

自動漫画生成システムにおける GE を用いた コマ割り生成手法

Layout Generation Method Using GE for an Automatic Comic Generation System

福本 亮†
Akira Fukumoto

THAWONMAS Ruck
Ruck THAWONMAS

1. はじめに

漫画の自動生成については様々な研究がなされてきた。従来研究として[1]がある。これは一人称視点でのシューティングゲーム(FPS:First Person shooter)のプレイログから漫画を自動生成する研究であった。この研究を参考に著者の所属する研究室では多人数同時参加型 RPG (MMORPG: Massively Multiplayer Online Role-Playing Game) のログから漫画の自動生成を行う研究がある[2]。また、ゲームだけでなく映画の要約の形式として漫画形式を採用している研究もある[3]。これら漫画自動生成の研究は様々な環境で進められているが、活用例として自身の経験の回顧と情報の共有化があげられる。漫画という媒体は長い間文化として親しまれてきたため馴染みやすく、流し読みをするだけでも全体を把握できる。さらに動画と違い容量が軽く、自身のペースで読み進められるためユーザにとって自由度が高い。よって自動で漫画が生成されるシステムが完成すれば、新しい体験の共有方法として漫画が期待できる。

その中でも筆者はコマ割りに注目した。コマ割りは漫画媒体独自の表現技法であり、漫画を漫画たらしめている要因である。従来研究[1]や[2]でもコマ割りは行われているがシンプルな横割りだけであり、昨今の漫画雑誌等と比べると漫画としての視覚的な魅力に欠ける。本研究ではコマ割りを改善し、漫画媒体としての魅力を高めるとともに、体験集約という観点からよりユーザー体験を印象深いものにすることを目的としている。

本研究ではメタバースとして代表的な Second Life[4]の博物館の訪問に注目している。SecondLife の特徴として、ブラウザのソースコードを公開しており、実験環境においては自由度が高いことが上げられる。また、空間を自由に動き回れる環境があるため、他のデジタルコミュニケーションとは違った環境がメタバースにはある。

また漫画自動生成の研究においても、博物館見学等の経験を漫画を用いて共有することは代表的な活用例としてあげられている[5]。よって博物館の訪問体験の漫画化は、メタバースの特徴と漫画自動生成の研究において代表的なテーマの一つであると筆者は考えている。

2. システム

今回はメタバース空間として SecondLife を使用し、その SecondLife 上の美術館内での行動を漫画形式で出力する際に、ユーザ好みのコマ割りを施すことが目的となっている。ユーザの好みを反映するために後述のインタラクティブ GE[6]を活用し、ユーザの介入を受けながらコマ割りを進化させていく。

この漫画生成システムは出力する漫画分のコマ割りを作成し、何コマ必要かを計算した後、プレイログを読み込んでコマを取得、取得したコマを作成しておいたコマ割りに割り当て出力し、ユーザに評価を行ってもらう。そこで得た評価を元に規程世代数に到達するまで進化とコマ割りを行う。最終的に一番評価値が高い漫画を出力する(図1)。

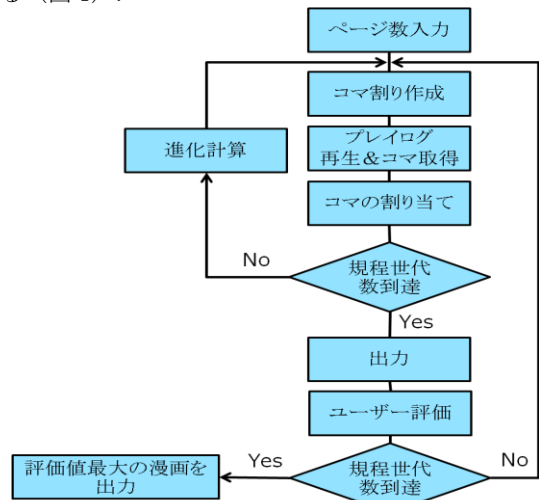


図1 システムの流れ

2.1 コマ割り手法と進化

漫画のコマの配置は、ページごとに直線で分割して繰り返すと各コマに分割することが可能で、その分割過程は木構造で表現できるという既存研究[7]を応用し、コマ割りを表現する木構造(以下、コマ割り木)から漫画のコマ割りを生成していく。このコマ割り木を生成するのに Grammatical Evolution(GE)[8]という進化手法を用いるが、進化過程でユーザーの介入を行いよりユーザー好みのコマ割りが生成されるように改良を加えたインタラクティブ GE(以下、iGE)を使用する。

コマ割り木は GE の文法により生成されるが、実際の漫画ではページごとにコマ割りの仕方が異なり、また多くのパターンが存在するため、1つの GE の文法で表現しようとすると煩雑になる。そこで、筆者は文献[9]よりコマ割りのパターンを4つに種類分けし、そのパターンごとに使用する GE の文法を変えることにした。

よって、iGE で進化させる個体は、ページごとに大まかにどのようなコマ割りにするかを4種類の中から決定するパターン設定個体と、実際に文法を用いてコマ割りを行うコマ割り個体の2つとなる。この2つの個体はパターン設定個体同士、コマ割り個体同士で交叉、突然変異等の進化処理を行う。コマ割り個体の長さはページ数で決まる可変長である。

†立命館大学院理工学研究科情報理工学専攻, Ritsumeikan University Graduate School of Science and Engineering

2.2 コマ割りパターン

ページごとのコマ割りは 4 パターンから選ばれる．パターンはそれぞれ以下の通りである．

1. スタンダード
2. インパクト
3. ターニング
4. テンポ

スタンダードパターンは，横割り(図 2 参照)のみを行いコマ割りを行う．1 ページでのコマ割りは 5, 6 コマになるようにコマ割りを行う．

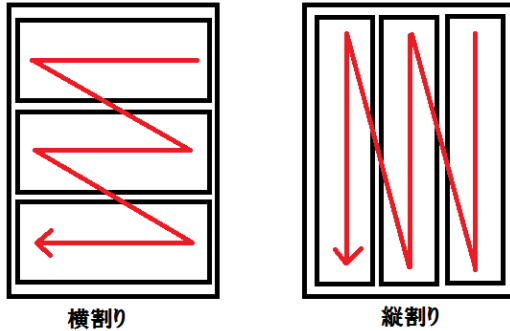


図 2 コマの割り方の違い

インパクトパターンではページ中に 1 つだけ大コマを割り当て，大コマに割り当てられたコマを印象深いものにする．このパターンでは，横割りと縦割りのどちらかが採用される．

ターニングパターンは，場面の移り変わりを表現することに特化したコマ割りである．参考文献[9]によると，縦割りは状況の遷移を表現するのに向いているとあるので，このパターンでは縦割りのみ使用する．

テンポパターンは，読者が体感する漫画のスピード感を上げるため，小さなコマを多く載せるようにしている．次々とコマを読み進めてもらうため読みやすい横割りのみを使用する．

図 3, 4, 5, 6 はスタンダード，インパクト，ターニング，テンポでそれぞれ用いる文法である．各文法中に見られる<slant>は斜め割をするかどうかを表す文法であり，<angle>により斜めの角度が決定する．横方向の斜め割が連続する場合は，図 7 のように平行にならないように角度の正負を逆転させる．

また，leaf は終端記号でありコマ割り木におけるコマを表している．図 8 はスタンダード文法を用いて生成したコマ割り木とコマ割りとなっている．

```
<standard> ::= <ver><slant><ver><slant><ver> |
               <ver><slant><ver><slant>leaf |
               <ver><slant>leaf<slant><ver> |
               leaf<slant><ver><slant><ver>
<ver> ::= leaf<slant>leaf;
<slant> ::= <angle> | false;
<angle> ::= 5; | 10; | 15;
```

図 3 スタンダード時の文法規則

文法<standard>は漫画が何段に別れるかを表しており，例えば<ver><slant><ver><slant><ver>が選択された場合，生成されるコマ割りは 3 段に分けられたものになる．インパクト文法規則中の<hor_base>，<ver_base>，ターニング文法規則中の<tarning>，テンポ文法規則中の<tempo>も同様に処理される．

```
<impact> ::= <root>
<root> ::= <Hor_base> | <Ver_base>

<hor_base> ::= big<slant><Hver> | <Hver><slant>big
<Hver> ::= leaf<slant>leaf | leaf<slant>leaf<slant>leaf |
            leaf<slant>leaf<slant>leaf<slant>leaf |
            <Hhor><slant><Hhor> | <Hhor><slant>leaf |
            leaf<slant><Hhor>
<Hhor> ::= leaf<slant>leaf;

<ver_base> ::= big<slant><Vhor> | <Vhor><slant>big
<Vhor> ::= leaf<slant>leaf | leaf<slant>leaf<slant>leaf |
            leaf<slant>leaf<slant>leaf<slant>leaf |
            leaf<slant><Vver> | <Vver><slant>leaf
<Vver> ::= leaf<slant>leaf;
<slant> ::= <angle> | false;
<angle> ::= 5; | 10; | 15;
```

図 4 インパクト時の文法規則

<root>で横割るか縦割りかを選択した後，それぞれ対応した文法規則を使いコマ割り木を構築する．big は大コマでありページの半分を占めるようになっている他，コマの外枠を使ってコマを配置する断ち切りと呼ばれるコマ割り手法を採用している．これにより大コマがより印象深いものになる効果がある．

```
<tarning> ::= leaf<slant>leaf<slant><hor> | leaf<slant><hor><slant>leaf |
               <hor><slant>leaf<slant>leaf | <hor><slant><hor><slant>leaf |
               <hor><slant>leaf<slant><hor> | leaf<slant><hor><slant><hor>
<hor> ::= leaf<slant>leaf;
<slant> ::= <angle> | false;
<angle> ::= 5; | 10; | 15;
```

図 5 ターニング時の文法規則

```
<tempo> ::= <ver><slant><ver><slant><ver>
<ver> ::= leaf<slant>leaf | leaf<slant>leaf<slant>leaf |
            leaf<slant><hor> | <hor><slant>leaf
<hor> ::= leaf<slant>leaf;
<slant> ::= <angle> | false;
<angle> ::= 5; | 10; | 15;
```

図 6 テンポ時の文法規則

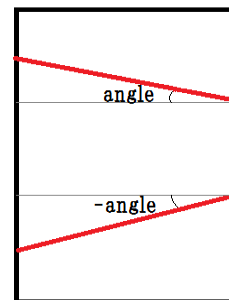


図 7 連続した横方向の斜め割

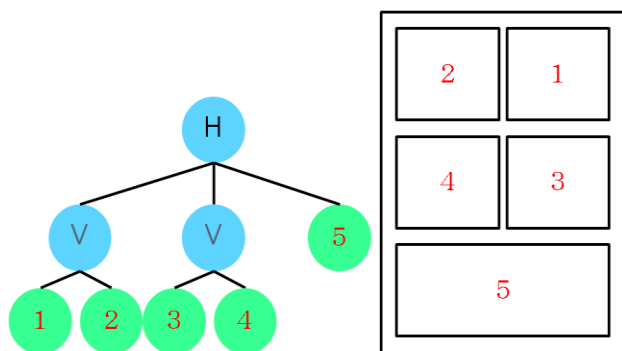


図 8 コマ割り木とコマ割り

3 コマ割りの微調整

完成したコマ割りにコマ候補群からコマを割り当てていく。この際、視線誘導を考慮したコマ割りの微調整を行う。上のコマと下のコマの横幅が同じだと、読者が次にどのコマを読めばいいのか分からなくなってしまう。そうならないように上下のコマの横幅に差を持たせることで読者がスムーズに、且つこちらが意図した通りの順番に漫画を読み進めてもらうことを視線誘導という。

3.1 コマの横幅の微調整

図 9 のような横割り 3 段の各段 2 コマ構成のコマ割りの場合、最初のコマは「つかみ」と呼ばれる、読者を引き込む漫画技法のため横幅を大きく取る。2 段目の 3, 4 コマの横幅は 4 コマ目を若干大きく取る。3 段目は「めくり」と呼ばれる、次のページの展開を期待させる漫画技法を活用するため 6 コマ目の横幅を大きく取るようにしている。横幅の微調整幅の決定は指定された範囲内からランダムで決定される。以下に 1 段中に 2 コマ含まれる場合の微調整アルゴリズムの偽コードを示す。

```
float adjust_1st = random(1)
float adjust_Final = random(1)
float adjust_else =
    (adjust_1st+adjust_Final) / (1+random(1))
for(i=0; i < RowNumber; i++){
    if(FrameOfRow[i]==2){
        if(1stFrameExists==true || FinalFrameExists==true){
            if(Frame_number[0] == 1st){
                Frame_number[0].width *= (1 + adjust_1st)
                Frame_number[1].width *= (1 - adjust_1st)
            }
            else(Frame_number[1] == Final){
                Frame_number[1].width *= (1 + adjust_Final)
                Frame_number[0].width *= (1 - adjust_Final)
            }
        }
    }
    else{
        Frame_number[0].width *= (1-adjust_else)
        Frame_number[1].width *= (1+adjust_else)
    }
}
```

adjust_1st, *adjust_final*, *adjust_else* はそれぞれ 1 コ

マ目と最後のコマ、およびそれ以外の調整幅を示している。*RowNumber* はそのページ中に何段あるかを示しており、*FrameOfRow* は段中に何コマあるかを示している。

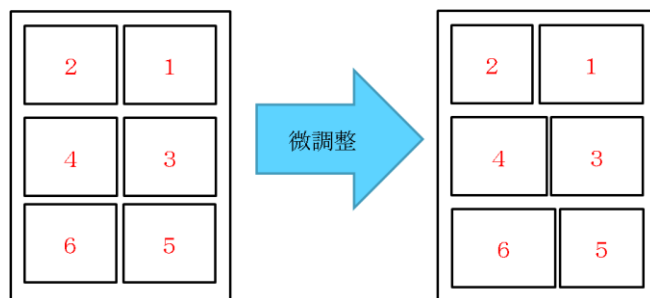


図 9 微調整例

他の横割りパターンでも最初と最後のコマが若干大きくなるように調整する。

また、テンポ文法やインパクト文法で取りうる 1 段に 3 コマある場合(図 10 参照)でも最初と最後のコマを若干大きく取るようにする。

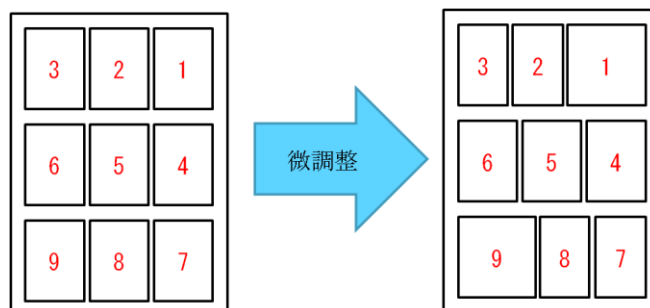


図 10 1 段に 3 コマある場合の微調整

ターニング文法での縦割りの横幅調整も同様に行うが、縦割りは特殊なコマ割りのため横幅を変えなくても読みやすさは変わりにくいと考えられる。そのため、横幅に調整を加えるかどうか自体をランダムに決定し、調整する場合の調整幅も横割りでの場合と同様にランダムで決定する。

3.2 コマの縦幅の微調整

横幅の調整を行ったあとで縦幅についての調整を行う。横割りにおける各段の縦幅は、図 11 のように各段で一番大きいコマの横幅を比較し、その比率を使って調整する。偽コードは以下のとおりである。

```
for(i=0; i < RowNumber; i++){
    SumLargestFrameWidth += LargestFrameWidth[i]
}

for(i=0; i < RowNumber; i++){
    AdjustRowHeight =
        LargestFrameWidth[i] / SumLargestFrameWidth
    RowHeight *= AdjustRowHeight
}
```

LargestFrameWidth は各段中の最大のコマの大きさであり、*SumLargestFrameWidth* はその合計である。

$AdjustRowHeight$ は段の高さを調整する幅である。

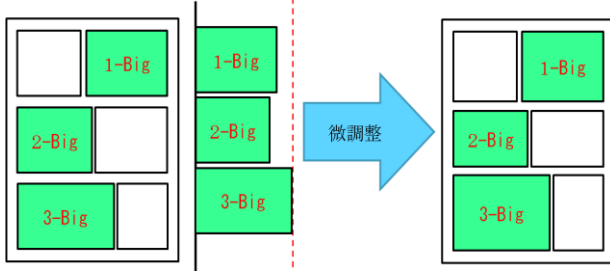


図 11 横幅の比率による縦幅の調整

4 進化とユーザー評価

コマ割りの微調整を行った後に漫画を一度出力し、ユーザーに漫画の評価を行なってもらい、評価の仕方は、まず出力された 5 つの漫画の中で最も好ましいもの 1 つと最も好ましくないもの 1 つを選んでもらう。好ましい漫画を best 漫画、好ましくない漫画を worst 漫画とし、各漫画をそれら 2 つと比較して評価値 ($Fitness$) を算出する。漫画の比較は木構造の比較で行われる。

4.1.1 木の編集距離

この手法は木構造 A と完全一致するまで木構造 B に分解と合成の変形操作を施し、その操作回数によって類似度を求めるものである [10]。まず、比較対象となるコマ割り木 T_a の部分木集合 $SF(T_a)$ とコマ割り木 T_a に用いられている文法を使ってできる木構造集合 (以下、文法規則部分木集合) $IF(T_a)$ を作成する。また、操作回数 n 回目にできる部分集合 (以下、操作部分集合) を $CF(T_a, T_b, n) = \{(\beta_{b,1}), (\beta_{b,2}) \dots (\beta_{b,n})\}$ とし、編集回数 0 回目の操作部分集合を $CF(T_a, T_b, 0) = \{T_b\}$ とする。

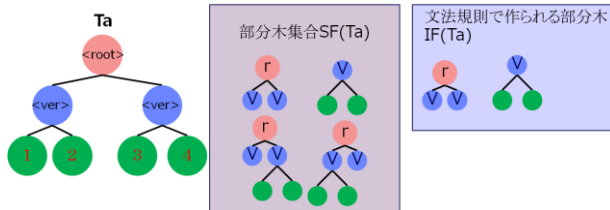


図 12 部分木集合 $SF(T_a)$ と文法規則による部分木集合 $IF(T_a)$

4.1.2 分解

次にコマ割り木 T_a へと変形させるコマ割り木 T_b の分解操作を行う。操作部分木集合 $CF(T_a, T_b, i)$ の 1 つの部分木 $\beta_b \in CF(T_a, T_b, i)$ に対して、その一番上の親を削除し、新しい部分 $\beta_{b,n+1}, \beta_{b,n+2}$ を作成し追加する。これは式 (1) で表される。

$$CF(T_a, T_b, i+1) = CF(T_a, T_b, i) \cup \{\beta_{b,n+1}, \beta_{b,n+2}\} - \{\beta_{b,1}\} \quad (1)$$

図 13 は 1 回目の分解操作にあたる分割例で、1 つの部

分木が一番上の親の削除によって分割操作が行われ、2 つの部分木が作成されている。尚、終端記号 (図 7 中の緑のノード) は部分木集合には含まれない。

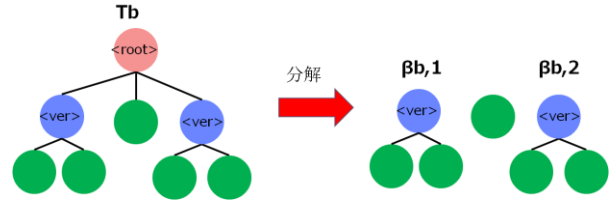


図 13 コマ割り木 T_b の分解

4.1.3 文法規則部分木と合成

部分木を分解して得られた操作部分木集合 $CF(T_a, T_b, i)$ 内の部分木 β_b と、コマ割り木 T_a の文法規則部分木集合 $IF(T_a)$ 内の部分木 α_a^1 を親になるようにし、 β_b がその左側の子になるように合成して得られる部分木を α' 、 β_b が右側へなるように合成されて得られる部分木を α'' とする。この α' α'' のうち T_a の部分木集合 $SF(T_a)$ に含まれているものを $CF(T_a, T_b, i+1)$ に加える。図 14 は合成操作の様子を表したものである。図 14 の例では α' 、 α'' とともに $SF(T_a)$ に含まれているので、両方とも $CF(T_a, T_b, 2)$ に格納されることとなる。式で表すと以下の式 (2) ~ 式 (4) である。

$$\alpha'_a = \alpha_a^1 \circ (\beta_b, \phi), \alpha''_a = \alpha_a^1 \circ (\phi, \beta_b) \quad (2)$$

$$\alpha_a^1 \in IF(T_a), \beta_b \in CF(T_a, T_b, i) \quad (3)$$

$$CF(T_a, T_b, i+1) = CF(T_a, T_b, i) \cup \{\alpha'_a, \alpha''_a\} \quad (4)$$

ここで $\alpha \circ (\beta, \gamma)$ は α を親部分、 β を左の部分木、 γ を右の部分木となるように合成した木を表し、 $\circ$ は合成記号を表す。

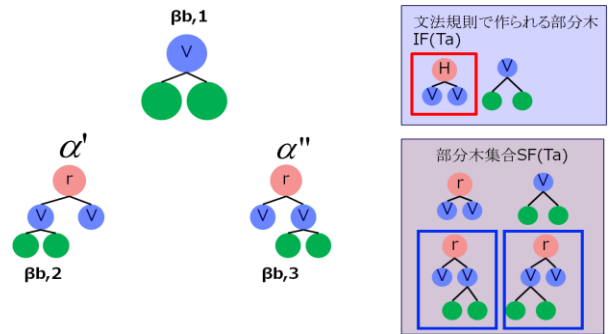


図 14 文法規則部分木との合成操作

4.1.4 部分木集合からの合成

操作部分木集合 $CF(T_a, T_b, i)$ 内の部分木 β_b と、コマ割り木 T_a の文法規則部分木集合 $IF(T_a)$ 内の部分木 α_a^1 、コマ割り木 T_a の部分木集合 $\alpha_a \in SF(T_a)$ の 3 つを合成する。

α_a^1 を親とし、子として右側に β_b を繋げ左側に α_a を繋げて得られる部分木を α' とする。また、 α_a^1 を親とし、子として右側に α_a を繋げ左側に β_b を繋げて得られる部分木を α'' とする。 α' 、 α'' は $SF(T_a)$ に属していないか T_a と一致しない場合も操作部分木集合に追加しない。図 15 は部分木集合との合成操作の例である。 $\beta_{b,1}$ と α_a 、 α_a^1 とを合成し得られた部分木が T_a と一致するのでここで編集を終えることになる。式で表すと式 (5)～式 (7) である。

$$\alpha'_a = \alpha_a^1 \circ (\beta_b, \alpha_a), \alpha''_a = \alpha_a^1 \circ (\alpha_a, \beta_b) \quad (5)$$

$$\alpha_a^1 \in IF(T_a), \beta_b \in CF(T_a, T_b, i), \alpha_a \in SF(T_a) \quad (6)$$

$$CF(T_a, T_b, i+1) = CF(T_a, T_b, i) \cup \{\alpha'_a, \alpha''_a\} \quad (7)$$

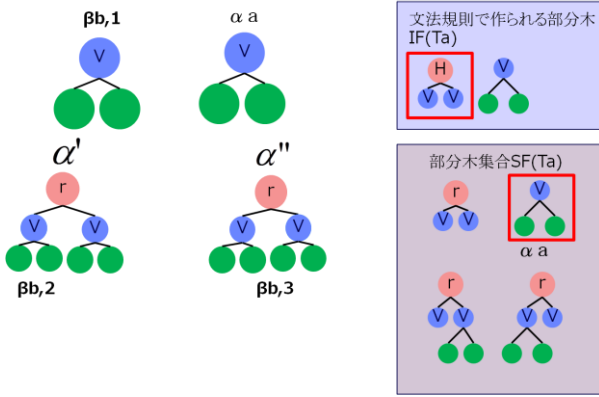


図 15 部分木集合との合成操作

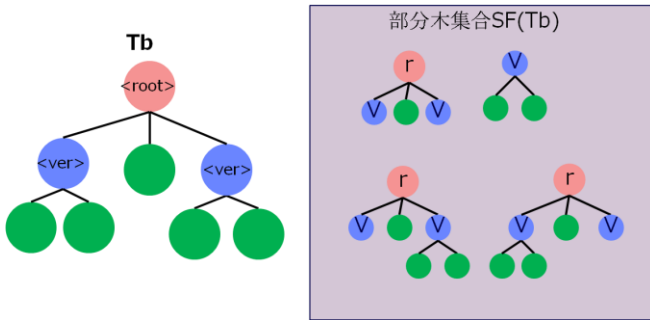


図 16 コマ割り木 Tb の部分木集合 SF(Tb)

4.1.5 編集距離類似度

操作回数を $com(T_a, T_b)$ とし、編集距離類似度 $sim(T_a, T_b)$ を以下のように定義する。

$$sim(T_a, T_b) = \frac{(N_a - com(T_a, T_b)) + (N_b - com(T_a, T_b))}{N_a + N_b} \quad (8)$$

ここで、 N_a 、 N_b はそれぞれコマ割り木 T_a 、 T_b の部分木の数であり、 $sim(T_a, T_b)$ が取る値の範囲は $[0, 1]$ である。今回の例では図 13 の分解操作と図 14 の合成操作を

行っているので $com(T_a, T_b) = 2$ であり、また、 $N_a = 4$ 、図 16 より $N_b = 4$ であるので $sim(T_a, T_b) = 0.5$ となる。

4.2 評価

評価はユーザーの好みによる主観評価値 $sub_fitness$ と、コマ割りの多様性を考慮した客観評価値 $obj_fitness$ の計算を行いその合計値をその漫画の評価値 $Fitness$ とする。

主観評価はユーザーに選んでもらった最も好ましい best 漫画と最も好ましくない worst 漫画の 2 つを比較対象とし、全ての生成された漫画を先述の類似度計算法を用いて類似度を計算する。評価値である $Fitness$ は best 漫画に類似しているほど良く、worst 漫画に類似しているほど良くない漫画だと言えるので主観的评价値 $sub_fitness$ の計算式は次のようになる。

$similarity_1(best, comic, n)$ 、 $similarity_1(worst, comic, n)$ は best 漫画と比較したい漫画 $comic$ のそれぞれの n ページ目同士を比較、および worst 漫画と比較したい漫画 $comic$ の n ページ目同士を比較しているという意味になっており、 $page$ は出力する漫画のページ数となっている。また、 $Fitness$ の計算に $sub_fitness$ を用いるが取りうる値は $[-1, 1]$ になるので、 $\max$ 関数を使い非負になるようにした。よって、取りうる値は $[0, 1]$ である。

$$sub_fitness = \max \left\{ \begin{array}{l} 0, \\ \frac{1}{page} \left(\sum_{k=1}^{page} similarity_1(best, comic, k) - \sum_{l=1}^{page} similarity_1(worst, comic, l) \right) \end{array} \right\} \quad (9)$$

次に客観評価を行う。客観的に見て漫画内に同じコマ割りが複数見られると、コマ割りの多様性から考えるに面白くない漫画といえると筆者は考えた。よって、先述の類似度計算方法を漫画のページごとに用いることで漫画内のコマ割りの重複具合を調査する。重複具合の調査は際限なく出来てしまうが、筆者の経験より直近 3 ページくらいまでならコマ割りを覚えていられるので、基準となる n ページ目から 3 ページ先までの重複具合を調査することにした。

同じようなコマ割りのページが連続すると良くないので客観的评价値 $obj_fitness$ の計算式は次のようになる。ここで $similarity_2(n, n+1)$ は、 n ページ目と $n+1$ ページ目を比較すると言う意味である。また、3 ページ先までページが存在しない場合はページが存在しているところまでで比較計算を終わる。

$$obj_fitness = \frac{1}{\max(page-2, 1)} \sum_{k=1}^{page-1} \sum_{l=1}^3 \frac{similarity_2(k, k+l)}{\min(page-l, 3)} \quad (10)$$

上の式は *Fitness* の計算に用いるので *obj_fitness* が取りうる値は[0, 1]になるように正規化している.

漫画の評価値 *Fitness* は best 漫画に似ていてかつ漫画内のコマ割りがあまり重複していないと良いと筆者は考えるので次のようになる.

$$\text{Fitness} = \text{sub_fitness} - \text{obj_fitness} \quad (11)$$

導出された評価値から学習を行い, ある程度個体が学習した段階で再度 5 つの漫画を選びユーザーによる漫画評価を行う. そこで評価値の更新を行なっている.

5 まとめ

本研究では GE の文法を使いコマ割り木を作成し, それをもとにコマ割りを作成. ユーザーの評価を進化時の評価値に用いてユーザの好みをコマ割りに反映させるようにした. 今後は被験者実験を行い, 本研究の目的であるユーザ好みのコマ割りが生成されているかを検証する. 検証の際に, 今回は客観評価として 3 ページ先までのコマ割りと比較しているが, この 3 ページという設定が正しいのかも検討したい. 予備実験を行い, 人間がコマ割りを覚えていられるのは直近何ページなのかを調査しておく必要があると考えられる.

また, AttentionCueing というキャラクター間やフキダシ間の距離を使って, 読者が感じるスピード感を調整する技法を取り入れれば更なる効果があると考えている.

文 献

[1] Ariel Shamir, Michael Rubinstein, Tomer Levinboim. Parametric Comics Creation from 3D Interaction. IEEE Transactions on Computers, Vol. C-35, No. 8, pp. 677-691, August 1986.

[2] 首田大仁, Ruck Thawonmas. オンラインゲームのプレイログを用いた漫画の自動生成

立命館大学大学院理工学研究科ゲーム学会誌 2008 Vol. 3 No. 1 p 41-p46.

[3] Hiroaki Tobita. Comic Computing: Creation and Communication with Comic User Interface.

SIGDOC'11 29th ACM international conference on Design of communication pp. 91-98, Oct. 3-5, 2011.

[4] Second Life

<http://secondlife.com/?lang=ja-JP>

[5] 坂本 竜基, 角 康之, 中尾 恵子, 間瀬 健二, 国藤 進: "コミックダイアリ: 漫画表現を利用した経験や興味の伝達支援", 情報処理学会論文誌, Vol. 43, No. 12, pp. 3582-3595, 2002.

[6] Ruck Thawonmas, Yoshinori Tani. Frame Selection Using Interactive Grammatical Evolution for Automatic Comic Generation from Game Logs. IEEE CIG2011, pp. 31-38, Aug. 31-Sep. 3, 2011.

[7] Takamasa Tanaka, Kenji Shoji, Fubito Toyama, and Juichi Miyamichi. Layout Analysis of Tree-Structured Scene Frames in Comic Images.

IJCAI'07, pp. 2885-2890, 2007.

[8] 岩沢 博人, 北栄 輔, 名古屋大学大学院. Grammatical Evolution の性能改善及びGPとの性能比較. 情報処理学会研究報告, 2007 号 86(2007-MPS-066), 5 - 8, 2007-09-03.

[9] 編集: えんぴつ倶楽部 監修: 塚本博義

漫画バイブル 5 コマ割り映画技法編 マール社 2007. 4. 20 第1版

[10] 椎名広光, 秋友克俊, 岡山理科大学総合情報学部. 構文解析木の類似度の判定アルゴリズム. 数理解析研究所講究録, Vol. 1426, pp. 26-31, Apr 2005.