第一篇文章

Coevolution in Management Fashion:

An Agent-Based Model of Consultant-Driven Innovation

關於第一篇文章 Coevolution in Management Fashion: An Agent-Based Model of Consultant-Driven Innovation，描述了一個假想市場機制，由廠商與顧問公司構成，依據理想水準、交易後報酬的差異，以及隨機過程的設定來完全支持市場的運作。依據論文的建議模擬上分為七個步驟：

其中，前三個步驟為初始化與基礎設定，並使得廠商與顧問公司進行第一次媒合：

1. consultants have a current innovation they supply
2. firms have a current innovation they demand
3. firms and consultants are matched

而後續的四個步驟則以迴圈重複計算：

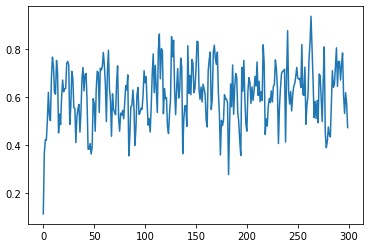
4與5做的是對於交易後商品所做的評價與報酬的評估與計算

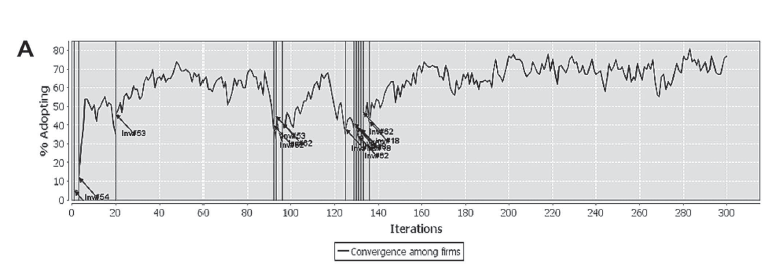
1. each consultant receives a return based on demand for its services
2. each firm receives a outcome based on its performance

6與7做的是對於交易後商品計畫的沿用以及變更，並在變更後使其在交易

1. consultants decide (in light of their returns) whether to continue to offer their current innovation versus abandon it foranal ternative
2. firms decide (in light of their out comes) whether to continue to utilize their current innovation versus abandon it for an alternative

由這七個步驟，我們得以建構出論文所描述的市場機制，並依據論文中圖表的繪製，複製出圖二的結果如下：





雖然此分布為一隨機過程，但經與原始論文比對，可以發現其所展現的分布與發展趨勢上的相似性。

最模擬上，發現最大的障礙在於對於論文本身理解度的不足，許多名詞都未能充分理解，其中，對於交易後判斷的機制中存在一 probability of abandonment 的設定則使人有較高度的疑惑，開始時對於其使用時機不慎清楚。如果時間在更充分應當能將其完成的更臻完備。

第二篇文章

Why Do Liberals Drink Lattes?

關於第二篇文章 Why Do Liberals Drink Lattes?，描述了一個假想社會的人際互動與思想交流模式，由許多獨立個體構成，依據觀念上的「距離」來作為連結與否的考量，並存在思想相互影響的機制。依據論文模擬步驟如下：

首先是初始化：

1. random N 個points

2. 賦予各個point具有各自的static attributes和dynamic dimensions

3. 將pointer間互相計算dynamic dimensions

4. 得出各點與各點間的d 作為T=0的E(d)

再來則開始做計算：

5. 將各點間互相配對並給予機率改變其dynamic dimensions

6. 再次計算各點得出T=1時的d

7. 將E(d)-d(1)可以得出w權重

8. 使用w得出p的機率

9. 將p加入，作為各點再次改變dynamic dimensions的機率

並依以上步驟重複運算：

10. For loop

但由於此模擬在儲存上時發生嚴重狀況，導致最後模擬程式有缺漏，最終導致無法跑動，由於礙於時間因素，最後距離模擬成功仍差臨門一腳。

Coevolution in Management Fashion: An Agent-Based Model of Consultant-Driven Innovation

import random, math

def creturn(m, f, s):

  eta = 0.5

  retu = eta\*m\*f\*s

  return retu

def fout(v, q):

  alfa = 0.7

  beta = 0.2

  e = 0.2

  o = alfa\*v + beta\*q + (1 - alfa - beta)\*e

  return o

def aspir(ga, ah, ass):

  ah = ga\*ah + (1- ga)\*ass

  return ah

def fprob(oo, aa):

  a = 0.6

  b = 0.4

  gg = a + b\*( oo - aa )

  pp = 1/(1 + math.exp(gg))

  return pp

def cprob(avgk, sums):

  if (sums == 0):

    pp = 0

  else:

    pp = avgk / sums

  return pp

## fundamental conditions

aah = 0

out = [] # performance outcome of flim

ret = [] # consultant’s return

probc = [] # the probability that a firm will select consultant

probf = [] # the probability of abandonment

oit = []

pool = [] # innovation pool

con = [] # innovation of consultant

firm = [] # innovation of firm

on = [] # selected inno.

selected = [] # supply inno.

match = [] # choice of firm

trans = [] # transaction between consultant and firm

vvd = []

qqd = []

mmvd = []

zeta = 0.8

cconn = []

#make inno. pool

for m in range(0,10):   #####

  pool.append(m)

  on.append(0)

print("pool:"+str(pool))

for t in range(0,10):   #####

  cret = []

  fac = []

  fprobabi = []

  cac = []

  cprobabi = []

# step1 consultants have acurrent innovation they supply

  for c in range(0, 10):   #####

    if len(pool) >= 2:

      chosen = random.randrange(len(pool)-1)

      con.append(pool[chosen])

      on[chosen] = 1

    else:

      con.append(0)

      on[0] = 1

  print("con:"+str(con))

# mark chosen inno.

  var = 0

  for x in on:

    if x == 1:

      selected.append(pool[var])

    var +=1

# step2 firms have a current innovation they demand

  for f in range(0, 1):   #####

    if len(selected) >= 2:

      chosen = random.randrange(len(selected)-1)

      firm.append(selected[chosen])

    else:

      firm.append(selected[0])

  print("firm:"+str(firm))

# step3 firms and consultants are matched

  choice = []

  for r in firm:

    var = 0

    for c in con:

      if r == c:

        choice.append(var)

      var +=1

    match.append(choice)

    choice=[]

  print("match:"+str(match))

  choice=[]

  var = 0

  m = 0

  for r in match:

    choice.append(var)

    m = random.randrange(len(match[var]))

    choice.append(match[var][m])

    trans.append(choice)

    choice=[]

    var+=1

  print("trans:"+str(trans))

for t in range(0,300):

  cret = []

  fac = []

  fprobabi = []

  cac = []

  cprobabi = []

  oit = []

# step4 each consultant receives a return based on demand for its services

  for kk in range(0, len(con)):

    innov = con[kk]

    team = kk

    # the number of firms pursuing innovation j

    for i in range(0, len(firm)):

      if (firm[i] == innov):

        k = 1

      else:

        k = 0

      t = t + k

    s = t

    t = 0

    # the number of consultants offering innovation j

    for i in range(0, len(con)):

      if (con[i] == innov):

        k = 1

      else:

        k = 0

      t = t + k

    f = t

    t = 0

    # the number of clients served by consultant c

    t = 0

    for i in range(0, len(firm)):

      if (trans[i][0] == team):

        k = 1

      else:

        k = 0

      t = t + k

    m = t

    t = 0

    cret.append( creturn(m, f, s) )

# step 5 each firm receives a outcome based on its performance

  vdv = 0

  qdq = 0

  for k in range(0, len(firm)):

    v = firm[k]\*0.5 + 0.5

    q = trans[k][1]\*0.5 + 0.5

    oit.append(fout(v, q))

    vdv = vdv + v

    qdq = qdq + q

  vvd.append(vdv/len(firm))

  qqd.append(qdq/len(firm)/40)

# step 6 con decide (in light of their out comes) whether to continue to utilize their current innovation versus abandon it for an alternative

  for k in range(0, len(con)):

    team = k

    sws = []

    for i in range(0, len(firm)):

      if ( trans[i][1] == team):

        ddp = trans[i][1]

        sws.append(ddp)

    dds = 0

    for i in range(0, len(sws)):

      ssf = cret[ sws[i] ]

      dds = dds + ssf

    if ( len(sws) == 0 ):

      avgk = 0

    else:

      avgk = dds / len(sws)

    sdg = 0

    pdg = 0

    for i in range(0, len(con)):

      if (con[i] == innov):

        ffg = cret[i]

        sdg = sdg + ffg

        pdg += 1

    if (pdg == 0):

      sums = 0

    else:

      sums = sdg / pdg

    ccpr = cprob(avgk, sums)

    cprobabi.append( ccpr )

    ffn = 0

    for i in range(0, len(con)):

      sns = cret[i]

      ffn = ffn + sns

    if (ffn == 0):

      ass = 0

    else:

      aas = 1 / ( (len(con)-1)\*ffn )

    aah = (1- zeta)\*aah + zeta\*cret[team]

    ccac = aspir(0.3, aah, aas)

    cac.append( ccac )

    if (ccpr >= ccac):

      pass

    else:

      sds = random.randrange(10)

      if (sds < 5):

        con[k] = random.randrange(len(pool)-1)

      else:

        lll = cret.index( max(cret) )

        con[k] = con[lll]

# step 7 firm decide (in light of their returns) whether to continue to offer their current innovation versus abandon it foranal ternative

  for k in range(0, len(firm)):

    innov = firm[k]

    team = k

    ffn = 0

    for i in range(0, len(firm)):

      sns = oit[i]

      ffn = ffn + sns

    aas = 1 / ( (len(firm)-1)\*ffn )

    aah = (1- zeta)\*aah + zeta\*oit[team]

    ffac = aspir(0.3, aah, aas)

    fac.append( ffac )

    ffpr = fprob(oit[k], aah)

    fprobabi.append( ffpr )

    if (ffpr >= ffac):

      mmm = random.random()

      if (mmm < ffpr):

        sds = random.randrange(10)

        if (sds < 5):

          hh = random.randrange(len(con)-1)

          firm[k] = con[hh]

          vvv = []

          for m in range(0, len(con) ):

            if (con[m] == firm[k]):

              vvv.append(m)

            match[k] = vvv

          io = random.randrange(len(vvv))

          trans[k][1] = vvv[io]

        else:

          lll = oit.index( max(oit) )

          firm[k] = firm[lll]

      else:

        pass

    else:

      sds = random.randrange(10)

      if (sds < 5):

        hh = random.randrange(len(con)-1)

        firm[k] = con[hh]

        vvv = []

        for m in range(0, len(con) ):

          if (con[m] == firm[k]):

            vvv.append(m)

          match[k] = vvv

        io = random.randrange(len(vvv))

        trans[k][1] = vvv[io]

      else:

        lll = oit.index( max(oit) )

        firm[k] = firm[lll]

  for ii in range(0, len(firm)):

      mmvd.append(firm[ii])

Why Do Liberals Drink Lattes?

def dist(dim1, dim2):

  tt = 0

  for i in range(0, nn):

    ss = (dim1[i] - dim2[i])\*\*2

    tt = tt + ss

  d = tt\*\*(1/2)

  return d

def weig(ed, dd):

  w = ed - dd

  return w

def kisang(ss):

  supu = []

  for i in range(0, point):

    for k in range(i+1, point):

      a = i\*10 + k

      supu.append(a)

  s=[]

  for b in range(0, len(supu)):

    if ( int(supu[b]/10) == ss)|(supu[b] % 10 == ss):

      s.append(b)

  return s

from numpy import \*

import random

import math

##### 前置準備

time = 10

ed = []

ss1 = []

oo1 = []

oo2 = []

oo3 = []

supu = []

nn = 3

dis = []

point = 10

percent = 0.8

max\_goal = math.ceil(point\*percent)

dim\_ini = []

dim\_com = []

dd = []

weight = []

vv = 0

ccc = 0

for i in range(0, point):

  ss1.append(random.randrange(5))

  oo1.append(random.randrange(2))

  oo2.append(random.randrange(2))

  oo3.append(random.randrange(2))

# 初始 ED

for i in range(0, point):

  dim1 = []

  dim1.append(oo1[i])

  dim1.append(oo2[i])

  dim1.append(oo3[i])

  dim\_ini.append([])

  dim\_ini[i]+=dim1

  dim\_com.append(dim1)

  for j in range(i+1, point):

    dim2 = []

    dim2.append(oo1[j])

    dim2.append(oo2[j])

    dim2.append(oo3[j])

    ddd = dist(dim1, dim2)

    ed.append(ddd)

    dis.append(ddd)

# max\_distance

s=[]

c=[]

for i in range(0,point):

  s.append(kisang(i))

  c.append([])

  for j in range(0,point-1):

    c[i].append(ed[s[i][j]])

s=[]

for i in range(0, point):

  s.append(max(c[i]))

print("dim\_com:"+str(dim\_com))

print("ed:"+str(ed))

### 尋找連結標的

var = 0

confuse = []

for i in range(0, point):

  mmmg = s[i]

  goal = 0

  for k in range(i+1, point):

    if (i != k):

      if (k != point-2):

        if (ed[var] < mmmg):

          goal+=1

          for j in range(0,3): #loop dynamic dimensions

            p=random.random()

            if (p >= 0.5):

              dim\_com[i][j] = dim\_ini[k][j]

        elif (ed[var] == mmmg):

            confuse.append([i,k])

      else:

        if (i < point-1):

          if (ed[var] < mmmg):

            goal+=1

            for j in range(0,3): #loop dynamic dimensions

              p=random.random()

              if (p >= 0.5):

                dim\_com[i][j] = dim\_ini[k][j]

          elif (ed[var] == mmmg):

            confuse.append([i,k])

        elif (i == point-1):

          if (ed[var] < mmmg):

            goal+=1

            for j in range(0,3): #loop dynamic dimensions

              p=random.random()

              if (p >= 0.5):

                dim\_com[i][j] = dim\_ini[k][j]

          elif (ed[var] == mmmg):

            confuse.append([i,k])

          if (goal != max\_goal):

            print("g"+str(goal))

            for l in range(0,max\_goal-goal):

              print(confuse)

              r = random.randrange(len(confuse))

              for j in range(0,3):

                p=random.random()

                if (p >= 0.5):

                  dim\_com[confuse[r][0]][j] = dim\_ini[confuse[r][1]][j]

              del confuse[r]

      var+=1

print("dim\_com"+str(dim\_com))

# record initial odd

dim\_ini = []

for m in range(0,len(dim\_com)):

  dim\_ini.append([])

  dim\_ini[m] += dim\_com[m]

#dd = d(t)

dd=[]

dis=[]

for i in range(0, point):

  for j in range(i+1, point):

    ddd = dist(dim\_com[i], dim\_com[j])

    dd.append(ddd)

    dis.append(ddd)

print("dd:"+str(dd))

#max\_distance

s=[]

c=[]

for i in range(0,point):

  s.append(kisang(i))

  c.append([])

  for j in range(0,point-1):

    c[i].append(dd[s[i][j]])

s=[]

for i in range(0, point):

  s.append(max(c[i]))

#weight

for i in range(0, len(ed)):

  ppp = ed[i]

  poo = dd[i]

  weight.append(weig(ppp, poo))

print("weight:"+str(weight))

for i in range(0, len(weight)):

  vv = abs(weight[i])

  ccc = ccc + vv

sigw = ccc

print("sigw:"+str(sigw))

v = 0

confuse = []

for i in range(0, point):

  mmmg = s[i]

  goal = 0

  for k in range(i+1, point):

    if (i != k):

      if (k != point-2):

        if (dd[v] < mmmg):

          goal+=1

          for j in range(0,3): #loop dynamic dimensions

            inte = abs(int(sigw\*10000))+1

            if (sigw != 0):

              www=random.randrange(inte)

            else:

              www = 1

            if (www < abs(weight[v]\*10000)+1):

              fff = dim\_ini[i][j]

              dim\_com[k][j] = fff

        elif (dd[v] == mmmg):

            confuse.append([i,k])

      else:

        if (i < point-1):

          if (dd[v] < mmmg):

            goal+=1

            for j in range(0,3): #loop dynamic dimensions

              inte = abs(int(sigw\*10000))+1

              if (sigw != 0):

                www=random.randrange(inte)

              else:

                www = 1

              if (www < abs(weight[v]\*10000)+1):

                dim\_com[k][j] = dim\_ini[i][j]

          elif (dd[v] == mmmg):

              confuse.append([i,k])

        elif (i == point-1):

          if (dd[v] < mmmg):

            goal+=1

            for j in range(0,3): #loop dynamic dimensions

              inte = abs(int(sigw\*10000))+1

              if (sigw != 0):

                www=random.randrange(inte)

              else:

                www = 1

              if (www < abs(weight[v]\*10000)+1):

                dim\_com[k][j] = dim\_ini[i][j]

          elif (dd[v] == mmmg):

                confuse.append([i,k])

          if (goal != max\_goal):

            for l in range(0,max\_goal-goal):

              r = random.randrange(len(confuse))

              for j in range(0,3):

                inte = abs(int(sigw\*10000))+1

                if (sigw != 0):

                  www=random.randrange(inte)

                else:

                  www = 1

                if (www < abs(weight[v]\*10000)+1):

                  print("y")

                  print(dim\_com[confuse[r][0]][j])

                  print(dim\_ini[confuse[r][1]][j])

                  dim\_com[confuse[r][1]][j] = dim\_ini[confuse[r][0]][j]

              del confuse[r]

      v+=1

# record initial odd

dim\_ini = []

for m in range(0,len(dim\_com)):

  dim\_ini.append([])

  dim\_ini[m] += dim\_com[m]

#dd = d(t)

dd=[]

dis=[]

for i in range(0, point):

  for j in range(i+1, point):

    ddd = dist(dim\_com[i], dim\_com[j])

    dd.append(ddd)

    dis.append(ddd)

print("dd:"+str(dd))

#max\_distance

s=[]

c=[]

for i in range(0,point):

  s.append(kisang(i))

  c.append([])

  for j in range(0,point-1):

    c[i].append(dd[s[i][j]])

s=[]

for i in range(0, point):

  s.append(max(c[i]))

#weight

for i in range(0, len(ed)):

  ppp = ed[i]

  poo = dd[i]

  weight.append(weig(ppp, poo))

print("weight:"+str(weight))

for i in range(0, len(weight)):

  vv = abs(weight[i])

  ccc = ccc + vv

sigw = ccc

print("sigw:"+str(sigw))

print("dim\_com:"+str(dim\_com))

for t in range(0, time): # time

  v = 0

  confuse = []

  for i in range(0, point):

    mmmg = s[i]

    goal = 0

    for k in range(i+1, point):

      if (i != k):

        if (k != point-2):

          if (dd[v] < mmmg):

            goal+=1

            for j in range(0,3): #loop dynamic dimensions

              inte = abs(int(sigw\*10000))+1

              if (sigw != 0):

                www=random.randrange(inte)

              else:

                www = 1

              if (www < abs(weight[v]\*10000)+1):

                dim\_com[i][j] = dim\_ini[k][j]

          elif (dd[v] == mmmg):

              confuse.append([i,k])

        else:

          if (i < point-1):

            if (dd[v] < mmmg):

              goal+=1

              for j in range(0,3): #loop dynamic dimensions

                inte = abs(int(sigw\*10000))+1

                if (sigw != 0):

                  www=random.randrange(inte)

                else:

                  www = 1

                if (www < abs(weight[v]\*10000)+1):

                  dim\_com[i][j] = dim\_ini[k][j]

            elif (dd[v] == mmmg):

                confuse.append([i,k])

          elif (i == point-1):

            if (dd[v] < mmmg):

              goal+=1

              for j in range(0,3): #loop dynamic dimensions

                inte = abs(int(sigw\*10000))+1

                if (sigw != 0):

                  www=random.randrange(inte)

                else:

                  www = 1

                if (www < abs(weight[v]\*10000)+1):

                  dim\_com[i][j] = dim\_ini[k][j]

            elif (dd[v] == mmmg):

                  confuse.append([i,k])

            if (goal != max\_goal):

              for l in range(0,max\_goal-goal):

                r = random.randrange(len(confuse))

                for j in range(0,3):

                  inte = abs(int(sigw\*10000))+1

                  if (sigw != 0):

                    www=random.randrange(inte)

                  else:

                    www = 1

                  if (www < abs(weight[v]\*10000)+1):

                    print("y")

                    print(dim\_com[confuse[r][0]][j])

                    print(dim\_ini[confuse[r][1]][j])

                    dim\_com[confuse[r][0]][j] = dim\_ini[confuse[r][1]][j]

                del confuse[r]

        v+=1

  # record initial odd

  dim\_ini = []

  for m in range(0,len(dim\_com)):

    dim\_ini.append([])

    dim\_ini[m] += dim\_com[m]

  #dd = d(t)

  dd=[]

  dis=[]

  for i in range(0, point):

    for j in range(i+1, point):

      ddd = dist(dim\_com[i], dim\_com[j])

      dd.append(ddd)

      dis.append(ddd)

      dis.sort()

  print("dd:"+str(dd))

  #max\_distance

  s=[]

  c=[]

  for i in range(0,point):

    s.append(kisang(i))

    c.append([])

    for j in range(0,point-1):

      c[i].append(dd[s[i][j]])

  s=[]

  for i in range(0, point):

    s.append(max(c[i]))

  #weight

  for i in range(0, len(ed)): #weight

    ppp = ed[i]

    poo = dd[i]

    weight.append(weig(ppp, poo))

  print("weight:"+str(weight))

  #p

  for i in range(t\*10+10, len(weight)):

    vv = abs(weight[i])

    ccc = ccc + vv

  sigw = ccc

  print("sigw:"+str(sigw))

  print("dim\_com"+str(dim\_com))

  #change reverse

  v = 0

  confuse = []

  for i in range(0, point):

    mmmg = s[i]

    goal = 0

    for k in range(i+1, point):

      if (i != k):

        if (k != point-2):

          if (dd[v] < mmmg):

            goal+=1

            for j in range(0,3): #loop dynamic dimensions

              inte = abs(int(sigw\*10000))+1

              if (sigw != 0):

                www=random.randrange(inte)

              else:

                www = 1

              if (www < abs(weight[v]\*10000)+1):

                dim\_com[k][j] = dim\_ini[i][j]

          elif (dd[v] == mmmg):

              confuse.append([i,k])

        else:

          if (i < point-1):

            if (dd[v] < mmmg):

              goal+=1

              for j in range(0,3): #loop dynamic dimensions

                inte = abs(int(sigw\*10000))+1

                if (sigw != 0):

                  www=random.randrange(inte)

                else:

                  www = 1

                if (www < abs(weight[v]\*10000)+1):

                  dim\_com[k][j] = dim\_ini[i][j]

            elif (dd[v] == mmmg):

                confuse.append([i,k])

          elif (i == point-1):

            if (dd[v] < mmmg):

              goal+=1

              for j in range(0,3): #loop dynamic dimensions

                inte = abs(int(sigw\*10000))+1

                if (sigw != 0):

                  www=random.randrange(inte)

                else:

                  www = 1

                if (www < abs(weight[v]\*10000)+1):

                  dim\_com[k][j] = dim\_ini[i][j]

            elif (dd[v] == mmmg):

                  confuse.append([i,k])

            if (goal != max\_goal):

              for l in range(0,max\_goal-goal):

                r = random.randrange(len(confuse))

                for j in range(0,3):

                  inte = abs(int(sigw\*10000))+1

                  if (sigw != 0):

                    www=random.randrange(inte)

                  else:

                    www = 1

                  if (www < abs(weight[v]\*10000)+1):

                    print("y")

                    print(dim\_com[confuse[r][0]][j])

                    print(dim\_ini[confuse[r][1]][j])

                    dim\_com[confuse[r][1]][j] = dim\_ini[confuse[r][0]][j]

                del confuse[r]

        v+=1

  # record initial odd

  dim\_ini = []

  for m in range(0,len(dim\_com)):

    dim\_ini.append([])

    dim\_ini[m] += dim\_com[m]

  #dd = d(t)

  dd=[]

  dis=[]

  for i in range(0, point):

    for j in range(i+1, point):

      ddd = dist(dim\_com[i], dim\_com[j])

      dd.append(ddd)

      dis.append(ddd)

      dis.sort()

  print("dd:"+str(dd))

  #max\_distance

  s=[]

  c=[]

  for i in range(0,point):

    s.append(kisang(i))

    c.append([])

    for j in range(0,point-1):

      c[i].append(dd[s[i][j]])

  s=[]

  for i in range(0, point):

    s.append(max(c[i]))

  #weight

  for i in range(0, len(ed)): #weight

    ppp = ed[i]

    poo = dd[i]

    weight.append(weig(ppp, poo))

  print("weight:"+str(weight))

  #p

  for i in range(t\*10+10, len(weight)):

    vv = abs(weight[i])

    ccc = ccc + vv

  sigw = ccc

  print("sigw:"+str(sigw))

  print("dim\_com"+str(dim\_com))

