GPの実装方法比較

藤井 陽介(touyou)

GP(遺伝的プログラミング)

地味にいろいろ種類がある

今回比較するのは**個体の表し方が違う**３つのGP

- Tree-Based GP
- Linear GP
- Graph-Based GP

とりあえずそれぞれの簡単な紹介

Tree-Based GP

一番**一般的**なやつ

なんか**一部の**言語では実装がめちゃくちゃ簡単らしい

木構造を個体として扱う

C/C++で扱うポインタの量が多すぎて僕みたいな**にわかC++勢が死ぬ**

Linear GP

線形リストを個体として扱う

時間・空間計算量を減らすために生まれたらしい

ポインタも必要ないのでやさしそうに見える、見えるだけ

Graph-Based GP

有向グラフを個体として扱う

なんかある論文ではGNP(Genetic Network Programming)と呼ばれていた

有向グラフなのでノードが再利用できたりできなかったりする

個体の構築難しそう

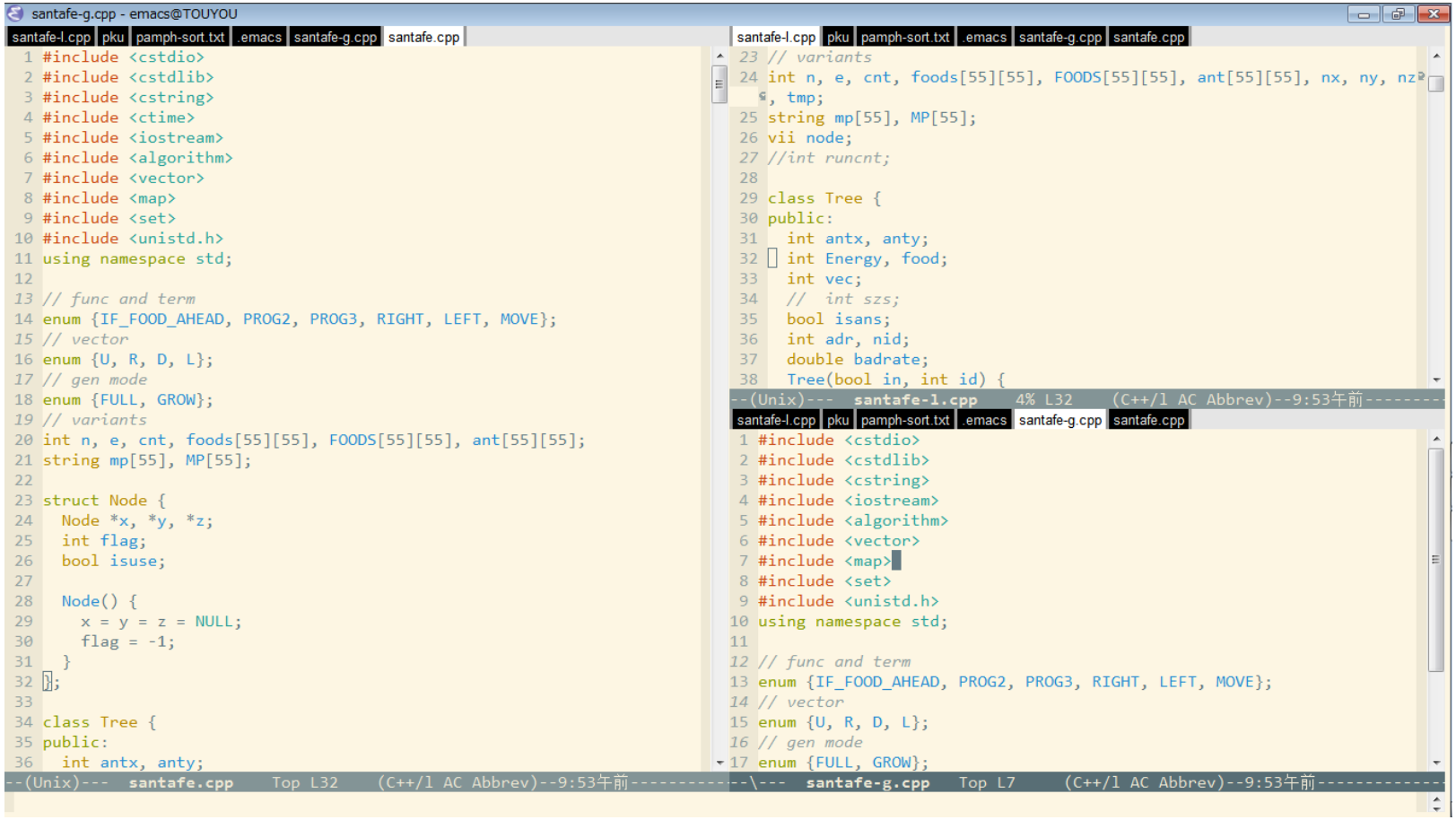
Santa Fe Trail問題

実際に実装してみよう

でもTree-Based GPでさえも
実装つまづいてるんだよね…

ふえええ

ごめんなさい頑張ります


santafe-g.cpp - emacs@TOUYOU
santafe-l.cpp pku pamph-sort.txt .emacs santafe-g.cpp santafe.cpp
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <iostream>
#include <algorithm>
#include <vector>
#include <map>
#include <set>
#include <unistd.h>
using namespace std;
// func and term
enum {IF_FOOD_AHEAD, PROG2, PROG3, RIGHT, LEFT, MOVE};
// vector
enum {U, R, D, L};
// gen mode
enum {FULL, GROW};
// variants
int n, e, cnt, foods[55][55], FOODS[55][55], ant[55][55];
string mp[55], MP[55];
struct Node {
Node *x, *y, *z;
int flag;
bool isuse;
Node() {
x = y = z = NULL;
flag = -1;
}
};
class Tree {
public:
int antx, anty;
--(Unix)--- santafe.cpp Top L32 (C++/l AC Abbrev)--9:53午前
santafe-l.cpp pku pamph-sort.txt .emacs santafe-g.cpp santafe.cpp
// variants
int n, e, cnt, foods[55][55], FOODS[55][55], ant[55][55], nx, ny, nz
, tmp;
string mp[55], MP[55];
vii node;
//int runcnt;
class Tree {
public:
int antx, anty;
int Energy, food;
int vec;
// int szs;
bool isans;
int adr, nid;
double badrate;
Tree(bool in, int id) {
--(Unix)--- santafe-1.cpp 4% L32 (C++/l AC Abbrev)--9:53午前
santafe-l.cpp pku pamph-sort.txt .emacs santafe-g.cpp santafe.cpp
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <algorithm>
#include <vector>
#include <map>
#include <set>
#include <unistd.h>
using namespace std;
// func and term
enum {IF_FOOD_AHEAD, PROG2, PROG3, RIGHT, LEFT, MOVE};
// vector
enum {U, R, D, L};
// gen mode
enum {FULL, GROW};
--\--- santafe-g.cpp Top L7 (C++/l AC Abbrev)--9:53午前

かたかた…

結果

GP失敗談

人間 乃屑

Tree-Based GP

- 一応動いた
- ろくに成長しない
- というか進化と退化を一定確率で繰り返している…
- ~~もしかしたらこっちのほうが本物の遺伝っぽいかも~~

Linear GP

- 同じ手を動かした
-
-
-

じゃなくて…

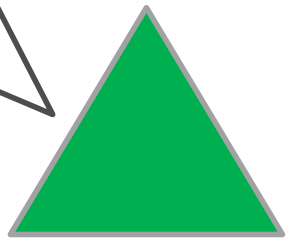
盛大にバグる

Graph-Based GP

- 実装一番難しそうなので後にまわしてた
- Tree-BasedとLinearをまともに仕上げようと頑張ってたら時間なくなった

ようするに…

GP便利なんだけど
微妙にバグる



あぉぉぉ

(´・ω・`)

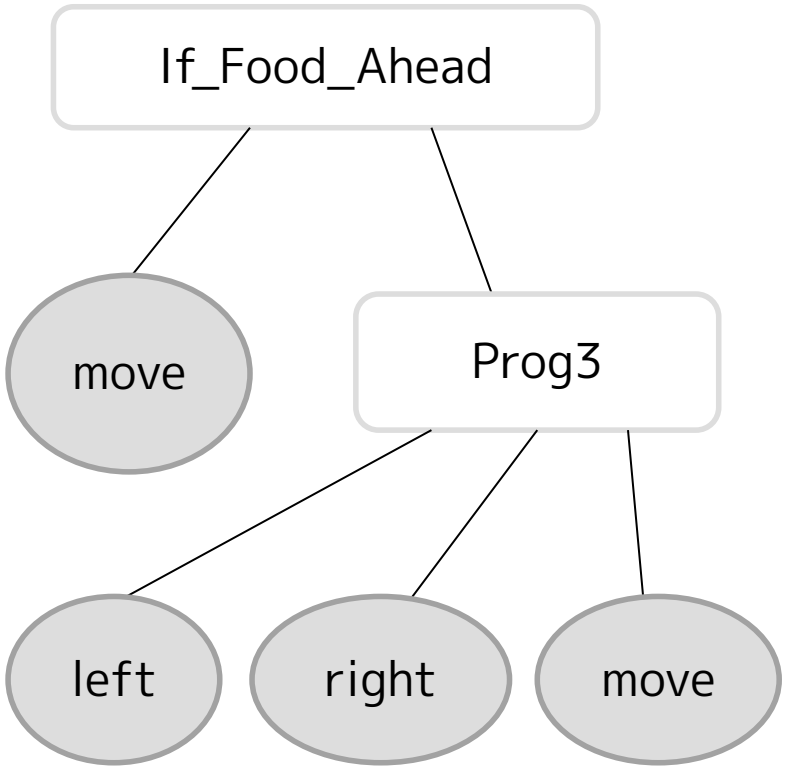
なんか本当にごめんなさい

GPの各実装所感

とうよう

Tree-Based GP

- 右のような木を個体として扱う（Santa Fe Trailの場合）
- 先程も書いたようにC/C++のポインタの扱いに慣れてないと死ぬ
- でも一番実装は楽。文献が多い。



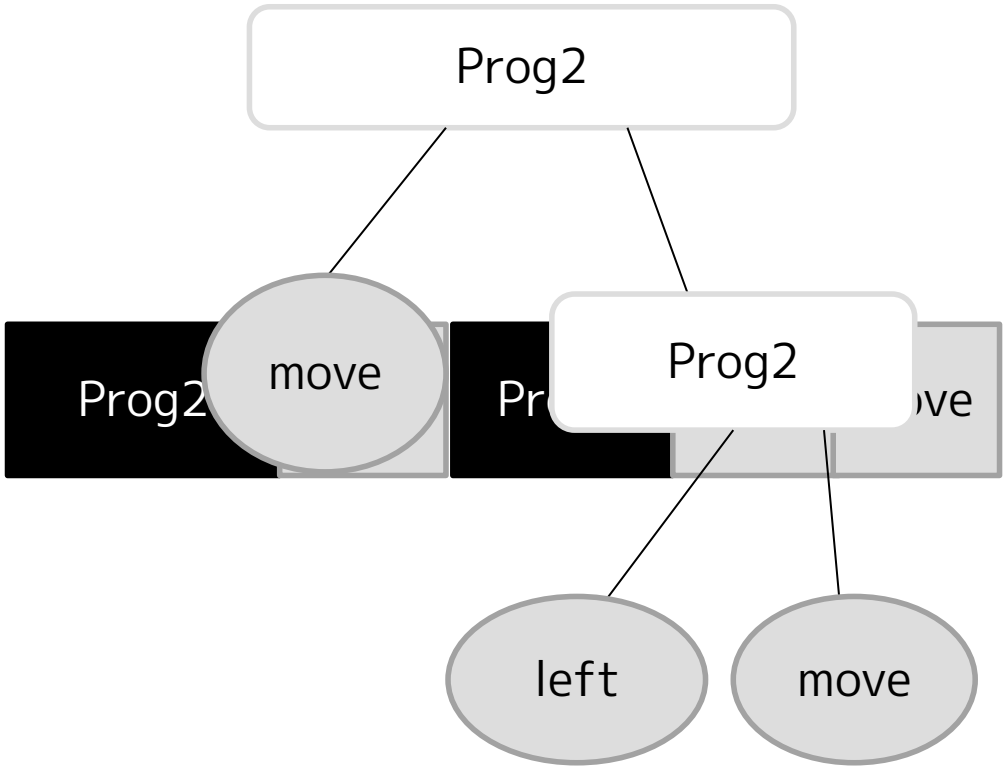
一応実行結果

```
result: food=9 Energy=87 evaluation=853.420
SZ=200 GENE=100 TSZ=5 MUTE=0.050 RESERVE=0.100
result: food=8 Energy=134 evaluation=756.880
SZ=200 GENE=100 TSZ=5 MUTE=0.050 RESERVE=0.100
result: food=5 Energy=300 evaluation=481.770
SZ=200 GENE=100 TSZ=5 MUTE=0.050 RESERVE=0.100
result: food=8 Energy=120 evaluation=767.600
SZ=100 GEN
result: fo
SZ=100 GEN
result: fo
SZ=100 GEN
result: food=12 Energy=130 evaluation=11811.250
SZ=500 GENE=100 TSZ=5 MUTE=0.090 RESERVE=0.100
result: food=5 Energy=150 evaluation=4957.130
SZ=500 GENE=10000 TSZ=5 MUTE=0.010 RESERVE=0.100
result: food=12 Energy=85 evaluation=774.200
SZ=500 GENE=100 TSZ=5 MUTE=0.090 RESERVE=0.100
result: food=8 Energy=250 evaluation=601.100
SZ=500 GENE=100 TSZ=5 MUTE=0.090 RESERVE=0.100
```

Food=89が目標なので全部ダメ

Linear GP

- 一見楽。
- メモリとかを気にしなければ、実装も楽だしポインタでごちゃごちゃしたものよりもまともに動く
- とりあえず右図みたいな個体を持つ



この木に対応する

結果はさっき言った通り
セグフォループ

どうしてこうなった＼(^o^)／


どうしてこうなった…
どうしてこうなった！？
@qnighy Masaki Hara
8/27
@touyoubuntu 日頃の行い
ど…！

だからみなさんは大丈夫だよ
（無責任）

普通の人でも気をつけるべき注意事項

```
==25641== Process terminating with default action of signal 11 (SIGSEGV)
==25641==  Access not within mapped region at address 0xBE601FFC
==25641==    at 0x804A15C: std::_Bit_iterator::_Bit_iterator(unsigned long*, unsigned int) (stl_bvector.h:196)
==25641==  If you believe this happened as a result of a stack
==25641==  overflow in your program's main thread (unlikely but
==25641==  possible), you can try to increase the size of the
==25641==  main thread stack using the --main-stacksize= flag.
==25641==  The main thread stack size used in this run was 8388608.
==25641== Stack overflow in thread 1: can't grow stack to 0xbe601ff8
==25641==
==25641== Process terminating with default action of signal 11 (SIGSEGV)
==25641==  Access not within mapped region at address 0xBE601FF8
==25641==    at 0x4025430: _vgnU_freeres (in /usr/lib/valgrind/vgpreload_core-x86-linux.so)
==25641==  If you believe this happened as a result of a stack
==25641==  overflow in your program's main thread (unlikely but
==25641==  possible), you can try to increase the size of the
==25641==  main thread stack using the --main-stacksize= flag.
==25641==  The main thread stack size used in this run was 8388608.
==25641==
==25641== HEAP SUMMARY:
==25641==     in use at exit: 188,340 bytes in 179 blocks
==25641==   total heap usage: 26,048 allocs, 25,869 frees, 1,737,876 bytes allocated
==25641==
==25641== LEAK SUMMARY:
==25641==    definitely lost: 0 bytes in 0 blocks
==25641==    indirectly lost: 0 bytes in 0 blocks
==25641==      possibly lost: 2,880 bytes in 64 blocks
==25641==    still reachable: 185,460 bytes in 115 blocks
==25641==         suppressed: 0 bytes in 0 blocks
==25641== Rerun with --leak-check=full to see details of leaked memory
==25641==
==25641== For counts of detected and suppressed errors, rerun with: -v
==25641== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

恐ろしく日本語の
ることが不可能
終わります。

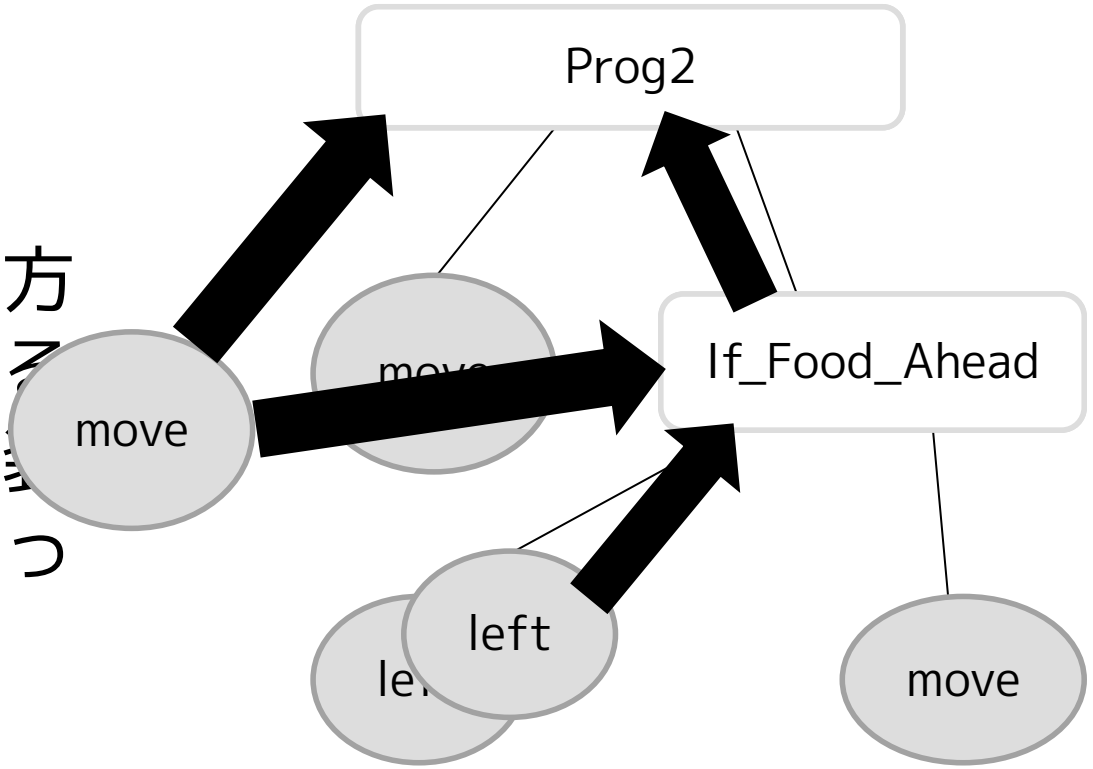
てろくに調べ

ようにしましょ

こんな怖いものに
遭遇します

Graph-Based GP

- 実装してないのに語っていいのか…
- 有向グラフの張り方が独特だったりするのでいろいろな実装が一気に難しくなってると僕は思う
- 右の木が…



まとめ

GPの概念は理解が容易だが実装が**僕のようなにわか勢**には難しい

その分マスターすれば**幸せ**になれる

計算量を落とそうとすると実装が**難しくなっていく**

Santa Fe Trailの蟻さんが動いてる様子を上手く表示すると**悪い個体でも可愛くしか見えない**（ただしUbuntuに限る）

ご静聴ありがとうございました。

@touyoubuntu