情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-CE-103 No.9
2010/3/6

GUIビルダツールを用いた教育用制約プログラミング環境の開発

櫻井友香[†1]　西田誠幸[†2]

プログラミングのための言語や手法は数多く存在し，その中の一つに,「制約条件を満たす答えを探して見つけ出す問題」を解くことを目的とした制約プログラミングがある．制約プログラミングは主にスケジューリングや要員配置，資源割り当てといった計画性アプリケーションを中心に実用化に至っており，この種の問題を解決することに対して優れた能力を持っている．しかし，制約プログラミングを理解するための教材は少なく，制約プログラミングを学習する環境が少ない．本稿では著者が開発している教育用制約プログラミング環境である G-Cope について述べる．G-Cope は GUI ビルダを用いており，制約のみのプログラムをユーザが記述する．それ以外の処理は G-Cope が自動生成する．

Development of Educational Constraint Programming Environment by use of GUI Builder Tool

Yuka Sakurai[†1] and Seikoh Nishita[†2]

Constraint programming is a programming paradigm where relations between variables are declaratively stated in the form of constraints. There are many applications of constraint programming, for example, human resource allocation problems and resource assignment problems in scheduling/planning systems. In addition, Constraint programming has a capability to solve constraints and find solutions satisfying the most number of constraints. A great deal of effort has been made on application of constraint system. What seems to be lacking, however, is the study on teaching materials on constraint programming. This paper states a educational constraint programming environment named “G-Cope”. G-Cope has GUI builder, and user write program of only constraint. The other processing is automatically generated by G-Cope.

1. はじめに

現在，プログラミングのための言語や手法は数多く存在し，その中の一つに制約条件を満たす答えを探して見つけ出す問題を解くことを目的とした制約プログラミングがある．制約プログラミングは主にスケジューリングや要員配置，資源割当てといった応用分野を中心に実用化に至っており，この種の問題を解決することに対して優れた能力を持っている．この制約を学習することを目的としたソフトに，ThingLab[1] と Skeleton[2] がある．これらのツールは，独自のインタフェースを用いて制約の考え方やその応用についての学習支援を目的としたものである．しかし，これらのツールでは，計算機が制約生成と解決の方法を隠蔽しており，制約プログラミングについての学習には適していない．

本稿では，我々が開発した制約プログラミングの学習支援ツール，G-Cope について述べる．G-Cope は学習題材としてパズルを用いており，ユーザはそのパズルを解くためのプログラムを考えながら制約プログラミングの学習をする．G-Cope の特徴として，ユーザが制約プログラムのうち，最も重要と思われる部分のみを記述する．ことによって，制約プログラムの作成に要する時間を軽減することができ，スムーズに制約プログラムを作成することができる．

2. 既存の教育用制約プログラミング環境

制約プログラミングと既存の教育用制約プログラミング環境について説明をする．

2.1 制約プログラミングとは

制約プログラミングとは,「問題を制約の集合として記述してコンピュータに与えることによって，コンピュータが制約を満たす答えを見つける」ことを目指すプログラミングパラダイムのことである．制約プログラミングは，制約による問題の記述とその問題を解く手順や方法によって構成される．また，制約プログラミングにおける制約の対象は，特定の領域に絞ることが多い．制約プログラミングで一般的に使用される領域を以下に示す．

- 真理値領域　　：真偽の制約
- 整数領域，有理数領域

†1 拓殖大学大学院工学研究科
Graduate School of Engineering, Takushoku University
†2 拓殖大学工学部情報工学科
Department of Computer Science, Takushoku University

- 線形領域　　：線形関数のみを記述し，解析する
- 有限領域　　：有限集合について制約を定義する
- 混合領域　　：上記のうち 2 つ以上を同時に扱う

本稿では，整数領域を使用する制約プログラミングを扱う．

制約プログラムの例として，ぶどうの房パズル[3] のプログラムを**図 1** に示す．このプログラムは Java 用の制約プログラムライブラリ Cream[4] を用いた制約プログラムである．n マスのぶどうの房パズルは，1 から n までの数字を 1 つずつ使用し，隣り合う 2 マスの差が隣接する下段のマスと等しくなるように数字を埋めるパズルである．**図 2** に 10 マスのぶどうの房パズルの例を，**図 3** にその解の一例をそれぞれ示す．なお，図 2 の図形の数字は図 1 のプログラムの配列の添数を意味する．

図 1 のプログラムについて説明する．3 行目では制約解決器の初期化を行う．4 行目から 7 行目ではドメイン変数の要素数 10 の配列を定義し，ドメイン変数の範囲を 1 から 10 と設定する．8 行目から 13 行目では制約生成関数を呼び出して制約を設定する．この制約生成関数は，ドメイン変数 v0，v1，v2 と，制約を保持する net を引数に持ち，2 つの変数 v0 と v1 の差が v2 と等しいという制約を設定する (27–32 行目)．例えば，8 行目では var[0] と var[1] の差が var[4] に等しいという制約を設定する (これは図 2 の左上の 3 マスについての制約である)．14 行目では配列の全ての変数が等しくないという制約を設定する．16 行目から 26 行目では解を導出して，結果を表示する．

2.2 教育用制約プログラミング環境の例

教育用制約プログラミング環境の例を 2 つ挙げる．

(1) **ThingLab**

ThingLab は 1981 年に開発されたソフトで，制約によるユーザインタフェースの構築についての体験的学習の支援を目的としている．ThingLab は Smalltalk[5] をベースとした，オブジェクト指向である．ThingLab はプロトタイプと呼ばれるオブジェクトを持つ．このオブジェクトは，図形で表現され，オブジェクトに制約を設定することで図形が制約通りに描画される．

(2) **Skeleton**

Skeleton は 2004 年に開発された，制約とスプレッドシートを合わせたソフトである．また，Skeleton はコンピュータに不慣れなユーザのための数学的，物理的なシミュレーションを作るシステムを備えた視覚的なスクリプトを書く環境である．Skeleton は図形に対して制約を設定する．設定方法として，Excel のワークシートに似た表を用いる．このセルに

```
 1:public class Grape {
 2:  public static void main (String[] args) {
 3:    Network net = new Network();
 4:    IntVariable var[] = new IntVariable[10];
 5:    for (int i = 0; i < 10; i++) {
 6:      var[i] = new IntVariable(net, 1, 10);
 7:    }
 8:    sub(net, var[0], var[1], var[4]);
 9:    sub(net, var[1], var[2], var[5]);
10:    sub(net, var[2], var[3], var[6]);
11:    sub(net, var[4], var[5], var[7]);
12:    sub(net, var[5], var[6], var[8]);
13:    sub(net, var[7], var[8], var[9]);
14:    new NotEquals(net, var);
15:
16:    Solver solver = new DefaultSolver(net);
17:    for (solver.start(); solver.waitNext(); solver.resume()) {
18:      Solution solution = solver.getSolution();
19:      System.out.println(solution.getIntValue(var[0])
                            + " " + solution.getIntValue(var[1])
                            + " " + solution.getIntValue(var[2])
                            + " " + solution.getIntValue(var[3]));
20:      System.out.println(" " + solution.getIntValue(var[4])
                            + " " + solution.getIntValue(var[5])
                            + " " + solution.getIntValue(var[6]));
21:      System.out.println("  " + solution.getIntValue(var[7])
                            + " " + solution.getIntValue(var[8]));
22:      System.out.println("   " + solution.getIntValue(var[9]));
23:      System.out.println();
24:    }
25:    solver.stop();
26:  }
27:  public static void sub (Network net,
                              IntVariable v0, IntVariable v1, IntVariable v2) {
28:    IntVariable max = v0.max(v1);
29:    IntVariable min = v0.min(v1);
30:    IntVariable v = max.subtract(min);
31:    new Equals(net, v, v2);
32:  }
33:}
```

図 1 制約プログラムの一例
Fig. 1 A Sample of The Constraint Program

Vol.2010-CE-103 No.9
2010/3/6

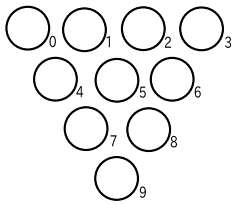

図 2 ぶどうの房パズル
Fig. 2 A Bunch of Grape Puzzle

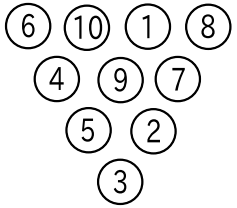

図 3 解の一例
Fig. 3 A Solution of The Bunch of Grape Puzzle

数式の代わりに制約を記述する．そして，セルと図形オブジェクトの座標の関連付けや，図形を数式にしたがって動かすことが可能である

2.3 問 題 点

前節では，既存の教育用制約プログラミング環境の 2 つを大まかに説明した．このようなツールを用いることで，制約プログラミングの学習の敷居が下がり，楽しく体験的に学ぶことができる．しかし一方で，ThingLab は，制約プログラムの一部しか記述せず，制約プログラムの全体が見えない．また，Skeleton は，制約は計算機が生成し，それらの解決するための方法を隠蔽している．そのため，制約プログラミングを学習するには十分といえない．

3. 提案する学習環境

本稿で提案する学習環境は，制約プログラミングの学習支援を目的とした，教育用制約プログラミング環境である．

3.1 開 発 方 針

開発するツール，G-Cope は，制約プログラミングの学習支援を目的としている．そのため，ユーザが制約プログラムを読み書きできる環境を作る．G-Cope は学習題材としてパズルを用いる．このパズルはぶどうの房パズルやグラフの 3 彩色問題[6]，数独[7]，ナイトツアー[8] などのパズルである．G-Cope がパズルを題材とする理由は，これらのパズルが制約プログラミングの学習に用いる代表的な問題であり，パズルを理解するためには予備知識が不要なので，ユーザは制約プログラムを作ることに注力ができるからである．また，パズルのルールは，制約に近い形式で表現されることが多い．そのため，ユーザはパズルの制約を作成しやすいのではないかと考えたからである．

制約プログラムの一般的な流れは次の通りになる．

```
public class Grape {
  public static void main(String[] args) {
    Network net = new Network();
    IntVariable var[] = new IntVariable[10];

    for (int i = 0; i  < 10; i++) {
      var[i] = new IntVariable(net, 1, 10);
    }
    sub(net, var[0], var[1], var[4]);
    sub(net, var[1], var[2], var[5]);
    sub(net, var[2], var[3], var[6]);
    sub(net, var[4], var[5], var[7]);
    sub(net, var[5], var[6], var[8]);
    sub(net, var[7], var[8], var[9]);
    new NotEquals(net, var);

    Solver solver = new DefaultSolver(net);
    for (solver.start(); solver.waitNext(); solver.resume()) {
      Solution solution = solver.getSolution();
      System.out.println(solution.getIntValue(var[0])
                         + " " + solution.getIntValue(var[1])
                         + " " + solution.getIntValue(var[2])
                         + " " + solution.getIntValue(var[3]));
      System.out.println(" " + solution.getIntValue(var[4])
                         + " " + solution.getIntValue(var[5])
                         + " " + solution.getIntValue(var[6]));
      System.out.println("  " + solution.getIntValue(var[7])
                         + " " + solution.getIntValue(var[8]));
      System.out.println("   " + solution.getIntValue(var[9]));
      System.out.println();
    }
    solver.stop();
  }
  public static void sub(Network net, IntVariable v0, IntVariable v1, IntVariable v2) {
    IntVariable max = v0.max(v1);
    IntVariable min = v0.min(v1);
    IntVariable v = max.subtract(min);
    new Equals(net, v, v2);
  }
}
```

1. 変数設定
   制約解決器の初期化
   ドメイン変数の宣言
   範囲設定
2. 制約設定
4. 解の導出
   結果表示
3. 制約生成関数の作成

図 4 注釈付き制約プログラム
Fig. 4 Constraint Program with Annotation

(1) 変数設定
(2) 制約設定
(3) 制約生成関数の作成
(4) 解の導出

制約プログラムは，変数宣言や制約解決器の初期化などの準備と，制約設定，制約生成関数の作成，制約の解決の大きく 4 つに分けられる．制約生成関数は必ずしも記述するわけではなく，同様の制約を複数作成する必要があり，複雑な制約を設定する場合に用いられる．

図 1 のプログラムに注釈を付加したものを再度**図 4** に示す．

(1)，(4) はどの問題でも同様の形式で記述する．(2) は制約生成関数を使用する場合，どのドメイン変数に対して制約設定をするかだけを考えて記述する．しかし，(3) は各問題にあったプログラムを記述する必要がある．

以上のことから，G-Cope の開発方針として以下の 4 つを挙げる．

- G-Cope は制約プログラムを生成するものとする
- 生成した制約プログラムを実行すると，制約を解決し結果を表示する
- 制約プログラムのうち，(3) の制約生成関数だけをユーザが直接エディタで記述できるようにする
- ユーザが本来作成しなくてはならない (1) や (2)，(4) などのプログラムのほとんどを G-Cope が自動生成する

制約プログラミングで最も大切なのは，問題から制約を見つけ出すことであり，制約プログラムにどのような制約を作る必要があるかを考えることがが重要である．このため，制約生成関数だけをユーザが記述し，それ以外の制約プログラミング学習に比較的重要でないと思われる部分を G-Cope が自動生成することとする．自動生成する部分とは，ドメイン変数の宣言や制約設定，制約解決処理の呼び出しなどである．G-Cope は全て自動生成するわけではなく，ドメイン変数をいくつ宣言するか，どの制約を設定するかなどはユーザが指定をする．指定するための方法として GUI を用いる．一般に GUI の生成とプログラムの作成を組み合わせたツールとして GUI ビルダがある．そこで，G-Cope を，GUI ビルダを用いて作成することとする．

G-Cope は ThingLab や Skeleton とは違い，学習題材としてパズルを用いる．G-Cope は制約生成プログラムを生成するため，ThingLab とは違い，制約プログラムの全体を把握することができる．また，制約プログラムの一部を直接記述することができる．さらに，GUI を用いてパズルを表現するため，楽しく体験的に制約プログラミングを学ぶことができる

3.2 G-Cope の構成

G-Cope は Eclipse プラグインであり，GUI エディタとプログラムエディタで構成される．G-Cope の構成図を**図 5** に示す．GUI エディタは，自由に図形を描画する機能を持つ．GUI エディタ上で図形を 1 つ作成することは，制約プログラムでドメイン変数を 1 つ宣言することを意味する．プログラムエディタは，制約生成関数を書くためのエディタである．G-Cope が自動生成したプログラムをコンパイルすることで，ソルバソフトウェアを作成する．ソルバソフトウェアは制約解決機能を持ち，実行することで解の導出と結果表示を行う．

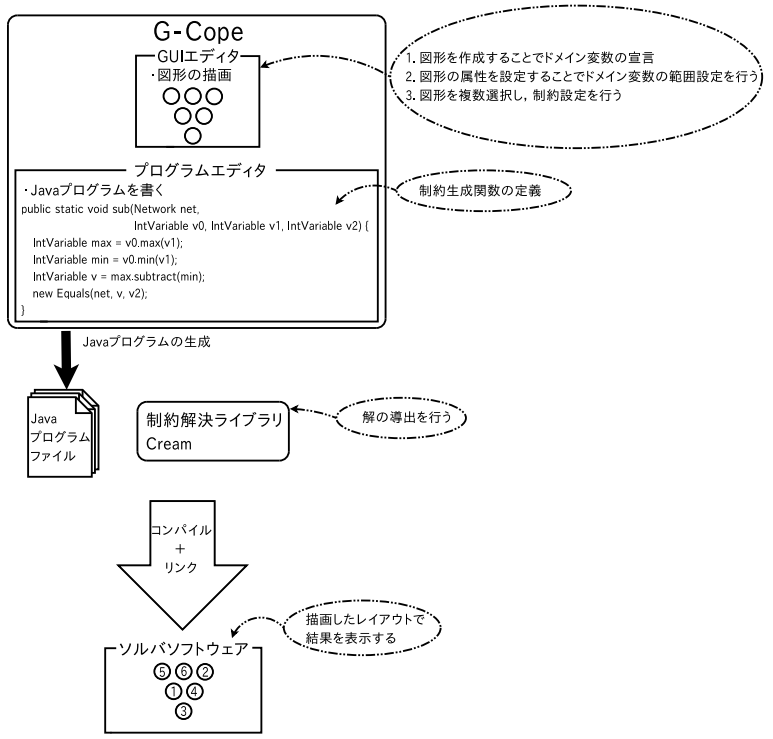

図 5 G-Cope の概要
Fig. 5 Structure of G-Cope

3.2.1 GUI エディタ

GUI エディタはユーザがパズルを作成するために，図形描画する機能や削除機能などのドローツールとしての基本的な機能を持つ．GUI エディタで図形の属性を設定することによって，制約プログラムにおけるドメイン変数の範囲設定を行えるようにする．**図 6** に GUI エディタの表示例を示す．

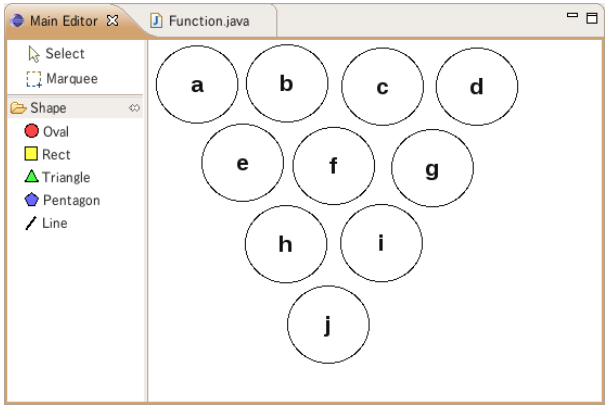

図 6 GUI エディタの表示例
Fig. 6 Appearance of GUI Editor

3.2.2 プログラムエディタ

プログラムエディタは，制約生成関数を書くためのエディタである．ユーザは関数名を自由に決めることができる．この関数内に Cream の API に記載されている関数を使用し，制約生成関数を作成する．**図 7** にぶどうの房パズルの制約生成関数を記述したプログラムエディタの表示例を示す．

3.2.3 ツリービューア

ツリービューアは制約生成関数，制約，図形を管理する．ユーザは GUI エディタ上の図形と，ツリービューア上の制約生成関数を選択し，制約設定を行うことができる．**図 8** にツリービューアの表示例を示す．

3.2.4 ソルバソフトウェア

解の導出にはソルバソフトウェアを使用する．ソルバソフトウェアは解の導出機能と，結果表示機能を持つ．ソルバソフトウェアの概観を**図 9** に示す．ユーザが解の導出命令を下すことで，ソルバソフトウェアは解を探索し，結果を GUI エディタで描画したレイアウトにしたがって表示する．結果の表示例を**図 10** に示す．

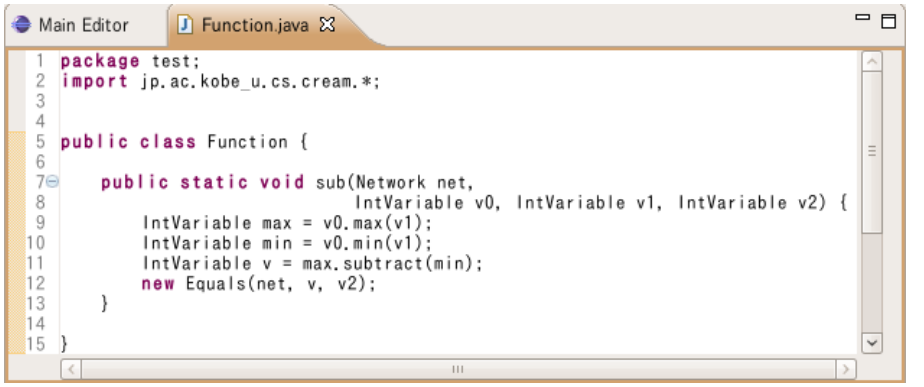

図 7 プログラムエディタの表示例
Fig. 7 Appearance of Program Editor

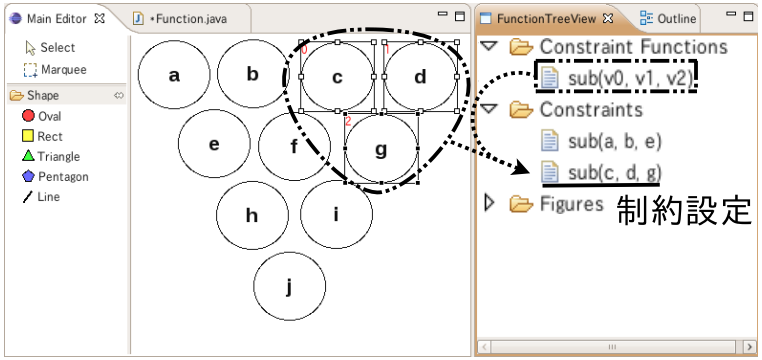

図 8 ツリービューアの表示例
Fig. 8 Appearance of Tree Viewer

3.3 G-Cope の実装

G-Cope は Eclipse のプラグインとして開発した．Eclipse は Java 開発における標準的な統合開発環境である．Eclipse はプラグインアーキテクチャを採用しており，プラグインをインス

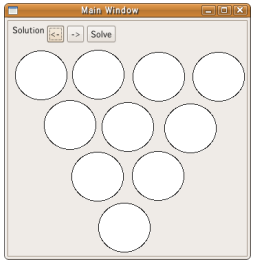

図 9 ソルバソフトウェアの概観
Fig. 9 Appearance of Solver Software

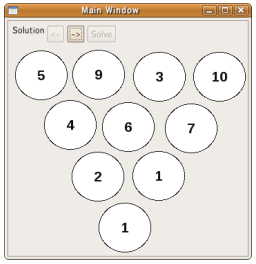

図 10 結果の表示例
Fig. 10 Appearance of The Solution

トールすることで機能拡張ができる．G-Cope の開発には，Eclipse が提供する PDF(Plugin Development Environment) を使用した．特に，GUI エディタには GEF(Graphical Editing Framework)，プログラムエディタには JDT(Java Development Tool)，ツリービューアには JFace のビューアクラスをそれぞれ使用した．また，ソルバソフトウェアには RCP(Rich Client Platform) を使用した．

3.4 評価実験

G-Cope の効果を測るために評価実験を行った．学習効果を測ることは難しい．そこで，本評価実験では G-Cope と全ての制約プログラムを記述する Cream を用いて，パズルの制約プログラムを作成させ，作成に要する時間を計測することとした．この時間によって，プログラミングがどれだけスムーズに行えたのか指標を測る．また，結果までのプロセスが短い方が制約を考えることに多く時間を割り当てられると考えられる．実験の目的を以下に掲げる．

- 制約プログラムを全て入力するのではなく，制約の部分だけをプログラムすることで入力の手間数を軽減できることを明らかにする
- 図形を用いて表示することで制約の入力ミスの防止，結果を見やすく表示できているかを明らかにする

評価実験は 2 日間実施した．各実験日でそれぞれ別々のパズルの問題を出題し，被験者に制約プログラムを作成させた．被験者は Java プログラミングの経験を持ち，制約プログラミングの知識がない 6 名である．6 名を G-Cope，Cream の順番で制約を作成するグループと，Cream，G-Cope の順番で制約を作成するグループの 2 つに分けて実験を実施した．以下，評価実験について 1 日目と 2 日目に分けて説明をする．

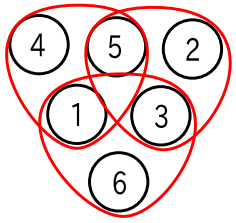

図 11 トライアングルパズルの一例
Fig. 11 A Sample of Triangle Puzzle

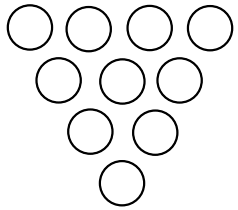

図 12 問題のトライアングルパズル
Fig. 12 The Problem of Triangle Puzzle

1 日目に評価実験を行った 10 マスのトライアングルパズルは，1 から 10 までの数字を 1 つずつ使用し，下段のマスと隣接する上段のマスの合わせて 3 マスの和が全て等しくなるように数字を埋めるパズルである．**図 11** にトライアングルパズルの例を，**図 12** に実験に用いたトライアングルパズルを示す．この実験では 30 分の時間制限を設け，制約プログラムの作成に要した時間を計測した．**表 1** に制約プログラムの作成に要した時間を示す．

表 1 トライアングルパズルの制約作成に要した時間
Table 1 Constraint making time of Triangle Puzzle

ツール / 被験者	Cream(分)	G-Cope(分)	備考
A	17.17	12.38	Cream
B	25.35	11.09	↓
C	28.08	10.93	G-Cope
D	12.05	9.27	G-cope
E	20.03	27.57	↓
F	19.02	26.03	Cream

被験者 A，B，C は Cream，G-Cope の順番で制約プログラムを作成し，被験者 D，E，F は G-Cope，Cream の順番で制約プログラムを作成した．被験者全員が 30 分以内に制約プログラムを作成できた．また，被験者のほとんどが，最初に使用したツールで制約プログラムを作成した方が作成に要する時間が長かった．

以下に被験者の感想を示す．

- G-Cope は実行結果をイメージしやすい
- 図形と変数名が対応しているので，制約と図形の関係がすぐにわかる

- 制約関数に割り当てるだけで制約を複製できるので，手間が省ける
- Cream はエラーを削除するのに時間がかかったが，G-Cope はエラーや入力ミスがなかった
- 結果表示に自分の組み立てた図形内で表示するので，わかりやすかった
- G-Cope だと制約プログラミングがわからない
- G-Cope は重くて，すぐに落ちる
- G-Cope は使い方がわかりづらい

2 日目の評価実験の説明を行う．2 日目に評価実験を行ったグラフの 3 彩色問題は，隣接する頂点同士が同じ色にならないように全頂点に 3 つの色を用いて彩色する問題である．**図 13** にグラフの 3 彩色問題の一例を示す．また，グラフの 3 彩色問題は解のある問題とない問題の 2 種類のパズルを用いた．実験に用いたグラフの 3 彩色問題を**図 14**，**図 15** に示す．図 15 は答えがなく，辺を 1 つ取れば解が導き出される問題である．この実験では，グラフの 3 彩色問題は時間制限を設けず制約プログラムの作成に要する時間を計測した．**表 2** に制約プログラムの作成に要した時間を示す．

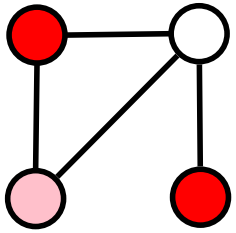

図 13 グラフの 3 彩色問題の一例
Fig. 13 A Sample of The Three Coloring Problem

被験者 A，B，C は Cream，G-Cope の順番で制約プログラムを作成し，被験者 D，E，F は G-Cope，Cream の順番で制約プログラムを作成した．被験者のほとんどが G-Cope で制約プログラムを作成した方が作成に要する時間が短かった．

以下に被験者の感想を示す．

- G-Cope はプログラムをあまり書かなくてよかった
- 制約生成関数を割り当てるだけなのでプログラム作成の手間が省ける
- GUI を使用しているので G-Cope の方が実行結果，制約の関係がわかりやすい

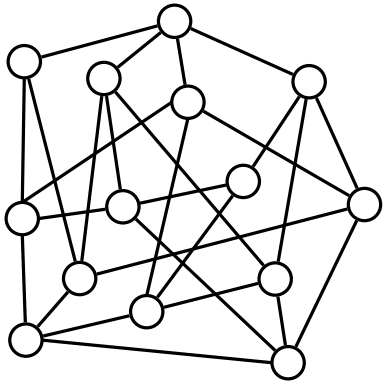

図 14 問題のグラフの 3 彩色問題
Fig. 14 Problem of The Graph Three Coloring Problem

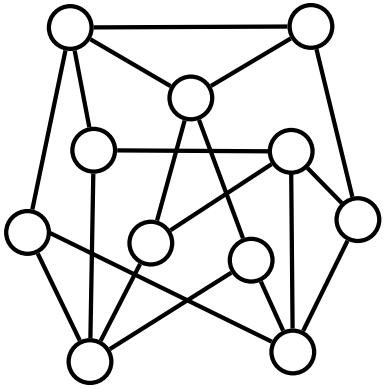

図 15 答えのないグラフの 3 彩色問題
Fig. 15 The Graph Three Coloring Problem without a Solution

表 2 2 つのグラフの 3 彩色問題制約作成に要した時間
Table 2 Constraint making time of Three Coloring Problem

ツール / 被験者	Cream(分)	G-Cope(分)	備考
A	132.22	39.53	Cream
B	47.15	24.00	↓
C	59.42	40.47	G-Cope
D	66.97	67.25	G-Cope
E	75.17	47.22	↓
F	49.83	75.92	Cream

- 2 つとも制約をどの変数に割り当てたか確認をするのが大変だった
- Cream は制約作成の手間がかかる
- Cream は制約の関係がわかりづらい
- Cream は結果表示のレイアウトを考えなくてはならないので大変だった
- Cream は同じ制約を何度も記述するため，小さなミスを起こしやすい
- 図形の描画処理が細かく設定できない
- 図形に対してだけでなく，線にも情報を持たせて欲しい

3.4.1 考　　察

1 回目の実験では G-Cope，Cream の順番で制約プログラムを作成した方が作成に要する時間が長かった．この理由として，被験者たちは制約プログラムの知識がなかったが，プログラムの全体を想像しながら，プログラムの一部 (制約生成関数) を記述することを強いられたためだと考えられる．一方，2 回目の実験では，ほとんどの被験者たちが Cream よりも G-Cope の方が短い時間で制約を作成し終えている．この理由として，制約プログラムを一度学習したので，プログラムの全体が想像でき，制約生成関数に何を記述すればよいかわかったからだと考えられる．また，2 回目のパズルは制約設定が不規則であるため，Cream ではプログラムリストと図を照らし合わせながら制約を作成する必要がある．このため，時間がかかったと考えられる．

以上のことから，G-Cope についての考察を述べる．制約プログラムの全体を知らない人に対して，制約生成関数のみを記述させる G-Cope を使用させることは難しい．しかし，制約プログラムを一度学習したことがあり，さらに図 14 のように制約設定をする変数が多く，複雑な制約を設定する場合は G-Cope の方が制約プログラムを記述する手間が省け，制約プログラムをスムーズに行うことができる．

そこで，G-Cope を使用した学習方法を考えた．第一段階で，簡単なパズルの制約プログラムのプログラム全体を記述させ，第二段階で，G-Cope を使用して難しい問題の制約プログラムを記述させる．与える問題は，数独のような規則正しいレイアウトで表示される問題ではなく，図 14 のような不規則なレイアウトで表示される問題とする．

3.5 おわりに

本稿では，教育用制約プログラミング環境である G-Cope について述べた．このツールは，パズルを題材とした問題を解くためのソフトウェアの作成を通じて，制約プログラミングについての学習支援を目的としたツールである．

また，このツールによる学習効果を知るため，評価実験を行った．評価実験により，G-Cope の使用方法の考察を行った．その結果，G-Cope は制約プログラムの知識がない人ではなく，制約プログラムの流れを把握している人により適した学習支援ツールであるとわかった．

参 考 文 献

1) ALAN Borning.: *The Programming Language Aspects of ThingLab, A Constraint-Oriented Simulation Laboratory* (1981).
2) Takashi Yamamiya.: *Skeleton, easy simulation system* (2004).
3) ぶどうの房パズル：http://www2.oninet.ne.jp/mazra/math104.htm
4) Naoyuki Tamura.: *Cream:Class Library for Constraint Programming in Java* (2003).
5) Smalltalk：http://www.smalltalk.org/main/
6) Vertex Coloring：http://mathworld.wolfram.com/VertexColoring.html
7) 数独：http://www.nikoli.co.jp/ja/puzzles/sudoku/
8) Knight's Tour Notes：http://www.ktn.freeuk.com/index.htm