Evolving Event-driven Programs with SignalGP
Supplemental Material

Contents

1 Implementation

SignalGP is under active development as part of the Empirical library (`https://github.com/devosoft/Empirical`). Implementations of the test problems as well as the code used to perform statistical analysis and data visualization can be found at:
`https://github.com/amlalejini/GECCO-2018-Evolving-Event-driven-Programs-with-SignalGP`.

2 Instruction Set Details

Here, we provide details on the instruction sets used across our experiments. Instructions in the default instruction set (Table 1) were used in both the changing environment and distributed leader election test problems. Problem-specific instructions are given by Table 2 (changing environment problem) and Table 3 (distributed leader election problem). For each instruction, we give the following: the instruction name, the number of arguments that affect the instruction's behavior, whether or not the instruction's tag affects instruction behavior, and a brief description of instruction behavior. Note that some instructions behave differently depending on experimental conditions (see full paper for details).

We use the following abbreviations in our descriptions:

- ARG1, ARG2, ARG3: First, second, and third argument for an instruction.
- WM: Indicates working memory. WM[i] indicates accessing working memory at location i.
- IM: Indicates input memory. IM[i] indicates accessing input memory at location i.
- OM: Indicates output memory. OM[i] indicates accessing output memory at location i.
- SM: Indicates shared memory. SM[i] indicates accessing shared memory at location i.

- EOF: Indicates end of function.

Table 1: Default instructions

Instruction	Arguments	Uses Tag	Description
Inc	1	No	WM[ARG1] = WM[ARG1] + 1
Dec	1	No	WM[ARG1] = WM[ARG1] - 1
Not	1	No	WM[ARG1] = !WM[ARG1] (logically toggle WM[ARG1])
Add	3	No	WM[ARG3] = WM[ARG1] + WM[ARG2]
Sub	3	No	WM[ARG3] = WM[ARG1] - WM[ARG2]
Mult	3	No	WM[ARG3] = WM[ARG1] * WM[ARG2]
Div	3	No	WM[ARG3] = WM[ARG1] / WM[ARG2]
Mod	3	No	WM[ARG3] = WM[ARG1] % WM[ARG2]
TestEqu	3	No	WM[ARG3] = (WM[ARG1] == WM[ARG2])
TestNEqu	3	No	WM[ARG3] = (WM[ARG1] != WM[ARG2])
TestLess	3	No	WM[ARG3] = (WM[ARG1] < WM[ARG2])
If	1	No	If WM[ARG1] != 0, proceed; else, skip until next Close or EOF
While	1	No	If WM[ARG1] != 0, loop; else, skip until next Close or EOF
Countdown	1	No	Same as While, but decrements WM[ARG1]
Close	0	No	Indicates end of looping or conditional instruction block
Break	0	No	Break out of current loop
Call	0	Yes	Call function referenced by tag
Return	0	No	Return from current function
Fork	0	Yes	Calls function referenced by tag on a new thread
SetMem	2	No	WM[ARG1] = ARG2
CopyMem	2	No	WM[ARG1] = WM[ARG2]
SwapMem	2	No	Swap values in WM[ARG1] and WM[ARG2]
Input	2	No	WM[ARG2] = IM[ARG1]
Output	2	No	OM[ARG2] = WM[ARG1]
Commit	2	No	SM[ARG2] = WM[ARG1]
Pull	2	No	WM[ARG2] = SM[ARG1]
Nop	0	No	No-operation

Table 2: Changing Environment Problem Instructions

Instruction	Arguments	Uses Tag	Description
Terminate	0	No	Kills the thread this instruction was executed on
SenseEnvState0	1	No	WM[ARG1] = 1 if environment in state 0; else WM[ARG1] = 0
SenseEnvState1	1	No	WM[ARG1] = 1 if environment in state 1; else, WM[ARG1] = 0
SenseEnvState2	1	No	WM[ARG1] = 1 if environment in state 2; else, WM[ARG1] = 0
SenseEnvState3	1	No	WM[ARG1] = 1 if environment in state 3; else, WM[ARG1] = 0
SenseEnvState4	1	No	WM[ARG1] = 1 if environment in state 4; else, WM[ARG1] = 0
SenseEnvState5	1	No	WM[ARG1] = 1 if environment in state 5; else, WM[ARG1] = 0
SenseEnvState6	1	No	WM[ARG1] = 1 if environment in state 6; else, WM[ARG1] = 0
SenseEnvState7	1	No	WM[ARG1] = 1 if environment in state 7; else, WM[ARG1] = 0
SenseEnvState8	1	No	WM[ARG1] = 1 if environment in state 8; else, WM[ARG1] = 0
SenseEnvState9	1	No	WM[ARG1] = 1 if environment in state 9; else, WM[ARG1] = 0
SenseEnvState10	1	No	WM[ARG1] = 1 if environment in state 10; else, WM[ARG1] = 0
SenseEnvState11	1	No	WM[ARG1] = 1 if environment in state 11; else, WM[ARG1] = 0
SenseEnvState12	1	No	WM[ARG1] = 1 if environment in state 12; else, WM[ARG1] = 0
SenseEnvState13	1	No	WM[ARG1] = 1 if environment in state 13; else, WM[ARG1] = 0
SenseEnvState14	1	No	WM[ARG1] = 1 if environment in state 14; else, WM[ARG1] = 0
SenseEnvState15	1	No	WM[ARG1] = 1 if environment in state 15; else, WM[ARG1] = 0
SetState0	0	No	Set internal state to 0
SetState1	0	No	Set internal state to 1
SetState2	0	No	Set internal state to 2
SetState3	0	No	Set internal state to 3
SetState4	0	No	Set internal state to 4
SetState5	0	No	Set internal state to 5
SetState6	0	No	Set internal state to 6
SetState7	0	No	Set internal state to 7
SetState8	0	No	Set internal state to 8
SetState9	0	No	Set internal state to 9
SetState10	0	No	Set internal state to 10
SetState11	0	No	Set internal state to 11
SetState12	0	No	Set internal state to 12
SetState13	0	No	Set internal state to 13
SetState14	0	No	Set internal state to 14
SetState15	0	No	Set internal state to 15

Table 3: Distributed Leader Election Problem Instructions

Instruction	Arguments	Uses Tag	Description
RotCW	0	No	Rotate clockwise (90 degrees)
RotCCW	0	No	Rotate counter-clockwise (90 degrees)
RotDir	1	No	Rotate to face direction specified by (WM[ARG1] % 4)
RandomDir	1	No	WM[ARG1] is set to a random direction (values 0 through 3)
GetDir	1	No	WM[ARG1] is set to agent's current orientation
SendMsgFacing	0	Yes	Send output memory as message to faced neighbor
BroadcastMsg	0	Yes	Broadcast output memory as message to all neighbors
RetrieveMsg	0	Yes	Retrieve message from message inbox
GetUID	1	No	WM[ARG1] = Agent UID
GetOpinion	1	No	WM[ARG1] = Current opinion
SetOpinion	1	No	Set opinion to value specified by WM[ARG1]

3 Environment-state/Tag Associations for the Changing Environment Problem

In the event-driven and combined treatments for the changing environment problem, environmental changes produced signals that had environment-specific tags that could trigger functions. Table 4 shows the tags associated with each environment-state across all replicates.

Table 4: Environment-state/Tag Associations.

Environment-state	Associated Tag
0	0000000000000000
1	1111111111111111
2	1111000000001111
3	0000111111110000
4	1111000011110000
5	0000111100001111
6	0000000011111111
7	1111111100000000
8	0110011001100110
9	1001100110011001
10	1001011001101001
11	0110100110010110
12	0110011010011001
13	1001100101100110
14	1001011010010110
15	0110100101101001

4 Hand-coded Solutions

Here, we give a variety of hand-coded program solutions to the changing environment and distributed leader election problems. In each program: Functions are denoted by `Fn-<TAG>` where `<TAG>` gives the function's associated tag. Instructions may be followed by tags given in square brackets, and/or followed by arguments given in parentheses.

4.1 Changing Environment Problem

4.1.1 Event-driven solution for eight-state environment

```
Fn-0000000000000000:
        SetState0
Fn-1111111111111111:
        SetState1
Fn-1111000000001111:
        SetState2
Fn-0000111111110000:
        SetState3
Fn-1111000011110000:
        SetState4
Fn-0000111100001111:
        SetState5
Fn-0000000011111111:
        SetState6
```

```
Fn-1111111100000000:
        SetState7
```

4.1.2 Imperative solution for eight-state environment

```
Fn-0000000000000000:
  SetState0
  Fork[0000000000000001]
  Fork[0000000000000011]
  Fork[0000000000000111]
  Fork[0000000000001111]
  Fork[0000000000011111]
  Fork[0000000000111111]
  Fork[0000000001111111]
  Fork[0000000011111111]
  SetMem(0,1)
  While(0)
    SenseEnvState0(1)
    If(1)
      SetState0

Fn-0000000000000001:
  SetMem(0,1)
  While(0)
    SenseEnvState1(1)
    If(1)
      SetState1

Fn-0000000000000011:
  SetMem(0,1)
  While(0)
    SenseEnvState2(1)
    If(1)
      SetState2

Fn-0000000000000111:
  SetMem(0,1)
  While(0)
    SenseEnvState3(1)
    If(1)
      SetState3

Fn-0000000000001111:
  SetMem(0,1)
  While(0)
    SenseEnvState4(1)
    If(1)
      SetState4

Fn-0000000000011111:
  SetMem(0,1)
  While(0)
    SenseEnvState5(1)
    If(1)
      SetState5
```

```
Fn-0000000000111111:
  SetMem(0,1)
  While(0)
    SenseEnvState6(1)
    If(1)
      SetState6

Fn-0000000001111111:
  SetMem(0,1)
  While(0)
    SenseEnvState7(1)
    If(1)
      SetState7
```

4.2 Distributed Leader Election Problem

4.2.1 Event-driven solution

```
Fn-0000000000000000:
  GetUID(0)
  SetOpinion(0)
  SetMem(15,1)
  While(15)
    GetOpinion(0)
    Output(0,0)
    BroadcastMsg(0,0,0)[1111111111111111]
  Close

Fn-1111111111111111:
  Input(0,1)
  GetOpinion(0)
  TestLess(1,0,2)
  If(2)
    SetOpinion(1)
```

4.2.2 Imperative (fork-on-retrieve) solution

```
Fn-0000000000000000:
  GetUID(0)
  SetOpinion(0)
  SetMem(15,1)
  While(15)
    GetOpinion(0)
    Output(0,0)
    BroadcastMsg(0,0,0)[1111111111111111]
    RetrieveMsg
  Close

Fn-1111111111111111:
  Input(0,1)
  GetOpinion(0)
  TestLess(1,0,2)
  If(2)
    SetOpinion(1)
```

4.2.3 Imperative (copy-on-retrieve) solution

```
Fn-0000000000000000:
  GetUID(0)
  GetUID(1)
  SetOpinion(0)
  SetMem(15,1)
  While(15)
    GetOpinion(0)
    Output(0,0)
    BroadcastMsg(0,0,0)[1111111111111111]
    RetrieveMsg
    Input(0,1)
    TestLess(1,0,2)
    If(2)
      SetOpinion(1)
    Close
  Close
```