混合粒子群與細菌覓食演算法解決容量限制車輛途程問題

高有成
大同大學資訊經營所
ykao@ttu.edu.tw

張怡凡
大同大學資訊經營所
rexmax0912@gmail.com

摘要

為滿足企業的物流配送需求，車輛路線安排與規劃就越顯重要，車輛途程問題的目標是希望找出各車的容積都能符合且行車總距離又達到最短，以節省成本並提高效率。車輛途程是一種組合最佳化問題，其求解難度更是屬於 NP-Hard，當問題規模越來越大時就很難找到確切解，所以希望透過巨集啟發式演算法在有效率的時間內找到品質精確的近似最佳解。本研究利用粒子群演算法的全域搜索能力和細菌演算覓食法的區域搜尋能力相結合來期望產生不錯的解的品質和程式執行效率。實驗結果顯示，本研究所提出之演算法能夠有效解決容量限制下之車輛途程問題。

關鍵詞：車輛途程問題、粒子群最佳化演算法、細菌覓食演算法

混合粒子群與細菌覓食演算法解決容量限制車輛途程問題

壹、簡介

組合最佳化問題的種類繁多常見的包含車輛途程問題、單元形成問題、投資組合最佳化問題等等。車輛途程問題(Vehicle Routing Problem)是眾多排列組合最佳化問題中的一門,在運輸領域上是相當實用且重要的一個客題。於 1959 年由 Dantzig and Ramser[6]所提出後，目前已發展出多種應用形式。2008 年 Jozefowiez 等人[7]針對問題定義、目標、演算法作了詳細的分類與比較。而最典型的車輛途程問題就是容量限制下車輛途程問題(Capacitated Vehicle Routing Problem，CVRP)。其中容量限制車輛途程問題的目標是要在容量符合的情況下總里程數要最小並且要拜訪完所有的客戶，限制條件為每輛車均須自場站出發並返回場站、每位客戶恰好由某輛車服務一次、每輛車的最大裝載容量不得超過最大限制容量。解容量限制下車輛途程問題的演算法通常會遇到一些困難：演化後的解違反限制式、陷入區域最佳解、收斂速度，計算時間過長等等。

由於車輛途程問題的解題難度屬於NP-hard，如果使用精確求解演算法(exact algorithms)很難在有效率的時間內找到一組近似最佳解，而巨集啟發式演算法(meta-heuristic algorithms)的特徵就是演化的解會逐漸朝向更佳的方向前進，且有一定能力搜索其他未探索過的區域，避免最終陷入過早收斂又跳不出區域最佳解的情況。巨集啟發式演算法包括基因演算法(Genetic Algorithm，GA)、模擬退火法（Simulated Annealing，SA）、禁忌搜尋法(Tabu Search，TS)、蟻群最佳化演算法 (Ants Colony Optimization，ACO)、粒子群最佳化演算法(Particle Swarm Optimization，PSO)、細菌覓食最佳化(Bacterial Foraging Optimization，BFO)等等。

其中如果用解的數量來進行區分，則可以將上述演算法區分為單一解和群體解兩類，模擬退火法和禁忌搜尋法屬於單一解；基因演算法、蟻群最佳化演算法、粒子群最佳化演算法、細菌覓食最佳化、蜜蜂最佳化演算法屬於群體解而這些最佳化演算法也被稱為群體智慧演算法。

1995年Alex學者[15]使用SA並結合區域搜尋將途程分為兩途程間交換顧客成員或是單途程的交換顧客順序，以此做較大幅度的變動希望解決過早陷入區域最佳的機率。2006年R. Tavakkoli-Moghaddam等學者[17]將SA和最近鄰居法、1-opt、2-opt結合都是為了盡量增加解的搜索範圍。

1998年P. Augerat等學者[16]將TS結合結構化啟發式演算法和貪婪法藉此導引產生更好的解，但是只有單一解在演化的情況下還是使結果受到很大受限。2009年Shih-Wei Lin等學者[14]在SA的架構下結合TS中的禁忌名單特色，使其搜尋過程適當地避免一些重複的操作，而SA也會依照演化的早期和晚期調整接受較差解的機率，使其有機會開拓其它的區域，避免陷入區域最佳。但其演算法只有一組解進行演化並非群體智慧，所以結果的好壞十分依賴初始解的品質。

以上方法的解的品質都很大程度依賴初始解上，如果初始解區域離真實最佳解區域很遠時，往往很難透過演化跳脫區域最佳，所以群體智慧演算法改善了這缺點，他除了對初始解的要求不會這麼嚴苛之外，還有著演化學習機制可以擺脫掉陷入區域最佳。

2003年Baker等人[8]提出以基因演算法解容量限制下車輛途程問題，該演算法對每個初始解先採用結構化的啟發式演算法產生不錯的初始解，在利用基因演算法的天擇、交配、突變等機制演化出不錯的解。2004年John E. Bell等學者[18]提出多蟻群機制，讓每群螞蟻都代表1輛車且擁有各自的費洛蒙，能夠讓解呈現多樣性。2005年Liu Zhishuo等學者[19]也使用了多蟻群機制，但有多蟻群費洛蒙更新公式，讓不同群的螞蟻也能夠相互整合費洛蒙資訊。

2006 年 Chen 等人[2]提出 DPSO 及 SA 之混合方法，以 DPSO 處理客戶分群，SA 則處理各車拜訪客戶順序，解的表達長度為車輛數x客戶數，其缺點為當遇大型問題時，因為解的長度冗長，浪費許多存儲空間且計算耗時。2009 年 Ai 等人[3]提出以 PSO 為架構、兩種解的表達及對應之編碼方法，兩種解的表達方式分別為 SR-1 及 SR-2。其方法將各群中心及半徑值以 GLNPSO 進行演化以處理客戶的分群來將各粒子引導，並透過多種區域搜尋法調整至最佳解位置，區域搜尋法有(2-opt，1-1 exchange，1-0 exchange)。優點為分群觀念簡單、計算時間快速、解的品質優。2010 年 Byung-In Kim 等人[10]提出以機率矩陣模式的 PSO，讓一組解產生的可能性不止一種，以此達到解的多樣化並有效避免過早收斂，然後使用區域搜尋法(2-opt，Or-opt，1-1 exchange，1-0 exchange) 等來調整解的品質。2012 年 BaBak Farhang Moghaddam 等人[11]提出一種離散模式的 PSO 可以在演化過程中同時分群和排序，然後使用 2-opt、1-1 exchange、變動鄰域搜尋法(Variable Neighborhood Search，VNS)、迭代貪婪法(Iterated Greedy (IG) algorithm)等來增大解的多樣性避免陷入過早收斂的窘境。

PSO 與 GA 不同之處在於他擁有記憶性可以記錄歷來最佳和自身最佳，讓演化過程可以擁有一定依據；與 ACO 不同之處在於螞蟻需要在演化過程中散布費洛蒙並累積一定的世代後才會產生影響力，此特性影響了 ACO 的收斂速度而 PSO 則無此限制。原始的 PSO 靠著自身最佳經驗和全體最佳經驗可以在廣大的區域中快速收斂到一組不錯的解，這在最佳化中是很重要的特點，因為解的品質和計算時間同樣的重要，這也就是 PSO 跟其他演算法相較之下的優勢，但有時太快的收斂有可能錯過真正最佳的區域，於是本研究將提出以粒子群演算法進行顧客分群，模擬退火演算法進行顧客排序，再搭配細菌覓食演算法機制加強區域搜尋，應用在容量限制下車輛途程問題，幫助 PSO 在收斂的過程中也能維持一定程度的區域搜尋而不會錯過了真正的最佳區域，架構以先分群後排序的方式期望能得到不錯的成果。

貳、重要演算法簡介

因為啟發式演算法繁多，以下會重點介紹在本論文中使用到的演算法：粒子群最佳化演算法、細菌覓食最佳化、模擬退火法。

PSO 是由 Kennedy 與 Eberhart 於 1995 年提出[1]，受到鳥群集體飛行尋找食物行為的啟發，每組解以粒子為名，在連續空間中飛行演化，於 1998 年 Kennedy 與 Shi 加入

慣性權重(inertia weight)後，終成為 PSO 的標準版本。PSO 最大的優點為收斂快速。其演化規則如公式(1)和(2)：

$$V_{id} = W \times V_{id} + C_1 \times Rand \times (P_{best} - X_{id}) + C_2 \times Rand \times (G_{best} - X_{id}) \quad (1)$$

$$X_{id} = X_{id} + V_{id} \quad (2)$$

本研究使用的組合式粒子群最佳化演算法(Combinatorial Particle Swarm Optimization, CPSO)，是為解離散型組合最佳化問題所設計，由Jarboui等人於2007年提出[12]。它在實際與虛擬兩種空間中不斷的進行轉換與演化。此法不僅保有PSO的演化特性，更將原本只能應用於連續空間的限制打破使之能應用到離散空間。

BFO自2002年由Passino [4]提出，受到大腸桿菌在腸胃中的覓食行為有所啟發，每組解以細菌為名，在連續的空間中進行趨化(chemotaxis)、複製(reproduction)、驅散(elimination and dispersal)等三個過程，細菌向富有營養區域聚集的行為稱為趨化。在趨化過程中，細菌運動模式包括翻轉(tumble)和前進(run)。細菌向任意方向移動單位步長定義為翻轉，當細菌完成一次翻轉後，若適應值得到改善，將沿同一方向繼續移動數步，直至適應值不再改善，或是達到預定的移動步數臨界值，此過程定義為前進。一但達到臨界趨化次數，細菌將進行複製，以趨化過程中各細菌適應值為標準，較差的半數細菌死亡，較好的半數細菌分裂成兩個子細菌。子細菌將繼承母細菌相同的位置及步長，複製過程中細菌總數保持不變。趨化過程可確保細菌的區域搜索能力，複製過程能加快細菌的搜索速度，但對於複雜的最佳化問題，趨化和複製無法避免細菌陷入區域最佳的現象發生。藉著驅散過程以加強演算法中全域搜索能力。細菌在完成一定次數的複製後，將以一定機率被驅散到搜索空間中任意位置。

2011 年 Miao Wan 等學者[5]提出以細菌演算法解決資料分群問題，以群中心做為解的表達，透過趨化(chemotaxis)中的翻滾(Tumble)和前進(Run)來進行解的演化並以複製(reproduction)加快收斂速度最後以驅散(elimination and dispersal)避免陷入區域最佳解，其實驗結果顯示細菌覓食最佳化應用於分群上有著不錯的成果。

SA最早由Kirkpatrick等人於1983年提出 (Osman, 1993)，為巨集啟發式演算法的一種。一般的區域搜尋法只接受更優的鄰域解(neighboring solution)，但SA給予一定機率接受未改善的鄰域解，增加多元性，但如果較好的區域離目前較遠時，SA通常無法跳出區域最佳到更好的地方，現在則常用於初始解的建構或是後期區域搜尋的微調。

參、數學模型

本節將介紹容量限制下車輛途程問題的目標函數及限制式之數學模型。

一、 符號

0：場站；

N：顧客數；

K：車輛數；

C_{ij}：顧客 i 到顧客 j 的成本；

S_i：在顧客 i 的服務時間；

Q：每輛車最大車容量限制；

T：每輛車最大里程數限制；

d_i：顧客 i 的需求；

X_{ij}^k：由客戶 i 至 j 的旅途是否由車輛 k 所服務；

p：違反限制的懲罰係數；

二、 目標函數

$$\text{Minimize } \sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=1}^{K} C_{ij}X_{ij}^k + p\sum_{k=1}^{K}\max\{0, \sum_{i=0}^{N}\sum_{j=0}^{N} X_{ij}^k d_i - Q\} + \max\{0, \sum_{i=0}^{N}\sum_{j=0}^{N} X_{ij}^k (C_{ij} + S_i) - T\} \quad (3)$$

三、 限制式

$$\sum_{k=1}^{K}\sum_{i=0}^{N} X_{ij}^k = 1, j = 1,2,\dots,N \quad (4)$$

$$\sum_{k=1}^{K}\sum_{j=0}^{N} X_{ij}^k = 1, i = 1,2,\dots,N \quad (5)$$

$$\sum_{i=0}^{N} X_{iu}^k - \sum_{j=0}^{N} X_{uj}^k = 0, k = 1,2,\dots,K; u = 1,2,\dots,N \quad (6)$$

$$\sum_{i=0}^{N}\sum_{j=0}^{N} X_{ij}^k d_i \le \mathrm{Q}, k = 1,2,\dots,\mathrm{K} \quad (7)$$

$$\sum_{i=0}^{N}\sum_{j=0}^{N} X_{ij}^k (C_{ij} + S_i) \le T, k = 1,2,\dots,\mathrm{K} \quad (8)$$

$$\sum_{j=1}^{N} X_{ij}^k = \sum_{j=1}^{N} X_{ji}^k \le 1, i = 0;\ k = 1,2,\dots,\mathrm{K} \quad (9)$$

$$X_{ij}^k \in \{0,1\}, \mathrm{i,j} = 0,1,\dots,\mathrm{N}; \mathrm{k} = 1,2,\dots,\mathrm{K} \quad (10)$$

公式(3)：總成本最小化，若有車輛違反容量和總里程數限制時，則會進行懲罰。所以當違反的量越多時，成本則會越高。此方法參照 2011 年 W.Y. Szeto 等學者[9]之目標函數，但本論文的懲罰係數通用於車容量與里程數，所以在此將公式簡化

公式(4)、(5)：每位顧客只能被一輛車所服務。

公式(6)：維持每輛車進出的連續性。

公式(7)：每輛車所服務的總需求不可超過最大車容量限制。

公式(8)：每輛車所行駛的總里程數不可超過最大里程數限制。

公式(9)：確保每輛車必須從場站出發 1 次(或 0 次)並返回場站 1 次(或 0 次)。

公式(10)：顧客是否由 k 車所服務。1 代表是；0 代表否。

肆、方法論

本論文所提出的方法論混合CPSO及BFO兩種演算法，命名為CPSBFO，以兩階段的先分群後路徑排序方式解容量限制下車輛途程問題，其中CPSO用來處理客戶分群問題，分群後的各途程以BFO安排區域搜尋及客戶順序拜訪。

一、 解的表達

方法論是以兩階段的先分群後排序方式解容量限制下車輛途程問題，表1表示各客戶由哪輛車來服務，所以由表可知客戶1由第一台車服務，客戶2由第二台車服務，客戶3由第二台車服務以此類推；表2表示分群後的客戶拜訪順序，表2表示客戶1、6由第一

車所服務，客戶2、3、5、9由第二車所服務，顧客4、7、8由第三車所服務。解字串長度為顧客數(N)+車輛數(K)+1，客戶拜訪順序則依照解字串由左至右，0表示場站。

表 1 客戶分群解的表達舉例說明

客戶編號	1	2	3	4	5	6	7	8	9
指派車號	1	2	2	3	2	1	3	3	2

表 2 途程解的表達舉例說明

途程解	0	1	6	0	2	3	5	9	0	4	7	8	0

二、 CPSO 應用於客戶分群

符號：

iter：表示目前演化的世代數；

p、d：分別表示粒子編號、維度編號；

y_{pd}^{iter} ：第*iter*世代解p的維度d(即客戶d被指派的車號)之轉換變數；

G_{d}^{iter} ：第*iter*世代全域最佳解的維度d值(即被指派的車號)；

P_{pd}^{iter} ：第*iter*世代解p最佳解的維度d值；

xv_{pd}^{iter} ：第*iter*世代解p的維度d值；

c1，c2：均為參數，分別表示全域最佳解及粒子最佳解產生新解的重要程度；

v_{pd}^{iter} ：第*iter*世代解p的維度d移動的速度；

r1，r2：分別表示介於0與1間的隨機亂數；

λ_{pd}^{iter} ：第*iter*世代解p的維度d經演化後的值；

α ：為一閥值，用來控制產生的新解之深度及多樣性。

演化公式：

$$y_{pd}^{iter} = \begin{cases} 1 & \text{if } (xv_{pd}^{iter-1} = G_{d}^{iter-1}) \\ -1 & \text{if } (\text{xv}_{\text{pd}}^{\text{iter -1}} = P_{pd}^{iter-1}) \\ -1 \text{ or } 1 \text{ randomly,} & \text{if } (\text{xv}_{\text{pd}}^{\text{iter -1}} = P_{pd}^{iter-1} = G_{d}^{iter-1}) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$d_1 = -1 - y_{pd}^{iter}$ 表示 xv_{pd}^{iter-1} 與 P_{pd}^{iter-1} 的距離 (12)

$d_2 = 1 - y_{pd}^{iter}$ 表示 xv_{pd}^{iter-1} 與 G_{d}^{iter-1} 的距離 (13)

$$v_{pd}^{iter} = w \cdot v_{pd}^{iter-1} + c1 \cdot r1 \cdot d1 + c2 \cdot r2 \cdot d2 \tag{14}$$

$$\lambda_{pd}^{iter} = y_{pd}^{iter} + v_{pd}^{iter} \tag{15}$$

$$y_{pd}^{iter} = \begin{cases} 1 & \text{if } (\lambda_{pd}^{iter} > \alpha) \\ -1 & \text{if } (\lambda_{pd}^{iter} < -\alpha) \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

$$xv_{pd}^{iter} = \begin{cases} G_d^{iter-1} & \text{if } (y_{pd}^{iter} = 1) \\ P_{pd}^{iter-1} & \text{if } (y_{pd}^{iter} = -1) \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

由公式(11)至(17)中可以看出CPSO在實際與虛擬兩種空間中不斷的進行轉換與演化。公式(11)由上一世代全域最佳解、粒子本身最佳解與粒子值比較後產生轉換變數。另將速度乘上慣性向量後，加上由 XV 值分別與粒子本身最佳解、全域最佳解計算得到的距離d1、d2，依公式(14)計算得到新速度，公式(15)將轉換變數加上新速度，公式(16)透過一個閥值將轉換變數切成三個區間，分別賦予不同的值，再由公式(17)來決定要賦予全域最佳、區域最佳或是其他。

三、 SA 應用於客戶排序

符號：

$f(S')$、$f(S)$：分別表示新解、原解的目標函數值；

t、t_0、t_f：分別表示目前溫度、初始溫度、終止溫度；

L：溫度長度，為每個t值執行搜尋鄰域解的次數；

θ：冷卻率，介於0與1間的實數，但不包含0及1；

rand：介於0與1間的亂數。

公式：

$$\Delta = f(S') - f(S) \quad (18)$$

$$p(S') = \exp(-\Delta / t) \quad (19)$$

SA於設定各參數後，在各途程中，隨機選取兩客戶交換其拜訪順序，若客戶交換後產生的新解其目標函數值小於原解，則取代原解；若新解目標函數值大於原解，則依公式(19)以一定機率給新解取代原解的機會。

四、 BFO 加強區域搜尋

演化完畢後車輛容積有可能會違反限制式，在此以BFO的結構結合修補法對超過容積限制的車輛進行移除顧客的動作，移除顧客的規則為以 prop 的機率選上距離該車之群中心最遠的客戶，(1-prop) 則以隨機選取，直到該車符合容量限制。被移除的客戶則列入未被指派車輛之客戶集合。直至所有車都符合容量限制時再針對未被指派車輛之客戶集合進行指派車號，以不違反容量限制的前提下，每個客戶對其能被指派的車算其與車之群中心距離，對每個能指派的車都計算完群中心距離後以輪盤法來選擇此客戶應該指派至何車，離群中心越近機率越大。修補完後再使用1-0 exchange、1-1 exchange和SA做區域搜尋如果有找到更好的解則取代之。

五、 CPSBFO 演算法

本研究提出在BFO之架構下結合CPSO，主要以CPSO產生分群解字串，而BFO負責收斂速度和區域搜尋的控制，每次趨化每隻細菌最多可以得到MaxStepLength次演化分群和區域搜尋的機會，如果此次的演化方向不好則停止演化換下一隻細菌，當趨化次數到達臨界值，進行複製步驟將精英解分裂讓整體品質提高，當複製次數到達上限則進行驅散步驟，驅散會以一定機率將細菌驅趕至未探索的區域以確保全域搜尋能力。高階虛

擬碼可見圖1。

```
1: Initialize Cluster.
2: FOR Number of Elimination/Dispersal Steps
3:    FOR Number of Reproduction Steps
4:          FOR Number of Chemotactic Steps
5:                FOR Each Bacterium i ∈ B
6:                      WHILE m < MaxStepLength
7:                            CPSO(i);
8:                            Local Search(i);
9:                            IF fitness(t+1) < fitness(t)
10:                                 m = m+1.
11:                           END IF
12:                           ELSE
13:                                 m = MaxSwimLength.
14:                           END ELSE
15:                     ENDWHILE
16:               END FOR
17:         END FOR
18:         Reproduce().
19:   END FOR
20:   Eliminate and disperse().
21: END FOR
```

圖1 CPSBFO演算法虛擬碼

伍、實驗及比較

本節將驗證本研究所提出的演算法效能，並與其它文獻進行比較。本論文程式由 Java 所撰寫，執行於 Intel Core 2 CPU T7500 2.20GHz with 2G RAM 環境。

在參數設定方面：母體數 pop=n、w=0.8、C_1=1.4、C_2=1.4、prop = 0.7、Ned=3、Nre=3、Nc=100、stepSize=c，演化迴圈(世代)=Ned× Nre × Nc；t_0 = 10、t_f =0.01、L=$n \times 2$ 、θ = 0.8 。n 指客戶數，c 指車數。

題庫位於網址 http://www.branchandcut.org/VRP/data/，表 3 為與其他演算法比較之實驗結果，計算時間表示找到最佳解的最短時間。本演算法比起其他文獻除了能夠解出較多的題目外，所花費的計算時間亦不會太多。

DPSO-SA 的解字串非常長，容易為此花費許多演化時間，在演化過程中也無法避免非法解的出現，導致很多解必須重新產生，使演化過程付之一炬。SR-2 使用群中心模式來演化，避免掉了許多 PSO 在連續轉離散時的困難，但是對於一些客戶點非群聚時的情況可能會遇到困難，CPSO-SA 在區域搜尋的部分比較沒有辦法做大幅度的變動，

這很有可能提前收斂導致陷入區域最佳解。

陸、結論

本研究提出的方法論以 CPSO 結合 BFO 進行 CVRP 的求解，客戶分群解的長度只有顧客數長，可以在演化時縮短運算時間，產生不合理解時的判斷及修正也會較容易。單獨使用 CPSO 的情況下容易造成收斂過快，跳脫不出區域最佳解的情況，所以結合 BFO 強大的區域搜索特性和驅散來解決早熟問題。實驗結果證明 CPSBFO 能持續且有效率的探索並且在解的品質和計算時間上與其他論文相比都有更優異的成果。

在此研究中 CPSO 只針對客戶分群做演化，因為只要分群不同那客戶排序就會不同，在每次演化分群後都必須再做一次客戶排序，未來如果能將 CPSO 同時具備分群和排序能力，讓排序也具有記憶性，應可將解的品質再次提升。

表 3 題庫實驗結果

No.	問題來源	N	K	目標函數 / 計算時間(秒)				
				BKS	DPSO-SA[2]	SR-2[3]	CPSO-SA[13]	CPSBFO
1	A-n33-k5	32	5	661	**661*** / 32.2	**661*** / 13	**661*** / 0.7	**661*** / 0.11
2	E-n30-k3	29	3	534	**534*** / 28.4	**534*** / 16	**534*** / 0.3	**534*** / 0.04
3	A-n46-k7	45	7	914	**914*** / 128.9	**914*** / 23	917 / 2.4	**914*** / 0.48
4	E-n51-k5	50	5	521	528 / 300.5	**521*** / 22	**521*** / 4.6	**521*** / 0.46
5	F-n72-k4	71	4	237	244 / 398.3	**237*** / 53	**237*** / 5.3	**237*** / 1.26
6	M-n121-k7	120	7	1034	1038 / 1733.5	1036 / 89	1039 / 51.5	**1034*** / 38.65

說明：BKS 表示題庫中目前已知最佳解。

DPSO-SA 使用 Intel Pentium IV CPU 1.8 GHz with 256M RAM。

SR-2 使用 Intel Pentium IV CPU 3.4 GHz with 1 GB RAM。

CPSO-SA 使用 Intel Core 2 CPU E8400 3GHz with 3.5G RAM。

CPSBFO 使用 Intel Core 2 CPU T7500 2.20GHz with 2G RAM。

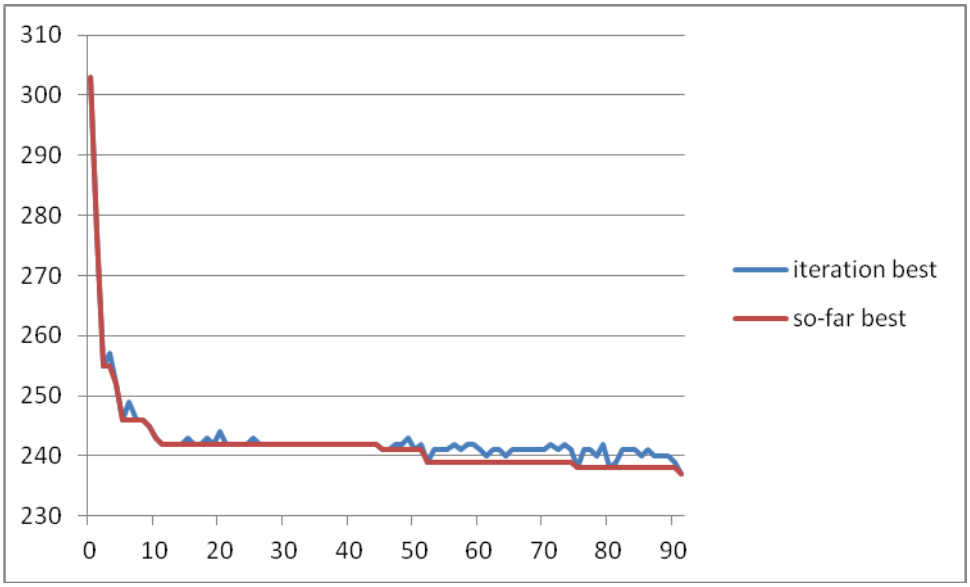

圖2 F-n72-k4 收斂圖

參考文獻

1. Kennedy, J. and Eberhart, R. “Particle swarm optimization” Proceedings of *IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.
2. Chen, A.L., Yang, G.K. and Wu, Z.M. ”Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem” *Journal of Zhejiang University* SCIENCE A (7:4), 2006, pp. 607-614.
3. Ai, T.J., Kachitvichyanukul, V. ”Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem” *Computers & Industrial Engineering* (56:1), 2009, pp. 380-387.
4. Kevin M. Passino, “Biomimicry of bacterial foraging for distributed optimization and control” *IEEE Control Systems* (22:3), 2002, pp. 52-67.
5. Miao Wan, Lixiang Li, Jinghua Xiao, Cong Wang and Yixian Yang. “Data clustering using bacterial foraging optimization” *Journal of Intelligent Information* (38:2), 2011, pp.321-341.
6. Dantzig, G. B. and Ramser, R.H., “The truck dispatching problem” *Management Science* (6:1), 1959, pp. 80–91.
7. Jozefowiez, N., Semet, F. and Talbi, E.G. ”Multi-objective vehicle routing problems” *European Journal of Operational Research* (189:2), 2008, pp. 293-309.
8. Baker, B.M. and Ayechew, M.A. “A genetic algorithm for the vehicle routing problem” *Computers & Operations Research* (30:5), 2003, pp. 787-800.
9. W.Y. Szeto, Yongzhong Wu, Sin C. Ho, “An Artificial Bee Colony Algorithm for the

Capacitated Vehicle Routingnext term Problem" *European Journal of Operational Research* (215:1), 2011, pp. 126-135.

10. Byung-In Kim and So-Jung Son, "A probability matrix based particle swarm optimization" *Journal of Intelligent Manufacturing* , Published online:10 September 2010
11. Babak Farhang Moghaddam, Rubén Ruiz, Seyed Jafar Sadjadi, "Vehicle routing problem with uncertain demands An advanced particle swarm algorithm" *Computers & Industrial Engineering* (62:1), 2012, pp. 306–317.
12. Jarboui, B., Cheikh, M., Siarry, P. and Rebai, A. "Combinatorial particle swarm optimization (CPSO) for partitional clustering problem" *Applied Mathematics and Computation* (192:2) 2007, pp. 337-345.
13. Y. Kao and M. Chen, "A hybrid PSO algorithm for the CVRP Problem" in Proceedings of ECTA 2011 - *International Conference on Evolutionary Computation Theory and Applications*, Paris, France, pp. 539-543.
14. Shih-Wei Lin, Zne-Jung Lee, Kuo-Ching Ying, Chou-Yuan Lee, "Applying hybrid meta-heuristics for capacitated vehicle routing problem" *Expert Systems with Applications* (36:2) , 2009, pp. 1505–1512.
15. Alex Van Breedam, "Improvement heuristics for the Vehicle Routing Problem based on Simulated Annealing" *European Journal of Operational Research* (86:3), 1995, pp. 480-490.
16. P. Augerat, J.M. Belenguer, E. Benavent, A. Corberin, D. Naddef , "Separating capacity constraints in the CVRP using tabu search" *European Journal of Operational Research* (106:2-3), 1998, pp. 546-557.
17. R. Tavakkoli-Moghaddam, N. Safaei, Y. Gholipour, "A hybrid simulated annealing for capacitated vehicle routing problems with the independent route length" *Applied Mathematics and Computation* (176:2), 2006, pp. 445–454.

Hybrid PSO and BFO for the CVRP

Yucheng Kao
Department of Information Management, Tatung University
ykao@ttu.edu.tw

Yi-Fan Chang
Department of Information Management, Tatung University
rexmax0912@hotmail.com

Abstract

Vehicle routing problems (VRP) becomes more and more important in modern logistics management. The goal of capacitated vehicle routing problems (CVRP) is to minimize the total distance of the routes under the constraints of vehicle's capacity, in order to save costs and improve efficiency. CVRP is a combinatorial optimization problem, and it is a kind of NP-hard problem. It is difficult to find exact solutions to large-size CVRPs. Recently meta-heuristics are adopted to find optimal or near optimal solutions to CVRPs within reasonable time. This paper aims to propose a hybrid algorithm combining combinatorial particle swarm optimization (CPSO) and bacterial foraging optimization (BFO) for solving CVRP. The experimental results show that the proposed algorithm is an effective approach for solving the CVRP.

Keywords: capacitated vehicle routing problem, Particle Swarm Optimization, Bacterial Foraging Optimization