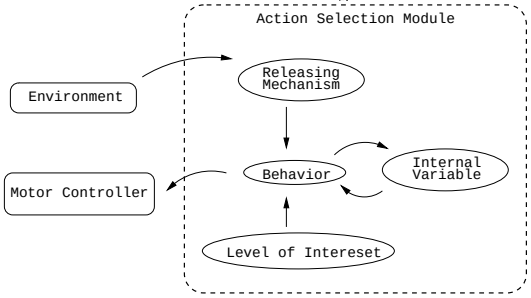
Agents that Have Desires and Adaptive Behaviors (Using Stochastic Cellular Automata)

December 4, 2002

1 摘要

這是一個希望能夠產生出可自動交易的 agent, scaView.cpp 中 的 AtTraining() 這個函式是整個 agent 運作的過程. 透過 implement Blumberg 的 Model[?]來實做出可自動交易之 agent. 其中內部是用 stochastic cellular automata(SCA) 來 implement 其 update 腦內活動的 方式. 這個可自動交易的 agent 會隨時依自己和對方的情形調整對於時間和 金錢的慾望(時間拖越久, 對方越不耐煩, 越能照你所要求的給付, 然而時間 一拖久, 自己也會不耐煩.) 透過這個可自動交易之 agent, 希望能時做出類 似人類所擁有的慾望和情緒.

Blumberg 的 model 主要如下圖, 其中有四個 component. Internal Varaible(IV) 是用來建構內在狀態, 可以根據不同時 間下的 growth rate, damping rate 和根據正在表現的行為而產生的影響來 改變. Releasing Mechanism(RM) 是用來感知外在的刺激. Level of Interest 是用來建構當時行為表現時所發生的影響.



Deterministic Cellular automata(DCA) 一開始是由 Von Neumann [?, ?] 提出的. 其中包含一個 discrete cellular space U, 4 個集合 X, Y, Q, N, 和 4 個function Υ, δ, β, Ω, 其中:

1. X 是 input 的集合
2. Y 是 output 的集合
3. Q 是 state 的集合
4. N 是 與鄰居的關係 的集合
5. Υ 決定要與那些鄰居相連

$$Q \to \otimes_N Q$$

6. δ 決定當 X 進來時要進到的下一個 Q

$$X \times \otimes_N Q \to Q$$

7. β 決定要 output 什麼東西給 Y

$$\otimes_N Q \to Y$$

8. Ω 決定整個 cellular automata 改變和 output 給 Y 的過程

$$\otimes_N Q \to \otimes_N (Q \times Y)$$

Stochastic cellular automata (SCA) 是由 Y.C. Lee 等 [?] 提出的, 維持 U, X, Q, N, Υ 的定義, 而將 δ 和 β 加上機率, 而叫做 F 和 G, 而因為是 SCA, 所以並沒有固定的改變過程, 所以 Ω 就被包含在 F 和 G 裡.

1. F: $X \times \otimes_N Q \to$, 對於 x, n, q, 其存在 一個機率 $f(x, n, q)$

$$\forall x \in X, \forall n \in N, \sum_{q \in Q} f(x, n, q) = 1$$

2. G: $\otimes_N Q \to Y$, 對於 n, y, 其存在一個機率 $g(n, y)$

$$\forall n \in N, \sum_{y \in Y} g(n, y) = 1$$

2 algorithm 解說

2.1 大致流程

對於我們的 agent 的參數是存在 ln1.txt 裡, 對於其他的 agent 的參數是存 在 vd1.txt裡, 在 initialize 的時候, 從 sample.txt copy 相關的參數到 ln1.txt 裡, 而關於 IV 的6個參數(對於 time 和 money 期望的值, damp 的值, 和 growth 的值) 則適用 random 產生的. 然後對於每次 iteration, 以同樣方 法將 sample.txt copy 到 vd1.txt, 並 random 產生其 IV 的6個參數. 然後用 Blumberg 的 model 來對 ln1.txt, vd1.txt 的參數做 update, 直到交易成功 (對於金錢的期望是一樣的), 或是交易結束了. 然後再用 SCA 的方法來對自己 的 agent 的 IV 做 update, 就完成了一次 iteration.

2.2 Blumberg's Life-like Equation

TimePay的update公式是:

$$TimePay = TimePay + Increase_{Time}$$

IV會自行update, update公式是:

$$IV_i(t+1) = IV_i(t) \times damp_{IV_i} + growthIV_i - \sum_{j=0}^{\#B-1} gain_{j,IV_i} \times B_j$$

LI的Update公式是:

$$LI_i(t+1) = LI_i(t) \times damp_{LI_i} + growth_{LI_i} - B_i \times bRate_{LI_i}$$

RM是根據各個RM的特性而決定的, 其Update公式是:

$$\begin{aligned} Stimulus_t(t+1) =& Find(s_i(t+1), dMin_i, dMax_i) \cdot \\ & Filter(s_i(t+1)) \cdot Weight(s_i(t+1), Opti) \\ RM_i(t+1) =& Clamp(TemporalFilter(t, rm_i(t)), Stimulus_i(t+1), min_i, max_i) \end{aligned}$$

B是根據RM, RIV, IV, LI等來決定的, 其公式為:

$$B_i(t+1) = Max[(LI_i(t+1) \cdot Combine(\sum_k RM_{ki}, \sum_j IV_j(t+1)) - \sum_m ig_{mi} \cdot v_m(t+1)), 0]$$

最後根據B決定的外顯行為來決定新的money要調成多少.

2.3 Stochastic Cellular Automata

x_i為 internal variable(IV)中所對應到的6個variable.

$x_i = n \Rightarrow$在x_i這個陣列中最多random出n 個設為1. 例:

$x_0 = 5$:

$x_0(0) = 1$
$x_0(3) = 1$
$x_0(12) = 1$
$x_0(40) = 1$
$x_0(50) = 1$

$x_1 = 3$:

$x_1(3) = 1$
$x_1(41) = 1$
$x_1(52) = 1$

$x_2 = 1$:

$x_2(13) = 1$

$x_3 = 4$:

$x_3(2) = 1$
$x_3(40) = 1$
$x_3(41) = 1$

$x_3(52) = 1$

$x_4 = 3$:

$x_4(2) = 1$

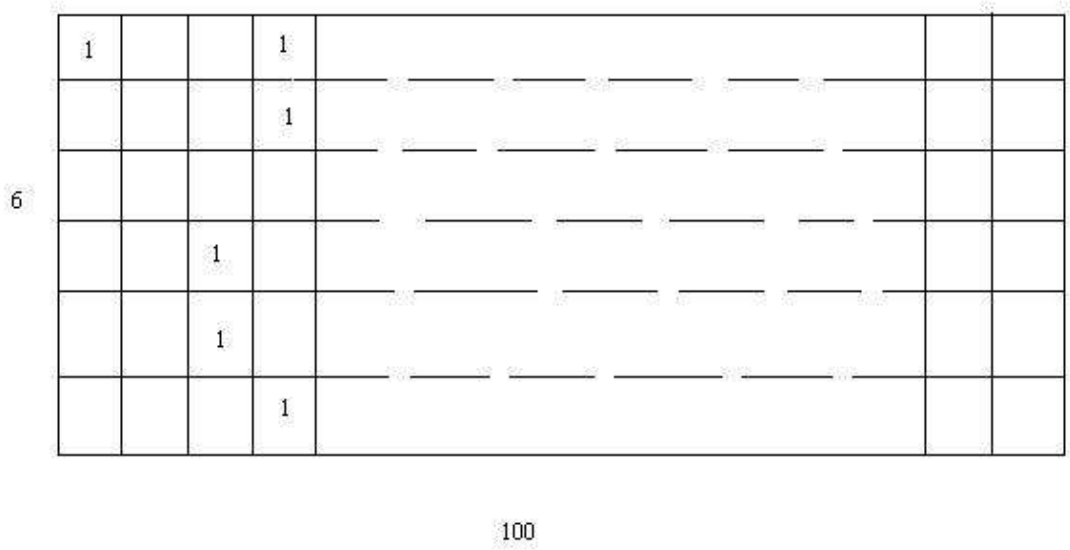
$x_4(13) = 1$

$x_4(50) = 1$

$x_5 = 2$:

$x_5(3) = 1$

$x_5(50) = 1$

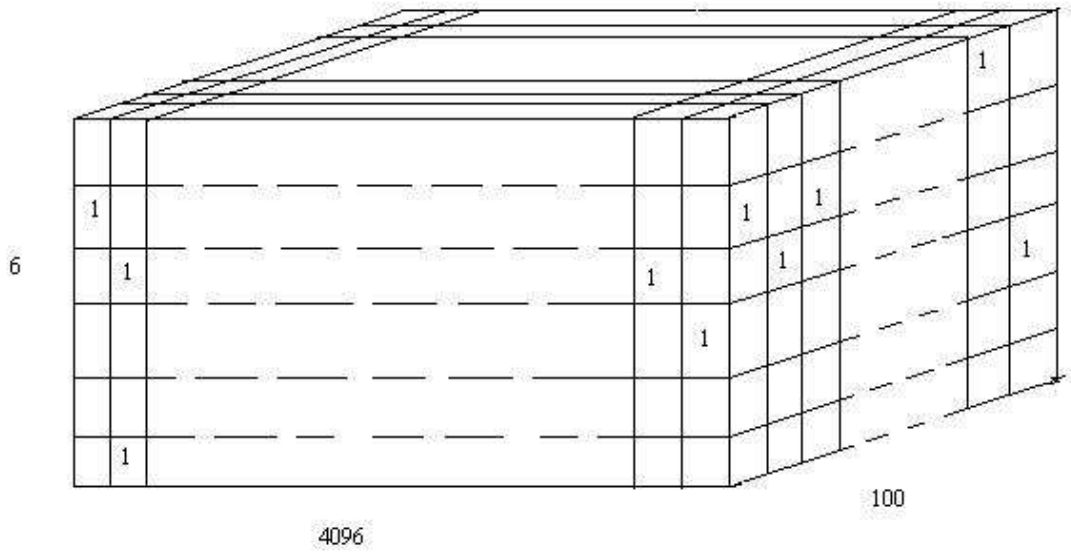


S(j)的值是$x_0(j)$到$x_5(j)$所構成的function所決定的. 其 function為:

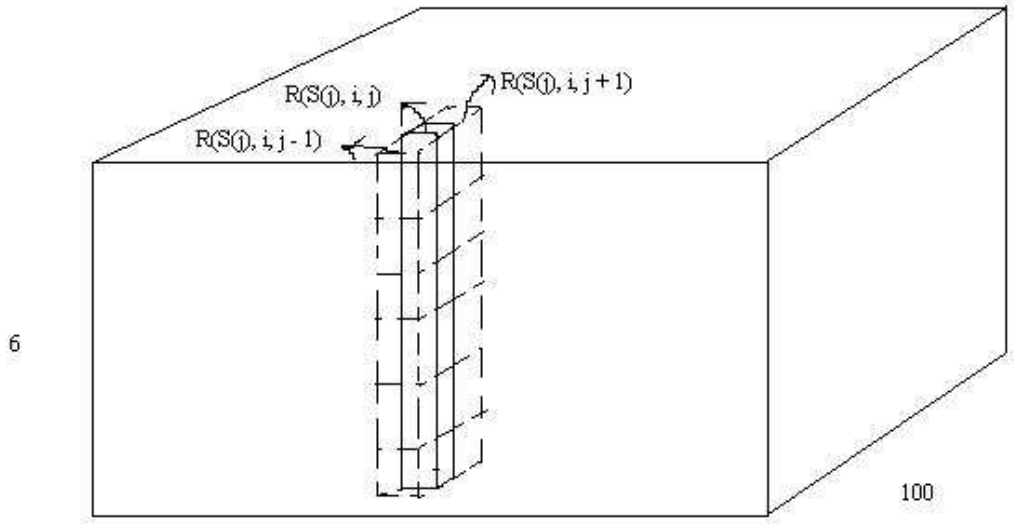
$$S(j) = \sum_{i=0}^{5} (x_i(j-1) + x_i(j) + x_i(j+1)) \times 4^i$$

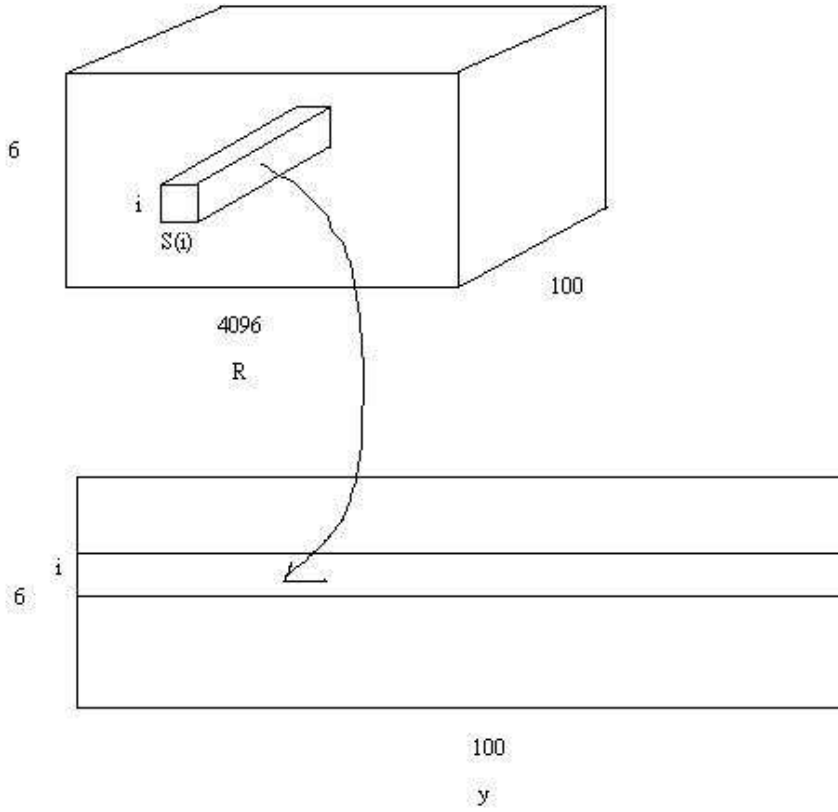
例:

$$\begin{aligned}
S(0) &= 1 \times 4^0 \\
&= 1 \\
S(2) &= 1 \times 4^0 + 1 \times 4^1 + 1 \times 4^3 + 1 \times 4^4 + 1 \times 4^5 \\
&= 1349 \\
S(3) &= 1 \times 4^0 + 1 \times 4^1 + 1 \times 4^3 + 1 \times 4^4 + 1 \times 4^5 \\
&= 1349 \\
S(12) &= 1 \times 4^0 + 1 \times 4^2 + 1 \times 4^4 \\
&= 273 \\
S(40) &= 1 \times 4^0 + 1 \times 4^1 + 2 \times 4^3 \\
&= 133 \\
S(51) &= 1 \times 4^0 + 1 \times 4^1 + 1 \times 4^3 + 1 \times 4^4 + 1 \times 4^5 \\
&= 1349
\end{aligned}$$



對於每個variable i和每個S_j, 如果是結果是好的, 就把附近 的R(S(j), i, j - 1) 和 R(S(j), i, j + 1)有機會設成跟R(S(j), i, j)一樣, 如果是不好的, 對於R(S(j), i, j - 1), R(S(j), i, j), R(S(j), i, j + 1), 有機會將其設為相反, 在本程式中, 機率是設成1 (scaView.cpp: 130), 亦即如果結果是好的, 就把R(S(j), i, j - 1) 和R(S(j), i, j + 1) 設成跟R(S(j), i, j), 若結 果是不好的, 對於R(S(j), i, j - 1), R(S(j), i, j), R(S(j), i, j + 1), 都設成相反.


R(S(j), i, j)
R(S(j), i, j + 1)
R(S(j), i, j - 1)
6
100
4096

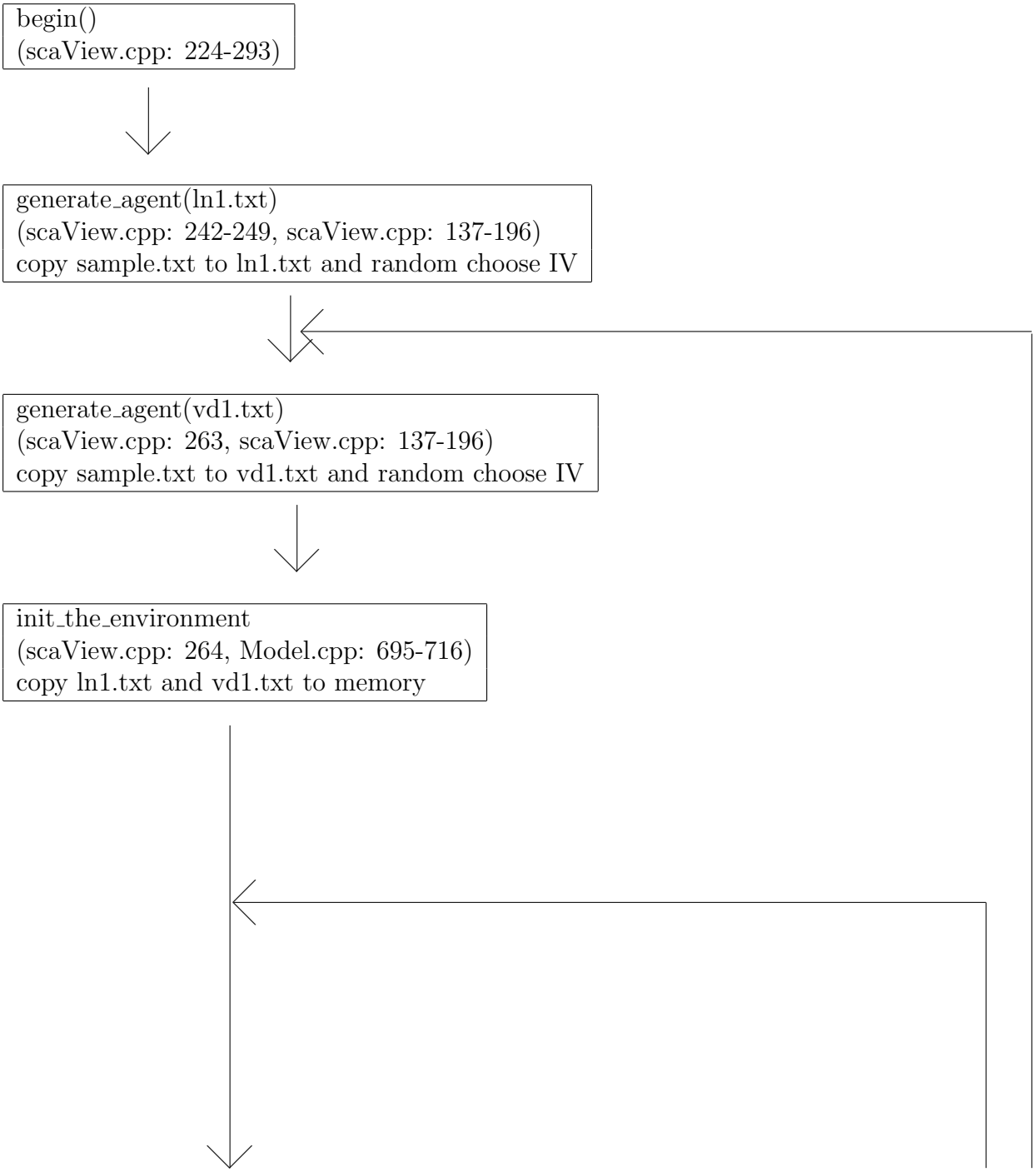


$y_i(j)$為其相對應之$R(S(i), i, j)$.

由$y_i(j)$map出來的y_i對應到internal variable(IV)的6個 variable, 為 IV(t + 1) 與 IV(t) 之間的差, 在 ± 0.05之間, 其 function 為:

$$diff_{IV_i} = \frac{\sum_{j=0}^{99} y_i(j)}{100} * 0.1 - 0.05$$

3 流程圖


begin()
(scaView.cpp: 224-293)
generate_agent(ln1.txt)
(scaView.cpp: 242-249, scaView.cpp: 137-196)
copy sample.txt to ln1.txt and random choose IV
generate_agent(vd1.txt)
(scaView.cpp: 263, scaView.cpp: 137-196)
copy sample.txt to vd1.txt and random choose IV
init_the_environment
(scaView.cpp: 264, Model.cpp: 695-716)
copy ln1.txt and vd1.txt to memory

update_both_outside_parameter()
(scaView.cpp: 268, Model.cpp: 717-731, Model.cpp: 679-687)
用 Blumberg的 model來做 update

updateRIV()
(Model.cpp: 681, Model.cpp: 380-385)
$TimePay = TimePay + Increase_{Time}$

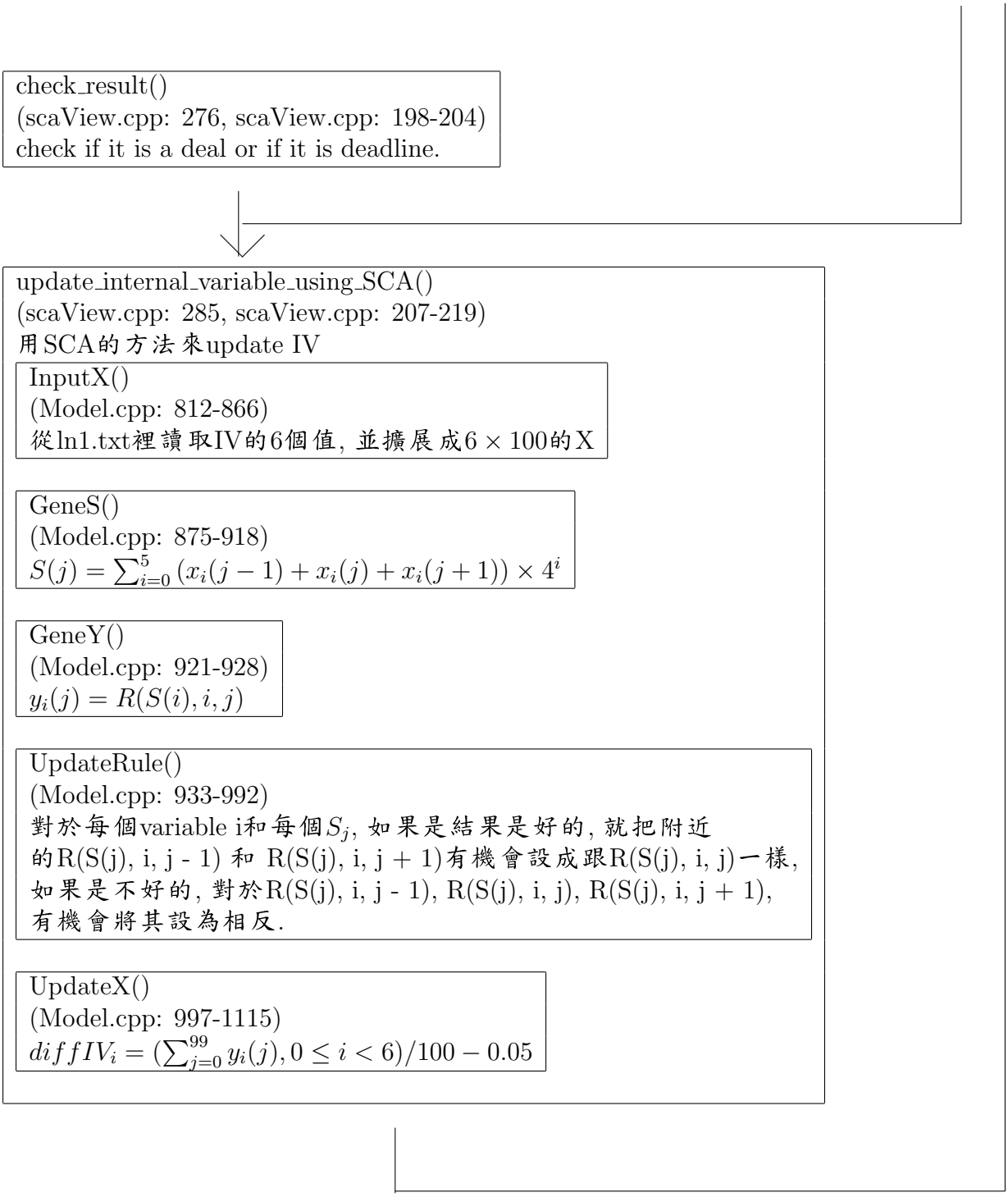
updateIV()
(Model.cpp: 682, Model.cpp: 386-399)
$IV_i(t+1) = IV_i(t) \times damp_{IV_i(t)} + growthIV_i(t) - \sum_{j=0}^{\#B-1} gain_{j,IV_i(t)} \times B_j$

updateLI()
(Model.cpp: 683, Model.cpp: 400-411)
$LI_i(t+1) = LI_i(t) \times damp_{LI_i(t)} + growth_{LI_i(t)} - B_i \times bRate_{LI_i(t)}$

updateRM()
(Model.cpp: 684, Model.cpp: 412-476)
$Stimulus_t(t+1) = Find(s_i(t+1), dMin_i, dMax_i) \cdot Filter(s_i(t+1))$
$\cdot Weight(s_i(t+1), Opti)$
$RM_i(t+1) = Clamp(TemporalFilter(t, rm_i(t)), Stimulus_i(t+1), min_i, max_i)$

updateB()
(Model.cpp: 685, Model.cpp: 477-622)
$B_i(t+1) = Max[(LI_i(t+1) \cdot Combine(\sum_k RM_{ki}, \sum_j IV_j(t+1)) -$
$\sum_m ig_{mi} \cdot v_m(t+1)), 0]$

Modify()
(Model.cpp: 686, Model.cpp: 623-678)


check_result()
(scaView.cpp: 276, scaView.cpp: 198-204)
check if it is a deal or if it is deadline.
update_internal_variable_using_SCA()
(scaView.cpp: 285, scaView.cpp: 207-219)
用SCA的方法來update IV
InputX()
(Model.cpp: 812-866)
從ln1.txt裡讀取IV的6個值, 並擴展成6 × 100的X
GeneS()
(Model.cpp: 875-918)
$S(j) = \sum_{i=0}^{5} (x_i(j-1) + x_i(j) + x_i(j+1)) \times 4^i$
GeneY()
(Model.cpp: 921-928)
$y_i(j) = R(S(i), i, j)$
UpdateRule()
(Model.cpp: 933-992)
對於每個variable i和每個S_j, 如果是結果是好的, 就把附近
的R(S(j), i, j - 1) 和 R(S(j), i, j + 1)有機會設成跟R(S(j), i, j)一樣,
如果是不好的, 對於R(S(j), i, j - 1), R(S(j), i, j), R(S(j), i, j + 1),
有機會將其設為相反.
UpdateX()
(Model.cpp: 997-1115)
$diffIV_i = (\sum_{j=0}^{99} y_i(j), 0 \leq i < 6)/100 - 0.05$

4 程式/公式/init值 對照表

公式	程式	init值(請見sample.txt)
$Money$	m_RIV[RIV_MoneyPay].val	150
$TimePay$	m_RIV[RIV_TimePay].val	0
$Increase_{Time}$	m_RIV[RIV_TimePay].inc	0
IV_i	m_IV[i].val	random
$damp_{IV_i}$	m_IV[i].damp	random
$growth_{IV_i}$	m_IV[i].growth	random
LI_i	m_LI[i].val	(LI部份第一個值)
$damp_{LI_i}$	m_LI[i].damp	(LI部份第二個值)
$growth_{LI_I}$	m_LI[i].growth	(LI部份第三個值)
RM_i	m_RM[i].val	0
$Filter$	m_RM.findMin, m_RM.findMax	(RM部份最後兩個值)
B_i	m_B[i].val	(B部份第一個值)
外顯行為	m_BOut	none
Find	RMFind()	none
Clamp, TemporalFilter	RMTemporalFilterSum()	none
Combine	UpdateB()	none
決定money調成多少	Modify()	none
/hline $x_i(j)$	X[i][j]	none
$S(j)$	S[i]	none
$R(i,j,k)$	R[i][j][k]	none
$y_i(j)$	Y[i][j]	none