有効遺伝的操作を導入した Linear GP の提案

A new effective genetic operation in Linear Genetic Programming

◯ [1] 檜垣　紀志,　[1] 原　章,　[2] 市村　匠,　[1] 高濱　徹行
◯ [1]Noriyuki Higaki,　[1]Akira Hara,　[2]Takumi Ichimura,　[1]Tetsuyuki Takahama
[1] 広島市立大学大学院　情報科学研究科
[1]Graduate, School of Information Science, Hiroshima City University
[2] 県立広島大学　経営情報学部経営情報学科
[2]Faculty of Managemeent and Information Science, Prefectural University of Hiroshima

Abstract: Genetic programming (GP) is one of the techniques for evolutionary optimization of programs. Linear GP (LGP) has been proposed as an extended GP. Individuals, whose structure consists of a linear array, are represented by a sequence of low-level language such as an instruction code. In LGP, a fitness value is calculated by processing codes serially from head position and changing contents of the register. The instructions that individual has are referred to as an effective code only if the result of its calculation makes an impact on the fitness value, otherwise a non-effective code. It is possible to change the representation of individuals effectively by applying genetic operations to effective codes. In this paper, to seek a better solution within lower generations by intensification of search, we proposed a new genetic operation that inserts the effective piece of codes existing in better individuals to the other individuals. We performed an experiment for comparing standard LGP and LGP with the proposed genetic operation using benchmark problem. The result indicated the proposal method is useful.

1　はじめに

遺伝的プログラミング (Genetic Programming：GP)[1] は進化的計算の内の一つであり，遺伝的アルゴリズム (Genetic Algorithm：GA)[2] を拡張した手法である．GP では個体の表現に木構造を用いているため，ポインタ操作が必要となり時間計算量および空間計算量が増大してしまう．それを解決する一つの手法として Linear GP(LGP)[3][4] が提案されている．LGP では，個体に線形配列が用いられており，機械語の様な命令コード（以下コード）が列をなすことにより表現される．このコードを先頭から逐次処理することで，個体の出力を得る．個体中のコードには，その演算結果が最終的な出力に影響を与えるような有効コードと，そうでない無効コードが存在する．この有効コードを操作することで，個体の出力に変化をもたらすことが可能となり個体の進化が速くなる．しかしながら，単に有効コードを操作するだけでは効果的に進化が行われないと考えられる．本研究では従来の遺伝的操作に加え，新たな有効遺伝的操作として，上位のエリート個体中に含まれる有効部分コードを集団内の個体に挿入する突然変異を提案する．この操作により，エリート個体に含まれる部分を他の個体に挿入することで，最適解探索の集中化を図り，早期世代における優良解の発見を期待している．従来手法と提案手法で実験を行い，その性能を比較することで提案手法の有用性を示す．

2　Linear　GP

GP の個体の遺伝子型は木構造によって表現されている．これにより，個体の評価および遺伝的操作時にポインタ操作が必要となり，また個体の成長に伴い木が大きくなることで計算量が増大する．この計算量を減少させるために，Linear GP（LGP）と呼ばれる手法が提案された．

2.1　LGP の個体表現

LGP での個体には線形配列が用いられ，命令コード（以下コード）が複数連なることで個体が表現される．各コードは，低級言語で使用されるレジスタ演算を模倣した簡単な演算式で表される．コードは 4 つの要素から構成され，第一要素が演算子，第二要素が第一オペランド，第三要素が第二オペランド，第四要素が目的レジスタ（演算結果を格納するレジスタ）を表している．遺伝子型の例を図 1 に，またその遺伝子型に対応する表現型を図 2 に示す．図 1,2 の個体において，レジスタ r[a] を入力および出力レジスタとすると，この個体は次の二次関数を示す．

$$f(x) = (x+1)^2 - 2$$

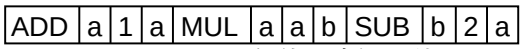

図 1: LGP の個体の遺伝子型

```
1:  r[a] = r[a] + 1;
2:  r[b] = r[a] * r[a];
3:  r[a] = r[b] - 2;
```
図 2: LGP の個体の表現型

2.2 LGP の有効コード・無効コード

LGP の特徴として，個体の表現に影響を与える有効コードと，そうでない無効コードが存在する．図 3 のようなコードがあった場合を考える．2 番目および 4 番目のコードでレジスタ r[c] に演算結果が格納されるが，それ以降の命令で使用されていないことが分かる．この場合，2，4 番目のコードは，出力に影響を与えてないという意味で無効コードと呼ぶ．それとは対照的に 1，3，5 番目のコードは個体の出力に影響を与えるため，有効コードと呼ばれる．また，有効コードに含まれるレジスタは有効レジスタと呼ばれる．図 3 における 3 番目のコード位置での有効レジスタは r[a] と r[b] である．一見，無効コードは必要のないコードのように思えるが，個体プログラムの有効な部分における破壊的な変化を減らす効果がある．これにより，交叉等の遺伝的操作による有効部分の分断・破壊を防ぐ．

```
1:  r[a] = r[a] + 1;
2:  r[c] = r[a] - 8;
3:  r[b] = r[a] * r[b];
4:  r[c] = r[a] / r[b];
5:  r[a] = r[b] - 2;
```
図 3: LGP の個体の表現型

2.3 LGP の遺伝的操作

LGP の遺伝的操作は，GP や GA 同様に交叉や突然変異等があり，以下にそれらを示す．

2.3.1 交叉

LGP の交叉は 1 点交叉または 2 点交叉などがある．本研究では二点交叉を扱っている．親となる 2 個体で各々交叉点をランダムに 2 点選び，2 点間に存在するコードをお互いに交換し子個体を 2 つ生成する．

2.3.2 突然変異

LGP の突然変異は，大きく分けるとマクロ突然変異とミクロ突然変異に分かれる．マクロ突然変異は個体に 1 つのコードを挿入するか，または個体から 1 つのコードの削除を行う．ミクロ突然変異は，あるコードの 1 要素を選択して変化させる．

2.3.3 有効突然変異

LGP の個体は有効コードによって表現される．そこで，LGP の突然変異には，個体が持つ有効コードを操作し個体の表現を必ず変化させる手法として有効突然変異 (effective mutation：effmut)[3] が提案されている．この遺伝的操作により，個体の表現が必ず変化することで進化がより活発的に行われるようになる．

2.4 LGP のアルゴリズム

一般的な LGP のアルゴリズムは図 4 のようになる．有効コードの探索アルゴリズム [3] によって，2. で有効コードと無効コードの判別をする．effmut を実装した LGP では，突然変異では必ず effmut を行う．

1. 個体群の初期化
2. 有効コードの探索
3. 各個体の評価
4. 個体の選択
5. 遺伝的操作
   (a) 交叉
   (b) 突然変異
6. 次世代の個体数が現世代の個体数と異なる場合は 4. へ
7. 個体群の置き換え
8. 最大世代数に到達していなければ 2. へ

図 4: LGP のアルゴリズム

3 提案手法

effmut を取り入れることで，通常の突然変異よりも個体の進化が効果的に行われるようになる．しかし，この effmut では適合度の停滞が世代の中盤辺りで見られ，最適解を導けないこともしばしば見られる．またコードが必要以上に成長することで，コード長が最大にまで達し進化が停滞することもあった．

本研究では，有効遺伝的操作として新たに，有効置換突然変異とエリート部分コード挿入（Sub-code of Elite individuals Insertion:SEIns）突然変異を提案する．有効置換突然変異は，削除や挿入とは異なり個体の長さを変えずに行う突然変異であり，任意の 1 コードを変化させる．置換突然変異により，コードの長さを抑えながら個体の進化が可能になると考えられる．SEIns 突然変異は，各世代で選択された 2 つのエリート個体間で類似している有効コードの一部分を抽出し他個体へ挿入する操作である．探索の集中化を計り，早期世代において優良解を発見させることを期待している．

3.1 置換突然変異

LGP では，世代が進むにつれ個体のコードの長さが最大まで達することがある．コードの長さが最大にまで達することで，突然変異でコードの挿入ができずそれ以降の世代で適合度の向上が見られないという問題がある．これを解決するための方法として，個体のコードの長さに依存しない進化法が考えられる．そこで任意の 1 コードを変化させる突然変異を提案し，これを置換突然変異 (replace mutation：rep) と呼ぶ．コードの 1 要素を変化させるマクロ突然変異とは異なり，置換突然変異は 1 度の操作でコードの 4 要素を変化させる．これにより，より大きなステップサイズの進化が期待できる．

さらに，置換突然変異用の有効突然変異を新たに加える．これを有効置換突然変異 (effective replace mutarion：effrep) と呼ぶ．適用したコードが有効になる様に置換操作を施す．操作の処理手順を以下に示す．

(i) 有効置換突然変異を適用するコード位置 i を選択
(ii) コード位置 $i-1$ における有効レジスタ r_{eff} を探索
(iii) コード位置 i の目的レジスタ r_{dest} を r_{eff} に変更し，他の 3 要素はランダムに置換する

3.2 SEIns 突然変異

この操作は，エリート個体中の有効コードを一部分取り出し，他個体へ挿入する．個体の進化を促進させ早期世代で優良解を発見することを期待している．この操作には，(1) 部分コードの抽出，(2) 操作適用条件の設定，(3) レジスタ番号の修正の 3 点が必要とされ，これらを以下に示す．

3.2.1 部分コードの抽出

適合度順にランク付けされた上位 2 つのエリート個体から，コード要素が類似した有効コードの一部を抽出する．このときユーザが定義した値 $efflen$ の長さだけ抽出する．（この章では，説明上 $efflen = 3$ とする）まず，コード間の類似度を計算するために，有効コードのみを取り出した 2 個体 A,B の各コード a_i,b_j 間の距離 $dist_{i,j}$ をハミング距離で求める．つまり，要素の異なる個数を各々のコード間の距離とする．そのときの計算を式 (1) に示す．a_{i_k} と b_{j_k} は，それぞれのコード a_i,b_j の k 番目の要素を意味する．

$$\delta_k(a_i, b_j) = \begin{cases} 0 & a_{i_k} = b_{j_k} \\ 1 & a_{i_k} \neq b_{j_k} \end{cases}$$
$$dist_{i,j} = \sum_{k=1}^{4} \delta_k(a_i, b_j) \quad (1)$$

例として，各コード間の距離が表 1 のようになったとする．表 1 のコード間の距離を元に，部分コード間の距離 $DIST_{i,j}$ を式 (2) で求める．式 (2) の計算により得られた各部分コード間の距離を表 2 に示す．

$$DIST_{i,j} = \sum_{k=0}^{efflen-1} dist_{i+k,j+k} \quad (2)$$

表 1: 各コード間の距離

$dist_{i,j}$	b_1	b_2	b_3	b_4	b_5	b_6	b_7
a_1	1	1	2	3	4	3	2
a_2	1	0	1	2	3	4	3
a_3	2	1	0	1	2	3	4
a_4	3	2	1	0	1	2	3
a_5	4	3	2	1	1	2	3
a_6	3	4	3	2	2	2	3

表 2: 各部分コード間の距離

$DIST_{i,j}$	$b_{1\sim3}$	$b_{2\sim4}$	$b_{3\sim5}$	$b_{4\sim6}$	$b_{5\sim7}$
$a_{1\sim3}$	1	3	6	9	12
$a_{2\sim4}$	3	0	3	6	9
$a_{3\sim5}$	6	3	1	4	7
$a_{4\sim6}$	9	6	4	3	6

表 2 の値を探索し，最小となるセル位置を記憶する．探索の過程で最小となるセルの値が複数ある場合は，ランダムに選択する．そして，個体 A,B で適合度が良い方の部分コードを抽出する．例に従うと，ここでは $DIST_{2,2}$ が選ばれ，個体 A の適合度が良い場合は部分コード $a_{2\sim4}$ が抽出される．

3.2.2 操作適用条件の設定

SEIns 突然変異を適用するにあたり，集団内の個体の多様性が低下する問題が考えられる．よって，以下の条件を満たした場合に限り，この操作を行えるものとする．これにより，部分コードの過度の挿入を抑え，同じコードが集団内に蔓延するのを防ぐ．

(i) 個体が進化した世代に適用
個体が進化したかは前世代と現在の世代の最良個体の適合度を比較し，向上したかどうかで判定する

しかし，(i) を満たした場合であっても，各世代で SEIns 突然変異の適用回数が最大適用回数 (limit) に達していたときはこの操作を行わないものとする．このとき，SEIns 突然変異の代わりに effmut のマクロ突然変異の挿入操作を行う．

3.2.3 レジスタ番号の修正

抽出された部分コードを，挿入される個体の位置に応じて有効コードとなるように変化させる．これにより，SEIns 突然変異で取り出した部分コードが挿入点によって無効化されることなく，個体の出力に影響を与えることが可能となる．ここでは，部分コードの有効化をレジスタ番号の修正によって行う．この操作手順を以下に示す．抽出された部分コードが図 5 のようなコード列であるとする．

```
r[1] = r[2] + 3;
r[3] = r[1] * r[1];
r[4] = r[1] - r[0];
r[1] = r[3] / 5;
r[2] = r[1] + r[4];
```

図 5: 抽出した部分コード (修正前)

まず，SEIns 突然変異を適用する個体の挿入位置の有効レジスタ r_{eff} を探索する．抽出された部分コードの最後尾に現れるコードの目的レジスタ r_{end} を有効レジスタ r_{eff} に揃える．図 5 から r_{end} は 2 であり，また r_{eff} の方は説明上 4 と仮定する．この場合，r_{end} の値を 2 増加させることで，r_{eff} と同じ値 4 にすることになる．r_{end} の増加値が 2 と同じように，他のレジスタ番号も 2 増加させる．

レジスタ番号は循環していると見なし，レジスタ番号の増加によって最大のレジスタ番号を越える場合はレジスタ番号を 0 から数えるようにする．最大レジスタ番号が 4 であった場合，0→1→2→3→4→0→1→$\cdots$ となり，レジスタ番号 4 を 2 増加させると，1 となる．この処理により，部分コードのサブルーチンを崩さないまま挿入時に部分コードの有効化が可能であると考えられる．$r_{end} = 2$,$r_{eff} = 4$ の場合で，部分コードを修正した結果は図 6 のような部分コードになる．

```
r[3] = r[4] + 3;
r[0] = r[3] * r[3];
r[1] = r[3] - r[2];
r[3] = r[0] / 5;
r[4] = r[3] + r[1];
```

図 6: 抽出した部分コード (修正後)

この操作により，有効化された部分コードによって個体の適合度に変化を与える．また，挿入点の位置に応じて部分コード内のレジスタ番号が変化するため，同一な部分コードを挿入することによる多様性の低下を防ぐことができると考えられる．

4 実験

従来手法と提案手法における性能を比較するために，関数近似問題の実験を行った．*sinepoly* 問題のパラメータを表 3 に示す．

表 3: *sinepoly* 問題のパラメータ

目的関数	$sin(x) \times x + 5$
定義域	[-5,5]
値域	[0,7)
入出力レジスタ	r[0]
計算用レジスタ数	r[1],$\cdots$,r[4]
総レジスタ数	5
訓練データ数	101
演算子種類	$\{+, -, \times, /, x^y\}$
定数	$\{1, 2, ..., 9\}$
適合度関数	SSE

4.1 置換突然変異の実験

従来の LGP(normal) と置換突然変異を導入した LGP(rep-LGP) の比較実験を行い，この操作が最適解の探索に与える効果を確認した．

実験で使用した各種パラメータを表 4 に示す．交叉率は従来手法で比較的性能の良かった 0.5，0.6，0.7 の値を用いた．最終的世代における適合度，標準偏差の値を 30 回試行の平均で求めた．実験の結果を表 5 に示す．

表 4: LGP パラメータ

共通パラメータ	
世代数および個体数	1000
最大/最小コード数	200/10
初期コード数	10
トーナメントサイズ	8
交叉率	Pc
突然変異率	1 - Pc
マクロ突然変異率	0.4
挿入：削除 (置換無)	0.67：0.33
挿入：削除：置換率（置換有）	0.6：0.3：0.1
ミクロ突然変異率	0.6

表 5: 実験結果（適合度と標準偏差）

	fitness		*std.*	
Pc	normal	rep-LGP	normal	rep-LGP
0.5	9.22012	**9.17093**	12.2403	**8.5762**
0.6	13.5596	**7.55694**	11.8	**10.2053**
0.7	10.3717	**6.88562**	19.4932	**11.1542**

4.2 置換突然変異の考察

置換突然変異を導入することで，適合度の値と標準偏差の値はともに良い結果を示した．

実際にコードの長さがどの様に変化したかを比較してみたところ，置換突然変異を適用した方が最終的な世代で平均 10 コード程度短くなっていた．有効コードに関しても平均 5 コード程短くなっており，若干コンパクトに解を表現できるようになった．コードを長くすることで個体を進化させるよりも，コードの成長を抑えながら進化させる方が効果的であると考えられる．

また，今回の置換突然変異率は挿入や削除と比べ，1 割程度の適用率で比較的小さい値に設定しているが，さらにこの値を調整することでより良い結果を導けると考えられる．

4.3 SEIns 突然変異の実験

SEIns 突然変異を導入したことで，最適解の探索にどう影響を与えたかを確認するために比較実験を行った．比較対象は，前節の置換突然変異を実装した rep-LGP と，従来 LGP に effmut と有効置換突然変異を実装した LGP(effmut+effrep) である．

また，SEIns 突然変異を導入した LGP ではレジスタ番号の修正の操作を行う場合 (seins) と，そうでない場合 (seins-old) に分けて実験を行う．これらの比較により，レジスタ番号の修正が最適解の探索に効果的な影響を与えているかを検証する．SEIns 突然変異で設定したパラメータを表 6 に示す．

前節の実験と同様に，LGP のパラメータは表 4 のものを使用する．最終世代における適合度と標準偏差の値を，30 回試行の平均で求め，それらを表 7 に示す．それぞれの実験で，最も良い結果において得られた適合度の値を世代毎にプロットしたものを，図 7 に示す．

表 6: SEIns 突然変異のパラメータ

抽出する部分コード数 (efflen)	5
各世代の適用回数の上限 (limit)	10

4.3.1 有効突然変異の考察

図 7 より，effmut+effrep と rep-LGP と比べると，effmut+effrep は収束速度が非常に速く，個体の進化が活発に行われているのが分かる．しかし，表 7 から標準偏差の値が大きく，結果にばらつきがあることが伺える．原因の一つに，有効突然変異内の削除操作が，任意の有効コードを削除するという操作であることが考えられる．有効コードの数を比較したところ，effmut+effrep は有効コードの数が 10 程度少なくなっていた．このことから，個体に必要な有効コードが失われた可能性があり，適合度が下がったと思われる．effmut+effrep では，有効コードの破壊を軽減するためにパラメータ調整を行い，削除率を今よりも減少させることで，より効果的な進化が期待できると思われる．

表 7: 実験結果（適合度と標準偏差）

fitness				
Pc	rep-LGP	effmut+effrep	seins-old	seins
0.5	9.17093	6.07709	5.8346	**5.27628**
0.6	7.55694	10.9474	**3.64494**	4.58383
0.7	6.88562	7.12468	4.51407	**2.28827**
std.				
Pc	rep-LGP	effmut+effrep	seins-old	seins
0.5	8.5762	9.2926	10.0051	**6.7925**
0.6	10.2053	24.7411	7.04234	**5.73838**
0.7	11.1542	12.4044	6.31703	**1.99528**

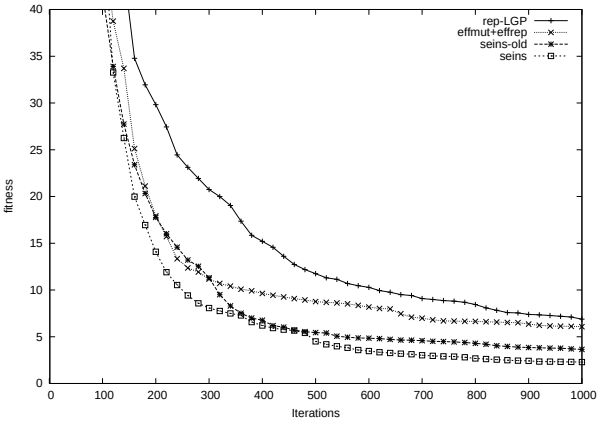

図 7: 世代数に対する適合度の推移

4.3.2 SEIns 突然変異の考察

まず，seins-old について考察する．表 7 より，seins-old での実験は rep-LGP や effmut+effrep よりも良い結果を示した．適合度は全体的に向上し，標準偏差の値も他実験と比較的には小さくなったが，突然変異率が高くなるにつれ，標準偏差の値が大きくなる傾向が見られる．交叉の割合が小さくなることにより SEIns 突然変異が行われる可能性が高くなることで，抽出された部分コードが集団内に蔓延する可能性も高くなる．よって，集団内の多様性低下による早期収束が起こりやすくなり，このような傾向が現れたと考えられる．

図 7 より，seins-old では 200 世代を越えたあたりから 300 世代までは進化速度が若干緩やかになった．この原因として，同一の部分コードの挿入によるものが

大きく影響していると思われる．seins-old ではレジスタ番号の修正が行われないため，同一の部分コードが複数の個体に挿入されてしまう．その結果，集団内の多様性が低下し進化速度が減少したと考えられる．

しかし，その直後の世代（300 世代辺り）で進化速度が増加しているのが分かる．SEIns 突然変異が適用された回数を計測したところ，この 300 世代付近では減少していた．以上のことから，突然変異が起こりやすくなることで，コードの要素が変化し集団内に多様性がもたらされ，再度個体が進化しだしたと考えられる．

次に seins について考察する．この操作で目標としていた早期世代で優良解を発見することに関しては，図 7 より effmut+effrep や seins-old と比べ若干速くなっており，期待通りの結果が得られたと言える．

また，seins は今回の実験の中で最終的な世代の適合度が良く，標準偏差の値も小さいことから，安定した結果が得られたと言える．レジスタ番号の修正を行うことにより，seins-old では早期に進化の速度が減少していたものが解消され，最終世代まで進化し続けているのが分かる．交叉率が 0.7 においては最も良い性能を示し，標準偏差の値が 2 弱と小さいことからこのパラメータにおける実験では早期収束に陥った回数は少ないと考えられる．同じサブルーチンを持った部分コードでも，挿入点に応じてレジスタ番号を随時変化させることで，多様性の減少を抑えることが可能になったと考えられる．

以上のことから，seins-old では従来の LGP よりも最適解に近い解を導くことができ，性能の向上が確認できた．さらに seins では，収束速度が若干向上し，レジスタ番号の修正を施すことにより最適解に近い解を安定して導くことができ，本研究で行った実験中で最も良い性能を示すことを確認した．

4.4 他の問題への適用

関数同定問題以外にクラス分類問題やパリティ問題でも seins と effmut+effrep の比較実験を行った．クラス分類問題は *iris* 問題を使用し，150 事例のクラス分けを行う．この問題では effmut+effrep でほぼ 100 %に近い分類ができており，seins での性能の向上は見られたもののその差はあまり見られなかった．パリティ問題は *even8parity* 問題を使用した．こちらの問題では seins を適用することで，逆に収束速度が遅くなり，その性能も悪くなった．

このことから，パリティ問題には今回設定した seins のパラメータが不向きであったと考えられる．

5 おわりに

本論文では，有効遺伝的操作として置換突然変異および SEIns 突然変異を提案し，比較実験によってその性能を検証した．

置換突然変異を導入した LGP では，解をよりコンパクトに表現でき，従来の LGP を上回る性能を示した．

SEIns 突然変異を導入した LGP では個体の適合度が収束する速度が若干速くなり，最終的な適合度の値も比較的良い値を得ることができた．さらに，抽出した部分コードのレジスタ番号に修正を施すことで，より安定した結果を得ることができた．

しかし，次の 2 点において改良が必要であると考えられる．

- SEIns 突然変異だけでなく，交叉などでもレジスタ番号の修正を行うこと
- コード間の類似度計算としてハミング距離以外の距離関数を用いること

前者により，お互いの部分的なコードを交換した後も有効コードとして存在し，より効果の大きい操作が期待できる．後者では，ハミング距離以外の距離関数・評価尺度を用いることで，より効果的な部分コードを抽出する方法も有効だと考えられる．また，集団内の多様性を維持する上でも，個体間の距離を正確に測定する評価基準の確立が必要である．

参考文献

[1] John R.koza, *Genetic Programming: On the Programming of Computers Means of Natural selectio*, MIT Press, 1992

[2] Goldberg, David E, *GENETIC ALGORITHMS in Search, Optimization , and Machine Learning*, Addison-Wesly inc., 1989

[3] Marlus Brameier and Wolfgang Banzhaf, *Linear Genetic Programming*, Springer Science+Business Media, 2007

[4] C.L.Alonso, J.Puente, J.L.Montaña, *Straight Line Programs:A new Linear Genetic Programming Approach*, 20th IEEE International Conference on Tool with Artificial Intelligence, 2008

連絡先

〒 731-3194
広島市安佐南区大塚東 3-4-1
広島市立大学大学院 情報科学研究科 知能工学専攻
高濱研究室
檜垣　紀志
E-mail: nhigaki@ints.info.hiroshima-cu.ac.jp