によるほど多くの問題を解けたことを示し，下によるほどより速く解けたことを示す．図 4 より，directed 符号化が，他の 2 つの符号化と比較して，より多くの問題を高速に解いていることが確認できる．

今回の実験では，ASP システム *clingo* のオプションは *trendy* を使用し，1 問あたりの時間制限は 30 分とした．また，directed と acyclicity 符号化については，個数制約の上限 (コード 3 と 4 の 1 行目) を省略したものを使用した．

5.2　最短ハミルトン路問題

表 2 に，最短ハミルトン路問題について，各符号化で得られた目的関数の値を示す．ASP システム *clingo* のオプションは *trendy* を使用し，1 問あたりの時間制限は 3 時間とした．各問題ごとに，最も良かった値をボールド体で示している．*マークは，最適値を表している．最適値と最良値の数は，undirected 符号化が 5 問，directed 符号化が 3 問，acyclicity 符号化が 3 問であり，undirected 符号化が 2 問ではあるが，他の符号化より精度の高い解を生成した．grid6 と US48 については，3 つの符号化すべてで最適値を得ることができた．最適値を得るのに要した CPU 時間は，最も速かった undirected 符号化で，grid6 は 1.451 秒，US48 は 6.68 秒であった．

5.3　コスト制約付きハミルトン路問題

表 3 に，図 1 に示した米国本土 48 州の隣接グラフに基づくコスト制約付きハミルトン路問題について，各符号化が解の全列挙に要した CPU 時間 (秒) を示す．ASP システム *clingo* のオプションは *crafty* を使用し，1 問あたりの時間制限を 3 時間とした．表の 1 列目はコスト制約の閾値と最短距離からの倍率，2 列目は解の総数を表している．各閾値ごとに，最も良かった値をボールド体で示している．表より，acyclicity 符号化が，他の符号化と比較して，より多くの問題を高速に解いていることがわかる．また，平

頂点数	問題数	undirected	directed	acyclicity
$0 \leq \lvert V \rvert < 1000$	171	156	**171**	155
$1000 \leq \lvert V \rvert < 2000$	165	120	**158**	124
$2000 \leq \lvert V \rvert < 3000$	177	125	**162**	73
$3000 \leq \lvert V \rvert < 4000$	185	104	**148**	42
$4000 \leq \lvert V \rvert < 5000$	128	92	**104**	28
$5000 \leq \lvert V \rvert < 6000$	80	63	**68**	23
$6000 \leq \lvert V \rvert < 7000$	55	39	**43**	21
$7000 \leq \lvert V \rvert < 8000$	28	12	**14**	5
$8000 \leq \lvert V \rvert < 9000$	10	2	**5**	1
$9000 \leq \lvert V \rvert < 10000$	2	**2**	**2**	1
合計	1001	715	**875**	473

表 1　ハミルトン閉路問題: 解けた問題数

問題名	頂点数	辺数	undirected	directed	acyclicity
grid6	49	84	***68,656**	***68,656**	***68,656**
grid8	81	144	113,250	113,335	**113,186**
grid10	121	220	**172,703**	174,861	172,759
grid12	169	312	**231,278**	236,211	233,657
grid14	225	420	317,617	**316,347**	318,419
grid16	289	544	**421,249**	421,696	422,963
US48	48	105	***11,698**	***11,698**	***11,698**
最適値または最良値の数			**5**	3	3

表 2　最短ハミルトン路問題: 得られた目的関数の値

閾値 (倍率)	解の総数	undirected	directed	acyclicity
11698(1.00)	1	**2.979**	7.531	4.586
11814(1.01)	8	5.587	15.322	**5.250**
11931(1.02)	28	**3.243**	18.600	3.578
12282(1.05)	388	10.003	19.818	**6.296**
12867(1.10)	16,180	16.548	28.555	**9.764**
14037(1.20)	939,209	48.262	40.717	**26.837**
15207(1.30)	4,525,541	88.172	55.276	**42.037**
16377(1.40)	6,702,964	99.154	47.647	**40.640**
17547(1.50)	6,876,526	95.390	45.265	**38.411**
18716(1.60)	6,876,928	98.937	49.138	**40.748**
平均 CPU 時間		46.8275	32.7869	**21.8147**

表 3　コスト制約付きハミルトン路問題: 解の全列挙に要した CPU 時間 (秒)

均 CPU 時間も最も短い.

6　おわりに

本稿では，解集合プログラミング (ASP) を用いたハミルトン閉路問題，最短ハミルトン閉路問題，コスト制約付きハミルトン閉路問題の解法について述べた．ハミルトン閉路問題を解く ASP 符号化として，undirected，directed，acyclicity の 3 つを考案し，ハミルトン閉路問題および関連問題の制約を，ASP のルール 10 個程度で簡潔に表現できることを確認した．考案した 3 つの符号化の有効性を評価するために，Flinders Hamiltonian Cycle Project (FHCP) で公開されているハミルトン閉路問題の問題集 (全 1001 問) を用いた実験を行った．その結果，directed 符号化が，875 問と最も多くの問題を解き，他の符号化と比較してその優位性が確認できた．

本研究で使用したハミルトン閉路問題の問題集は，2015 年から 2016 年にかけて行われた FHCP Challenge と呼ばれる国際競技会 [5] で使用されたものである．この競技会では，整数計画ソルバー CPLEX

を使用した手法 [3] が 985 問を解き優勝した．2 位となった SAT ソルバーを使用した手法は 614 問を解いた [9]．3 位から 5 位の解けた問題数はそれぞれ，488 問，464 問，385 問であった．よって，875 問を解いた directed 符号化は，ASP のルール 10 個程度の簡潔なプログラムであるにもかかわらず，実質 2 位の成績に相当する．

今後の課題としては，ハミルトン閉路問題の緩和問題と CEGAR (Counterexample Guided Abstraction Refinement) を用いた解法 [13][16] を応用した提案手法の高速化，巡回セールスマン問題への拡張などが挙げられる．

ハミルトン閉路問題は代表的な NP 完全問題であり，また重要な工学的応用が数多く存在するため，その解法については古くから盛んに研究されている．本研究と関連の深いものとしては，SAT ソルバーを用いた解法 [12][14][13]，整数計画ソルバーを用いた解法 [15] がある．これらや FHCP Challenge の上位ソルバー [3][9] との比較も重要な今後の課題である．

参 考 文 献

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

[2] Bomanson, J., Gebser, M., Janhunen, T., Kaufmann, B., and Schaub, T.: Answer set programming modulo acyclicity, *Fundamenta Informaticae*, Vol. 147, No. 1(2016), pp. 63–91.

[3] Cohen, N. and Coudert, D.: Le défi des 1001 graphes, *Interstices*, (2017).

[4] 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.

[5] Haythorpe, M.: FHCP challenge set: The first set of structurally difficult instances of the hamiltonian cycle problem, *Bulletin of the Institute of Combinatorics and its Applications*, Vol. 83(2018), pp. 98–107.

[6] 平田富夫: アルゴリズム設計とデータ構造, サイエンス社, 2015.

[7] 湊真一, 番原睦則, 堀山貴史, 川原純, 瀧川一学, 山口勇太郎: コスト制約つき組合せ問題に対する ZDD を用いた高速な解列挙手法, 信学技報, Vol. 120, No. 276, 電子情報通信学会, 2020, pp. 8–15.

[8] 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.

[9] Johnson, A.: Stedman and Erin Triples encoded as a SAT Problem, 2018.

[10] Knuth, D. E.: *The Art of Computer Programming: Bitwise Tricks & Techniques; Binary Decision Diagrams*, Vol. 4, Fascicle 1, Addison-Wesley Professional, 2009.

[11] Niemelä, I.: Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm, *Ann. Mathematics and Artificial Intelligence*, Vol. 25, No. 3–4(1999), pp. 241–273.

[12] Prestwich, S. D.: SAT problems with chains of dependent variables, *Discret. Appl. Math.*, Vol. 130, No. 2(2003), pp. 329–350.

[13] Soh, T., Berre, D. L., Roussel, S., Banbara, M., and Tamura, N.: Incremental SAT-based Method with Native Boolean Cardinality Handling for the Hamiltonian Cycle Problem, *Proceedings of the 14th European Conference on Logics in Artificial Intelligence (JELIA 2014), LNAI 8761*, Springer, 2014, pp. 684–693.

[14] Velev, M. N. and Gao, P.: Efficient SAT Techniques for Relative Encoding of Permutations with Constraints, *Proceedings of the 22nd Australasian Joint Conference on Artificial Intelligence*, LNCS, Vol. 5866, Springer, 2009, pp. 517–527.

[15] 沼田一道: 汎用 MIP ソルバによる巡回セールスマン問題の求解：多項式オーダ本数の部分巡回路除去制約, オペレーションズ・リサーチ：経営の科学, Vol. 56, No. 8(2011), pp. 452–455.

[16] 宋剛秀, 鍋島英知, 番原睦則, 田村直之, 井上克巳: CEGAR と反例の共有を用いた SAT 型 CSP ソルバーの並列化方法の考察, 人工知能基本問題研究会, Vol. 112(2020), pp. 6–11.