ภาคผนวก

ในการทดลองเพื่อสนับสนุนความเหมาะสมของขั้นตอนวิธีเชิงละโมบสำหรับปัญหาการคัดลอกข้อมูลที่น้อยที่สุดที่กำหนดความต้องการของผู้ใช้ (The Minimum Replication Problem with User Preferences) หรือขั้นตอนวิธี MRPUP (MRPUP Algorithm) ได้เลือกใช้ภาษาไพทอน (Python) สำหรับภาษาไพทอลจะใช้เครื่องหมาย # แทนการคอมเมนต์ทั้งบรรทัด ในการทดลองแบ่งโปรแกรมออกเป็นไฟล์ย่อยดังต่อไปนี้

InstanceGenFunc.py

เป็นไฟล์สำหรับฟังก์ชันที่กำหนดโดยผู้ใช้ (user-defined function) สำหรับสร้างกรณีตัวอย่างของปัญหาเพื่อใช้สำหรับการทดลอง โดยในไฟล์มีรายละเอียดดังต่อไปนี้

```
#import python defined function
import random

#generate a set of  m users {1, 2, 3, …, m}
def user_gen (m):
        return set([i for i in range(0,m)])

#generate a set of  n computers {1, 2, 3, …, n}
def comp_gen(n):
        return set([i for i in range(0,n)])

#generate a array of  n empty set {1, 2, 3, …, n}
def phy_init (n):
        return [set() for i in range(0,n)]

#generate a array of  m empty array {1, 2, 3, …, m}
def T_init (m):
        return [[] for i in range(0,m)]

#generate phy function with each users can access to at most max_size computers
def phy_gen (phy,user_size,max_size):
        comp_size = len(phy)
        for i in range(0,user_size):
                for j in range (0,int(random.uniform(1,max_size))):
                        y = int(random.uniform(0,comp_size))
                        phy[y] = phy[y].union(set([i]))
```

```
        #make sure that each computer covers at least 2 users
        for i in range (0,comp_size):
                s = len(phy[i])
                while s < 2:
                        y = int(random.uniform(0,user_size))
                        if y not in phy[i]:
                                phy[i] = phy[i] | set([y])
                                s = s + 1

        return phy

#generate T function for each ussers
def T_gen (T,user):
        for i in range(0,len(T)):
                r = random.sample(user,len(user))

                tmp = r[0]
                index = r.index(i)
                r[0] = i
                r[index]=tmp

                T[i] = r
        return T
```

เพื่อความง่ายในการทดลอง แต่ละกรณีตัวอย่างใช้เซตของตัวเลข แทนเซตของคอมพิวเตอร์และผู้ใช้ เช่น เซตของผู้ใช้ U = {1, 2, 3, 4} ฟังก์ชัน ฟี (phy) และ ที (T) นำเสนอโดยอาเรย์เพื่อความง่าย เช่น phy[2] แทนเซตของผู้ใช้ที่คอมพิวเตอร์ c_2 อนุญาติให้เข้าถึงได้ และ T(1) แทนรายการความต้องการของผู้ใช้ u_1 เป็นต้น

ในแต่ละกรณีตัวอย่าง ฟังก์ชัน ฟี (phy) และ ที (T) จะสร้างอย่างสุ่มโดยใช้ฟังก์ชันของภาษาไฟทอล ซึ่งในที่นี้ใช้ฟังก์ชัน random.uniform(a,b) ซึ่งจะให้ตัวเลข 1 ตัวในช่วง [a,b] อย่างสุ่ม และฟังก์ชัน random.sample(A,a) ที่ให้ค่ากลับเป็นลำดับอย่างสุ่มที่มีขนาด a สำหรับเซต A

Function.py

เป็นไฟล์สำหรับฟังก์ชันที่กำหนดโดยผู้ใช้ (user defined function) ที่จะใช้ในการทดลอง ประกอบด้วยฟังก์ชันดังต่อไปนี้

```
def RangeSet (iSet,fc):
        r = set([])
        for i in iSet:
                r = r | fc[int(i)]
        return r
```

เป็นฟังก์ชันที่รับข้อมูลเข้าเป็นเซตของจำนวนเต็มบวก iSet (สมาชิกของเซตผู้ใช้ และคอมพิวเตอร์ในการทดลองนี้ใช้จำนวนเต็มเพื่อความง่ายในการทดลอง) และอาเรย์ของเซต fc (ฟังก์ชันที่ใช้ในการทดลองใช้ตัวแปรชุดในทางปฏิบัติ) แล้วให้ผลลัพธ์เป็นเรนจ์ของฟังก์ชัน fc เมื่อกำหนดโดเมนคือ iSet

MRUPP.py

ฟังก์ชันการคำนวณต่างๆ ที่เกี่ยวข้องกับปัญหาการคัดลอกข้อมูลที่น้อยที่สุดที่กำหนดความต้องการของผู้ใช้จะอยู่ในไฟล์นี้ โดยในไฟล์มีรายละเอียดดังนี้

```
# Import Function
from Function import *
from copy import deepcopy
from itertools import combinations

# Rho Value Calculation Function
def Rho(t_i,S):
        rhoValue = 0

        for j in S:
                rhoValue = rhoValue + t_i.index(j)

        return rhoValue

# Find a Computer for Cover User u_i with Minimum Rho Value
def MaxPref(comp,u_i,phy,t_i):
        c = set([])
        pick = -1
        minRhoValue = float('inf')

        for j in comp:
                if u_i in phy[int(j)]:
                        c = c | set([j])
```

```
        # u_i has no Preference
        if t_i == []:
                cov = 0
                for j in c:
                        if len(phy[j])>cov:
                                pick = j
                return pick

        for j in c:
                rhoValue = Rho(t_i,phy[j])
                if rhoValue < minRhoValue:
                        pick = j
                        minRhoValue = rhoValue

        return int(pick)

# Overall User Preference calculation function
def OverallPref(user,T,sComp,phy):
        prfValue = 0

        for i in user:
                tmp = set([])
                for j in sComp:
                        if i in phy[int(j)]:
                                tmp = tmp | set([j])

                for c in tmp:
                        prfValue += Rho(T[i],phy[c])

        return prfValue

# Greedy MRUPP Algorithm
def MRUPP(comp_in,user_in,phy_in,T_in):
        comp = deepcopy(comp_in)
        user = deepcopy(user_in)
        phy = deepcopy(phy_in)
        T = deepcopy(T_in)
        Solution = set([])
```

```
        while len(user) > 0:
                u_i = user.pop()
                pick = MaxPref(comp,u_i,phy,T[u_i])
                if pick != -1:
                        comp = comp - set([pick])
                        user = user - phy[pick]
                        Solution = Solution | set([pick])

        return [Solution,OverallPref(user_in,T,Solution,phy)]
#---------------------------------------------------------------------

#brute force algorithm for MRPUP
def BF(comp_in,user_in,phy_in,T_in):
        comp = comp_in
        user = user_in
        phy = phy_in
        T = T_in
        Solution = set([])
        prfValue = float('inf')
        for i in range(1,len(comp)+1):

                tmp = list(combinations(comp,i))

                for j in range (0,len(tmp)):
                        if RangeSet(tmp[j],phy) == user:
                                jPrfValue = OverallPref(user,T,tmp[j],phy)
                                if  jPrfValue < prfValue:
                                        Solution = set(tmp[j])
                                        prfValue = jPrfValue

        return [Solution,prfValue]
```


สำหรับการคำนวณค่าความต้องการของผู้ใช้ และขั้นตอนวิธี MRPUP จะคำนวณดังที่นิยามไว้ในบทที่ 4 และขั้นตอนวิธีแบบบรูทฟอร์สจะใช้วิธีการตรวจสอบทุกกรณีของการจัดหมู่ (combination) และเลือกกรณีที่ให้ค่าความต้องการของผู้ใช้ทั้งหมดน้อยที่สุดเป็นคำตอบ

Experiment.py

เป็นไฟล์หลักสำหรับดำเนินการทดลอง ตั้งแต่กำหนดกำหนดคุณสมบัติของกรณีตัวอย่างที่จะใช้ในการทดลอง และเรียกใช้ฟังก์ชันต่างๆ เพื่อเก็บผลการทดลองที่ได้จากขั้นตอนวิธีเชอง

ละโมบ และขั้นตอนวิธีแบบบรูทฟอร์สสำหรับนำมาเปรียบเทียบกันต่อไป โดยรายละเอียดของการทดลองดังต่อไปนี้

```
# Import Part
from InstanceGenFunc import *
from MRUPP import *
import random
import time

st = time.time()

#Start Experiment
# 20 USER
user_num = 20
comp_num = random.uniform(1,25)

x = open(str(user_num),'w')
for i in range (0,20):
   user = user_gen(user_num)
   comp = comp_gen(comp_num)
   phy = phy_gen(phy_init(comp_num),len(user),comp_num)
   T = T_gen(T_init(len(user)),user)

   gr = MRUPP(comp,user,phy,T)
   bf = BF(comp,user,phy,T)
   frac = float(gr[1])/float(bf[1])

   #Write to Files
   x.writelines("CASE "+str(i+1)+'\n')
   x.writelines("user = "+str(user)+'\n')
   x.writelines("comp = "+str(comp)+'\n')
   x.writelines("phy = "+str(phy)+'\n')
   x.writelines("T = "+str(T)+'\n\n')
   x.writelines("Result\n")
   x.writelines("GR   :"+str(gr)+'\n')
   x.writelines("BF   :"+str(bf)+'\n')
   x.writelines("Frac :"+str(frac)+'\n')
   x.writelines("-----------------------\n\n")

x.close()
```

```
# 30 USER
user_num = 30
comp_num = random.uniform(1, 25)

x = open(str(user_num),'w')
for i in range (0,20):
   user = user_gen(user_num)
   comp = comp_gen(comp_num)
   phy = phy_gen(phy_init(comp_num),len(user),comp_num)
   T = T_gen(T_init(len(user)),user)


   gr = MRUPP(comp,user,phy,T)
   bf = BF(comp,user,phy,T)
   frac = float(gr[1])/float(bf[1])


   #Write to File
   x.writelines("CASE "+str(i)+'\n')
   x.writelines("user = "+str(user)+'\n')
   x.writelines("comp = "+str(comp)+'\n')
   x.writelines("phy = "+str(phy)+'\n')
   x.writelines("T = "+str(T)+'\n\n')
   x.writelines("Result\n")
   x.writelines("GR   :"+str(gr)+'\n')
   x.writelines("BF   :"+str(bf)+'\n')
   x.writelines("Frac :"+str(frac)+'\n\n')

x.close()
```

```
# 40 USER
user_num = 40
comp_num = random.uniform(1, 25)

x = open(str(user_num),'w')
for i in range (0,20):
   user = user_gen(user_num)
   comp = comp_gen(comp_num)
   phy = phy_gen(phy_init(comp_num),len(user),comp_num)
   T = T_gen(T_init(len(user)),user)


   gr = MRUPP(comp,user,phy,T)
   bf = BF(comp,user,phy,T)
   frac = float(gr[1])/float(bf[1])


   x.writelines("CASE "+str(i)+'\n')
   x.writelines("user = "+str(user)+'\n')
   x.writelines("comp = "+str(comp)+'\n')
   x.writelines("phy = "+str(phy)+'\n')
   x.writelines("T = "+str(T)+'\n\n')
   x.writelines("Result\n")
   x.writelines("GR   :"+str(gr)+'\n')
   x.writelines("BF   :"+str(bf)+'\n')
   x.writelines("Frac :"+str(frac)+'\n\n')

x.close()
```

```
# 50 USER
user_num = 50
comp_num = random.uniform(1, 25)

x = open(str(user_num),'w')
for i in range (0,20):
   user = user_gen(user_num)
   comp = comp_gen(comp_num)
   phy = phy_gen(phy_init(comp_num),len(user),comp_num)
   T = T_gen(T_init(len(user)),user)


   gr = MRUPP(comp,user,phy,T)
   bf = BF(comp,user,phy,T)
   frac = float(gr[1])/float(bf[1])

   x.writelines("CASE "+str(i)+'\n')
   x.writelines("user = "+str(user)+'\n')
   x.writelines("comp = "+str(comp)+'\n')
   x.writelines("phy = "+str(phy)+'\n')
   x.writelines("T = "+str(T)+'\n\n')
   x.writelines("Result\n")
   x.writelines("GR   :"+str(gr)+'\n')
   x.writelines("BF   :"+str(bf)+'\n')
   x.writelines("Frac :"+str(frac)+'\n\n')

x.close()
```

```
# 60 USER
user_num = 60
comp_num = random.uniform(1, 25)


x = open(str(user_num),'w')
for i in range (0,20):
   user = user_gen(user_num)
   comp = comp_gen(comp_num)
   phy = phy_gen(phy_init(comp_num),len(user),comp_num)
   T = T_gen(T_init(len(user)),user)


   gr = MRUPP(comp,user,phy,T)
   bf = BF(comp,user,phy,T)
   frac = float(gr[1])/float(bf[1])


   x.writelines("CASE "+str(i)+'\n')
   x.writelines("user = "+str(user)+'\n')
   x.writelines("comp = "+str(comp)+'\n')
   x.writelines("phy = "+str(phy)+'\n')
   x.writelines("T = "+str(T)+'\n\n')
   x.writelines("Result\n")
   x.writelines("GR   :"+str(gr)+'\n')
   x.writelines("BF   :"+str(bf)+'\n')
   x.writelines("Frac :"+str(frac)+'\n\n')

x.close()
```

ประวัติผู้เขียน

ชื่อ - สกุล	นาย วิริยะ เตชะปลูก
วัน เดือน ปี	23 พฤษจิกายน 2529
ประวัติการศึกษา	สำเร็จการศึกษาวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยเชียงใหม่ ปีการศึกษา 2551 สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลาย โรงเรียนลำปางกัลยาณี ปีการศึกษา 2547
ผลงานวิจัย	W. Techaploog and S. Kantabutra. "The Hardness of the Problems in the System Administration Model," In *Proceedings of 2011 International Conference on Information and Electronics Engineering (ICIEE 2011)*, May 28-29, 2011,Bangkok, Thailand. W. Techaploog and S. Kantabutra. "Graph Relabeling with Privileged Edge Labels," In *Proceedings of the 6th IEEE International Conference in Electrical Engineering / Electronics, Computer, Telecommuni-cations, and Information Technology (ECTI-CON 2009)*, May 6-9, 2009, Pattaya, Thailand.