# Brock University

Department of Computer Science

**Discrete PSO for the Uncapacitated Single Allocation Hub Location Problem**

Alexander Bailey, Beatrice Ombuki-Berman, and Stephen Asobiela
Technical Report # CS-13-05
January 2013

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

# Discrete PSO for the Uncapacitated Single Allocation Hub Location Problem

Alexander Bailey, Beatrice Ombuki-Berman, Stephen Asobiela
Department Of Computer Science
Bio-Inspired Computational Intelligence Group
Brock University, St. Catharines, Ontario, Canada, L2S 3A1
Email: bombuki@brocku.ca

***Abstract*—Efficient network design and optimization of transportation and distribution systems has significant economic and service efficiency implications for both the public and private sectors. We propose a new solution based on a particle swarm optimization (PSO) for the uncapacitated single allocation hub location problem (USAHLP). Although various meta-heuristics have been proposed for this problem, to the authors' knowledge, this is the first attempt to use a PSO framework for this problem. An empirical study is done using well-known benchmark problems from the Civil Aeronautics Board and Australian Post data sets with node sizes of up to 200. The beauty of the proposed approach is its simplicity and effectiveness, where its solution quality is compared with that of other meta-heuristics. The proposed discrete PSO matches or out-performs the solution quality of the current best-known methods for the USAHLP.**

## I. Introduction

Hub location problems (HLPs) are concerned with the optimality of telecommunication and transhipment networks, and seek to minimize the cost of transportation or construction. They arise when nodes, known as spokes, are required to send and receive mail, passengers, data, or some other commodity via transshipment points known as hubs. Hubs are responsible for re-routing the flow to the destination point which may involve forwarding it to another hub. All nodes are connected to a hub and all hubs are connected to each other. This allows all flow to move from an origin to a destination node by passing through at least one hub and at most two.

Spokes send and receive flow to and from hubs using cheap low-speed low-volume links while hubs forward flows between hubs using high-speed high-volume links which provide cheaper, more efficient, transportation cost at a higher link-establishment cost. If hubs are located well and spokes are allocated effectively so that cheap inter-hub flow volume is kept high, and expensive spoke to hub or hub to spoke flow is kept low the resulting cost of internodal flow is low while the speed and efficiency of transfer is high. In general the purpose of hub location problems is to construct such low cost, high efficiency networks and these types of networks can be seen in use in many areas.

Some common examples of hub-spoke networks include passenger airlines [1]–[3], express package delivery firms [4], message-delivery networks [5], trucking routes [6], telecommunication systems [7], supply-chains for chain stores [8], and many other similar applications. Well-constructed hub-spoke networks can often improve the performance of distribution systems at a lower cost than other methods and so research into effective design of hub-spoke networks and the HLPs have received much attention in the literature.

While the HLP has many variants, we focus on the Single Allocation Hub Location Problem (SAHLP) which is a NP-Hard [9] combinatorial optimization problem. In the SAHLP spokes are allocated to exactly one hub and the number and location of the hubs is not known *a priori* making the SAHLP one of the more difficult variants of the problem. The SAHLP problem has three facets where the first is the optimal choice of the number of hubs, the second is the optimal location of those hubs, and the third is the optimal allocation of spokes to hubs. Varying amounts of focus have been put on each of these facets by exact and heuristic methods to solve the SAHLP in the literature. Some of these methods include a quadratic integer programming formulation [10], a linear programming formulation [9], Genetic Algorithms (GA) [11], [12] as well as hybrid methods combining Genetic Algorithms and Tabu Search (GATS) [13], and Simulated Annealing (SA) and Tabu Search (TS) [14]. For the capacitated version of the problem a branch and bound technique [15], and Ant Colony Optimization (ACO) have been proposed [16]. The Uncapacitated Single Allocation Hub Location Problem (USAHLP) is a SAHLP in which hubs can process an unlimited amount of flow.

Although the USAHLP has received various attention from the meta-heuristics community, and PSO has been applied to other variants of HLP, to the authors' knowledge, no PSO implementation exists for the USAHLP investigated here. This paper aims to bridge this gap by proposing a Discrete Particle Swarm Optimizer (DPSO) approach for the USAHLP. An empirical study comparing the DPSO with published work using well-known benchmark problems from the Civil Aeronautics Board and Australian Post data sets with node sizes of up to 200 shows the effectiveness of the DPSO.

The remainder of this paper is as follows. Sections II and III give a brief overview of the USAHLP and the basic PSO algorithm respectively. Section IV explains the proposed DPSO, while Sections V provides the experimental results. The concluding remarks and future work is finally given in Section VI.

## II. Uncapacitated Single Allocation Hub Location Problem, USAHLP

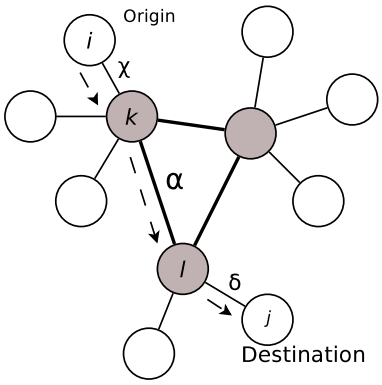


Fig. 1. Transportation Cost Within a Hub-Spoke Network

The USAHLP is a hub location problem in which spokes are assigned to a single hub. These hubs have no associated capacity limit with respect to the amount of flow they are able to process, although, there is a fixed cost for establishing them. The goal in the SAHLP is to minimize both the hub establishment costs as well as the transportation costs without violating some important constraints – spokes may only be connected to hubs, all hubs must be connected to each other, and flow originates and terminates at a spoke. Given this topology, there are three distinct steps to transporting flow from origin to destination and there are three associated costs. (1) The *collection cost*, $\chi$, the cost associated with flow from an origin node to its hub, (2) the *transfer cost*, $\alpha$, the cost associated with flow between hubs, and (3) the *distribution cost*, $\delta$, is the cost associated with flow from a hub to a destination node.

All costs are considered per unit distance of flow volume between nodes. For example, if as in Fig. 1 $W_{ij}$ units of flow are sent from node $i$ to node $j$ via hubs $k$ and $l$. The $W_{ij}$ units of flow are sent from node $i$ to hub $k$ then from hub $k$ to hub $l$, and finally to the destination node $j$. The transportation cost $C_{ijkl}$ is then said to be,

$$C_{ijkl} = W_{ij}(\chi d_{ik} + \alpha d_{kl} + \delta d_{lj}) \tag{1}$$

where $d_{ik}$ is the distance between spoke $i$ and hub $k$, $d_{kl}$ is the distance between hub $k$ and hub $l$, and $d_{lj}$ is the distance between hub $l$ and spoke $j$. The cost of the entire network is the sum of $C_{ijkl}$ for all node pairs, plus the cost of establishing any hubs. The cost of an optimal network and formal definition of the USAHLP using a Quadratic Integer Programming formulation is given by O'Kelly in [10] and shown below. Constraint $(a)$ ensures that all the traffic between an origin-destination pair has been routed via the hub sub-network. Constraint $(b)$ prevents non-hub nodes from being allocated to other non-hub nodes while Constraint $(c)$ restricts the commodity flow through each hub. For some hub-spoke networks the flow may not be symmetric i.e., $W_{ij} \neq W_{ji}$. Additionally, a node may route flow to itself i.e., $W_{ii} > 0$. Both symmetric and non-symmetric flows are considered in this work.

$$Minimize \quad \sum_{i\in N_N}\sum_{k\in N_N}\sum_{l\in N_N}\sum_{j\in N_N} W_{ij}(\chi d_{ik} + \alpha d_{kl} + \delta d_{lj})X_{ijkl} + \sum_{k\in N} F_k Z_{kk}$$

Subject to:

$$\sum_{k\in N_N}\sum_{l\in N_N} X_{ijkl} = 1, \qquad \forall i,j \in N_N, \qquad (a)$$

$$Z_{ik} \leq Z_{kk}, \qquad \forall i,k \in N_N, \qquad (b)$$

$$\sum_{j\in N_N}\sum_{l\in N_N}(W_{ij}X_{ijkl} + W_{ji}X_{jilk}) = (O_i + D_i)Z_{ik} \qquad \forall i,k \in N_N, \qquad (c)$$

$$Z_{ik} \in \{0,1\}, \qquad \forall i,k \in N_N, \qquad (d)$$

$$0 \leq X_{ijkl} \leq 1, \qquad \forall i,j,k,l \in N_N, \qquad (e)$$

Where:

$O_i = \sum_{j\in N_N} W_{ij}$
$D_i = \sum_{j\in N_N} W_{ji}$
$N$ is the number of nodes.
$N_N = \{0, 1, 2, ..., N-1\}$
$W_{ij}$ is the flow between the origin $i$ and destination $j$.
$\chi$ is the *collection* cost (from origin spoke to hub).
$\alpha$ is the *transfer* cost (between hubs).
$\delta$ is the *distribution* cost (from hub to destination spoke).
$d_{ik}$ represents the distance between nodes $i$ and hub $k$.
$d_{kl}$ is the distance between hubs $k$ and $l$.
$d_{lj}$ is the distance between hub $l$ and node $j$.
$X_{ijkl}$ is the decision variable that represents the fraction of traffic between origin node $i$ to destination node $j$ through hubs $k$ and $l$.
$F_i$ is the cost of establishing node $i$ as hub.
$Z_{ij}$ is 1 if node $i$ is assigned to hub $j$, otherwise it is 0.
$Z_{kk}$ is 1 if node $k$ is also a hub, otherwise it is 0.

## III. Particle Swarm Optimization

PSO algorithms are bio-inspired meta-heuristics that are inspired by the swarming behavior in nature such as fish schooling or flocking of birds [17], [18]. A PSO swarm consists of a pool of particles where a particle position in $n$-space represents a solution to a given problem. As each particle moves around in $n$-space it remembers its best position so far, and it is also aware of the global best position found by all particles. At each iteration each particle will move stochastically toward its own best-known position, i.e., the *cognitive* aspect of PSO, or the globally best-known position, the *social* aspect.

The magnitude and direction of this movement is given by a velocity vector, which is updated on each iteration to reflect the desired search direction. Subsequently the velocity vector is used to update the position of the particle, which is the guiding search mechanism in PSO. The velocity and position update rules are given in (2) and (3) respectively [19]:

$$\vec{v}' = \omega\vec{v} + c_1 r_1(\vec{p}_{\text{lbest}} - \vec{p}_{\text{present}}) + c_2 r_2(\vec{p}_{\text{gbest}} - \vec{p}_{\text{present}}) \quad (2)$$

$$\vec{p}'_{present} = \vec{p}_{\text{present}} + \vec{v}' \quad (3)$$

where $\omega$ is the inertia factor, $\vec{v}$ is the velocity vector, $\vec{v}'$ is the updated velocity vector, $\vec{p}_{present}$ is the position vector, $\vec{p}'_{present}$ is the updated position vector, $\vec{p}_{lbest}$ is the particle's current best-position, $\vec{p}_{gbest}$ is the globally-known best position so far. $r_1, r_2$ are random numbers belonging to the range $[0, 1]$ and $c_1, c_2$ specify the importance of the social moves versus cognitive moves. Algorithm 1 gives the pseudocode for a basic PSO. Particle velocity is bounded to a maximum velocity $v_{max}$, if the sum of accelerations would cause the velocity in a dimension to exceed $v_{max}$ then the velocity is left at $v_{max}$. Basic PSO is suited very well to solving optimization problems in continuous space – the standard update rule uses vectors in continuous space only. For discrete-space problems the framework of the algorithm can remain, but a new update rule may be required.

**Algorithm 1** Simple PSO

```
Initialize all particles;
while termination criteria is not met do
  for each particle do
    Calculate fitness value;
    if Fitness value is < particle's best-known value then
      Set current value as particle's best-known value;
    end if
  end for
  Choose the particle with the best fitness value of all the
  particles as the global best position;
  for each particle do
    Calculate particle velocity according to (2);
    Update particle position according to (3);
  end for
end while
```

## IV. Discrete Particle Swarm Optimization for USAHLP

### A. Particle Representation

We employed a particle representation scheme termed as the *List-Based* encoding from previous work [12]. This encoding presents a network as a list of hubs with $N$ entries giving the total number of nodes in the network. Each node in the network is assigned a unique label from 1 to $N$. The $i^{th}$ entry in the encoded list is the node label, $j$. This means the spoke node identified by label $i$ is assigned to the hub identified by the label $j$. A hub is considered to be assigned to itself, so in this case $i = j$. The representation is illustrated by Fig. 2.

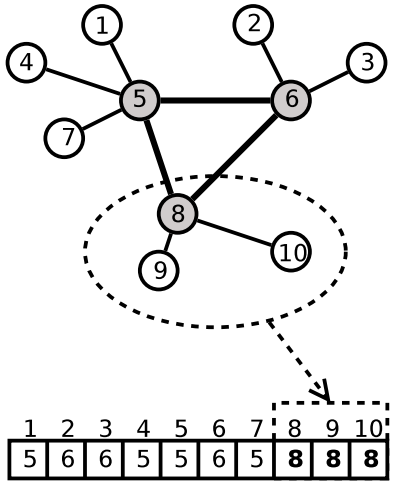


Fig. 2. A List-Based Network Encoding

Particles are initialized by first generating a random number $m$ to be the number of hubs in the network, $2 \leq m \leq N/2$. Second, $m$ random nodes are assigned as hubs. Finally, spokes are allocated to the hubs using a distance assignment rule which allocates them to the nearest hub in terms of Euclidean distance.

### B. Particle Evaluation

Particles are evaluated based on the transportation and hub-establishment cost of the network the particle represents. Before being evaluated particles must be translated into valid networks. Because the PSO may not adhere to the convention that hubs are assigned to themselves it does a scan from left to right of the particle position and it assigns hubs to themselves as it encounters them in the list. For example, if a 5 is encountered in the first position of the list then a value of 5 is inserted at the fifth position. This ensures valid network representations for evaluation. The transportation and hub-establishment costs are given as a weighted-sum objective function $f(x)$, which is to be minimized, and is defined as:

$$\begin{aligned} f(x) = \quad & \Sigma_{i\in N_N}\Sigma_{k\in N_N} \\ & \Sigma_{l\in N_N}\Sigma_{j\in N_N} \\ & \times W_{ij}(\chi d_{ik} + \alpha d_{kl} + \delta d_{lj})X_{ijkl} \\ & + \Sigma_{k\in N_N} F_k Z_{kk}. \end{aligned} \quad (4)$$

Where the first term represents the transportation cost of internodal flow, and the second term represents the cost of establishing hubs.

### C. Particle Update Rule

Traditionally PSO was proposed to solve problems in continuous space, and thus presents extra challenges when it comes to discrete optimization. In this work, in order to function in discrete space, the basic PSO particle update rules are modified. In the spirit of the standard PSO, the proposed DPSO takes the magnitude of the change during particle update to be the difference between a particle's current position and the best-known position. However, in the DPSO

the distance metric becomes the difference in values returned by the particle evaluation function. This effectively organizes particles on a one dimensional ordinal scale where functionally similar networks are grouped together. The exact difference in the change is calculated to be the ratio between the value of the current position and the value of either the globally best-known position or the particle's own best-known position. The probability of moving toward the particle's own best-known position is given in (5), while the probability of moving toward the global best position is given in (6). The value for the magnitude of the move is given by (7). These equations are defined as follows:

$$p_l = \frac{(c_1 r_1)}{(c_1 r_1 + c_2 r_2)} \tag{5}$$

$$p_g = \frac{(c_2 r_2)}{(c_1 r_1 + c_2 r_2)} = 1 - p_l \tag{6}$$

$$Magnitude = \frac{f(\vec{p}_{currentparticle})}{f(\vec{p}_{bestparticle})} \tag{7}$$

where $p_l$ is the probability of a particle moving towards its own best-known position, a cognitive move. $p_g$ is the probability of a particle moving towards the globally best-known position, a social move, and $Magnitude$ is the magnitude of the movement. $c_1, c_2$ are constant learning factors, and $r_1, r_2$ are randomly generated numbers in the range $[0, 1]$. Once the decision to move toward the global known-best or the particle's own known-best is made based on the probabilities $p_1$ and $p_2$, a random dimension in the particle's position vector is chosen as the start point $s$, and $n$ consecutive dimensions of $N$ total dimensions are changed to match the best-known solution where,

$$n = Magnitude * N. \tag{8}$$

If copying $n$ consecutive positions starting from $s$ brings the procedure past the end of the particle's position vector, the procedure starts copying from the beginning of the vector until $n$ positions have been changed.

In this work, we interpretated the velocity update rule in a different way than the original PSO. This was motivated by the idea that velocity as it is used in continuous space makes less sense in discrete space, and many other discrete particle swarm optimizers take similar approaches [20]–[23]. In continuous space the velocity defines the magnitude and direction of the change in particle position. In our discrete space model, (7) and (8) replace the velocity update equation (4) in the vanilla PSO algorithm. The social and cognitive components are taken care of by equations (5) and (6) and equations (5) to (8) describe the mechanics of the entire particle update rule.

### D. Exploitative Processes

The update procedure discussed so far encompass one of the core functions of the velocity vector, but with the current scheme particles have no mechanism to fully explore nearby regions of the search space once they have arrived at a best-position. To compensate for this a function, called *jitter*, was used to move particles to nearby positions in the search space. The jitter function probabilistically applied the *Shift node*, *Swap node*, and *Change hub* node operations employed in earlier work [12]. The operations are illustrated in Fig. 3. Using the jitter allows for greater exploitation and keeps particles moving.

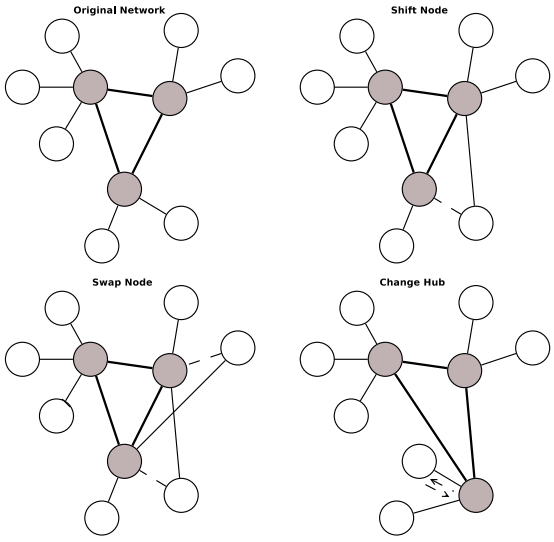


Fig. 3. Shift Node, Swap Node, and Change Hub Functions

## V. Experimentation and Results

The proposed DPSO algorithm was implemented using Java 1.5 and all experiments were run on a PC with an Intel Core 2 CPU T6400 running at 2.00GHz, with 2GB of RAM in a Linux environment. We now provide the details of data sets used and discuss experimental results.

### A. Data Sets

We used two well-known benchmark data sets, i.e., Civil Aviation Board (CAB) data set [24] and the Australian Post (AP) data set [15]. The CAB data set with problem instances of up to 25 nodes is based on air traffic between 25 cities in the U.S.A. and has been used extensively for hub location problems. Unlike the AP data with asymmetric flows between nodes, the internodal flow ($W_{ij}$) in CAB data set is symmetric, i.e., $W_{ij} = W_{ji}$ and is scaled by division with the total network flow, i.e.,

$$\sum_{i=1,j=1}^{n,n} W_{ij} \tag{9}$$

The collection, $\chi$, and distribution costs, $\delta$, are both set to be 1.0 [11], [13], [14]. The transfer cost, $\alpha$, is varied between 0.2 and 1.0 [11], [13], [14].

The AP data set was introduced by Ernst *et al.* [11] and is based on a real application to postal delivery system in Australia. AP data is the only data benchmark data set available for capacitated hub location problems. The AP data set was intended to be used for capacitated problems, it is usually adapted for the USAHLP by ignoring the capacity constraints [25] [12]. The AP data set problems have asymmetric flows

($W_{ij} \neq W_{ji}$) and the unit collection, transfer, and distribution costs are $\chi = 3.0$, $\alpha = 0.75$, and $\delta = 0.2$ respectively.

Problems in the AP data set are divided into two different problem types for the SAHLP – loose cost (L) and tight cost (T). Loose cost problems tend to have less variation in hub-establishment costs while tight cost problems have wider variation. The problem type and size is denoted by $nF$ where $n$ is the problem-size and $F$ is the problem type (either L or T). For example "10L" refers to a loose cost problem with 10 nodes.

### B. DPSO Parameter Setting

Table I describes empirically established parameter settings used for the DPSO. It should be noted that the jitter function listed below operates by using the probabilities of 0.2, 0.6, and 0.2 to determine if the *Shift Node*, *Swap Node*, or *Create Hub* functions would be applied respectively. All experiments were based on 50 runs of each DPSO per problem and execution was terminated after 500 iterations. The results provided here are for the best particle in the 50 runs. However averages were omitted due to space limitations, but they are readily available. Learning factors were found to impact the convergence of the DPSO methods very little but the values of 2 for each factor showed the best results in experimentation. Jitter values were set empirically, but based on previous work [12].

TABLE I
DPSO PARAMETER SETTINGS

| Parameter | Value |
|---|---|
| Swarm Size | 200 |
| Iterations | 500 |
| Social Learning Factor | 2 |
| Cognitive Learning Factor | 2 |
| Probability of Jitter | 0.4 |

### C. Experimental Results and Discussions

The DPSO was run on the AP and CAB data sets, and the solution quality compared with published work. First, before designing the DPSO, for comparison, a naive adaptation of the vanilla PSO algorithm for discrete space was implemented. This adaptation allows particles to move about in continuous space bounded to the range $[0, N]$ in each dimension where $N$ is the number of nodes in the problem. In order to translate a particle's position into a network, the decimal portion of the values in each dimension are truncated and the resulting vector is then interpreted as a list-based network representation. If a node appears as a hub in some position in the list but is not assigned to itself, this is corrected by assigning it properly to itself. Priority for correction is given in a left-right order. For example the vector $[1.14, 1.76, 4.98, 3.23]$ is translated to the vector $[1, 1, 4, 3]$, correcting the hub assignment problem it becomes, $[1, 1, 4, 4]$. The final network has the spokes 2 and 3 assigned to hubs 1 and 4 respectively.

Table II shows the performance of vanilla PSO, run with the parameters $c_1 = c_2 = 2$, compared with DPSO in terms of solution quality for the AP data. A tick means that the known best solution has been found. The values in brackets next to a tick indicate how many times in 50 runs a given algorithm found the known best solutions. Table II clearly shows that the DPSO outperforms the vanilla PSO for the hub location problem studied here. The same performance trend when comparing vanilla PSO with DPSO was observed for the CAB data and those results are available but not included here. However, in terms of computational time, while both algorithms are relatively fast, DPSO takes almost twice as long. An example of the time tradeoff is shown where for the 100LL problem, the average time for one vanilla PSO run over 50 runs is 4.7s, while the average time for one DPSO run over 50 runs is 8.759s. An example of convergency rates for vanilla PSO and DPSO is shown in Fig. 4 when considering the average of best fitness for the 100LL problem. An unpaired t-test was done and p-values are given in Table II. The p-values are all much lower than 0.05 and we can reject the null hypothesis that the difference in means is zero for each problem size at a 95% confidence level.

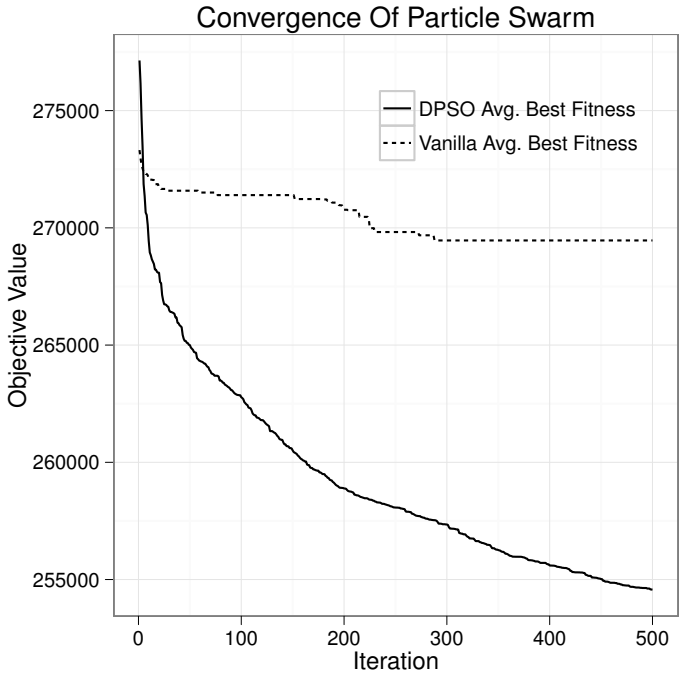


Fig. 4. Average Best Fitness Convergence of Particle Swarms

Further comparisons were done where the performance of the proposed DPSO was compared with other well-known meta-heuristics. Table III shows the comparison for AP data, while Tables IV, V, and VI compares results for CAB data, i.e., comparing the DPSO results to GA results [11], GATS [13], SATLUHLP [14], MSTS-3 [25], and HubTS [25]. A Table with problem size of 10 nodes is omitted but shows similar performance as shown here. A tick mark in the tables means that a best-known solution has been found, and the number in the brackets shows how many times in 50 runs the DPSO found the known best solution. In terms of computational time,

DPSO was only compared with the GA found in [12] which provided known best costs. The computational time for the DPSO and GA was found to be comparable. For example for the 100 LL problem, the average time for one DPSO run over 50 runs is 8.759s while that of GA was 8.836s. This shows that although the DPSO was shown to be slower than vanilla PSO, when compared to other best performing meta-heuristics such as GAs for the USAHLP, the DPSO's computational time is competitive.

As depicted in the various tables of results, the DPSO matches or beats performance of the most effective methods in the literature in terms of solution quality. We can conclude that a relatively simple PSO is equally effective in finding the best-known solutions by other meta-heuristics such as genetic algorithms, tabu search and hybrid approaches for the USAHLP.

TABLE II
COMPARISON TO VANILLA PSO WITH TRUNCATION, AP DATA. A $\surd$ INDICATES THE OPTIMUM WAS FOUND. THE NUMBERS IN BRACKETS INDICATE HOW MANY TIMES THE OPTIMUM WAS FOUND.

| Problem | Known-Best | Vanilla PSO | DPSO | t-test p-value |
|---|---|---|---|---|
| 10L | 224250.05 | $\surd$(27) | $\surd$ (50) | $1.501620 \cdot 10^{-4}$ |
| 20L | 234690.95 | $\surd$(6) | $\surd$ (38) | $1.459319 \cdot 10^{-11}$ |
| 25L | 236650.62 | 237290.38 | $\surd$ (36) | $1.059286 \cdot 10^{-14}$ |
| 40L | 240986.23 | 241260.24 | $\surd$ (5) | $1.342519 \cdot 10^{-6}$ |
| 50L | 237421.98 | 238847.35 | $\surd$ (11) | $3.118376 \cdot 10^{-6}$ |
| 100L | 238016.28 | 241609.21 | $\surd$ (2) | $2.382180 \cdot 10^{-8}$ |
| 200L | 233802.97 | 253398.92 | $\surd$ (1) | $1.780552 \cdot 10^{-14}$ |
| 10T | 263399.94 | $\surd$(13) | $\surd$ (50) | $1.451513 \cdot 10^{-7}$ |
| 20T | 271128.18 | 272185.00 | $\surd$ (37) | $3.748976 \cdot 10^{-8}$ |
| 25T | 295667.84 | $\surd$(2) | $\surd$ (32) | $2.926856 \cdot 10^{-12}$ |
| 40T | 293164.83 | 298919.01 | $\surd$ (7) | $1.501760 \cdot 10^{-11}$ |
| 50T | 300420.98 | 303867.95 | $\surd$ (5) | $4.917037 \cdot 10^{-3}$ |
| 100T | 305097.96 | 398414.63 | $\surd$ (2) | $3.236456 \cdot 10^{-23}$ |
| 200T | 272188.11 | 296844.38 | $\surd$ (1) | $4.686968 \cdot 10^{-17}$ |

## VI. CONCLUSIONS

While PSO has been applied to various optimization problems including similar discrete optimization problems to the uncapacitated single allocation hub location problem, such as the vehicle routing problem with time windows [20], minimum labeling steiner tree problem [22] and the p-median problem [23] with varying degrees of success, to the authors' knowledge, no PSO approach has been used for the single allocation hub location problem. This paper provides the first attempt at solving the USAHLP using PSO by proposing a simple but effective discrete particle swarm optimization (DPSO) approach for the USAHLP, which was inspired by the jumping particle swarm optimizer by Castro *et al.* [22]. Experimental evaluation using benchmark data showed the effectiveness of the proposed discrete PSO for USAHLP. The DPSO was able to find results equal to the best-known results for all problems in the benchmark data sets. Our next step is to further evaluate the DPSO for larger problems, and extend it to the capacitated version of the SAHLP.

TABLE IV
COMPARISON WITH OTHER METHODS, $n = 25$, CAB DATA. A $\surd$ INDICATES THE OPTIMUM WAS FOUND. THE NUMBERS IN BRACKETS INDICATE HOW MANY TIMES THE OPTIMUM WAS FOUND.

| $\alpha$ | $f$ | Optimal Cost | Other Methods [13] [14] [25] [11] | DPSO |
|---|---|---|---|---|
| 0.2 | 100 | 1029.63 | $\surd$ | $\surd$ (6) |
| | 150 | 1217.34 | $\surd$ | $\surd$ (12) |
| | 200 | 1367.34 | $\surd$ | $\surd$ (10) |
| | 250 | 1500.90 | $\surd$ | $\surd$ (33) |
| 0.4 | 100 | 1187.51 | $\surd$ | $\surd$ (5) |
| | 150 | 1351.69 | $\surd$ | $\surd$ (6) |
| | 200 | 1501.62 | $\surd$ | $\surd$ (20) |
| | 250 | 1601.62 | $\surd$ | $\surd$ (34) |
| 0.6 | 100 | 1333.56 | $\surd$ | $\surd$ (12) |
| | 150 | 1483.56 | $\surd$ | $\surd$ (5) |
| | 200 | 1601.20 | $\surd$ | $\surd$ (28) |
| | 250 | 1701.20 | $\surd$ | $\surd$ (32) |
| 0.8 | 100 | 1458.83 | $\surd$ | $\surd$ (8) |
| | 150 | 1594.08 | $\surd$ | $\surd$ (12) |
| | 200 | 1690.57 | $\surd$ | $\surd$ (10) |
| | 250 | 1740.57 | $\surd$ | $\surd$ (21) |
| 1.0 | 100 | 1556.63 | $\surd^{++}$ | $\surd$ (1) |
| | 150 | 1640.57 | $\surd$ | $\surd$ (5) |
| | 200 | 1690.57 | $\surd$ | $\surd$ (11) |
| | 250 | 1740.57 | $\surd$ | $\surd$ (6) |

$^{++}$ GA and GATS values for the $\alpha = 1.0$ and $f = 100$ case are 1559.19 and 1562.16 respectively.

TABLE V
COMPARISON WITH OTHER METHODS, $n = 20$, CAB DATA. A $\surd$ INDICATES THE OPTIMUM WAS FOUND. THE NUMBERS IN BRACKETS INDICATE HOW MANY TIMES THE OPTIMUM WAS FOUND.

| $\alpha$ | $f$ | Optimal Cost | Other Methods [13] [14] [25] [11] | DPSO |
|---|---|---|---|---|
| 0.2 | 100 | 967.74 | $\surd$ | $\surd$ (16) |
| | 150 | 1174.53 | $\surd$ | $\surd$ (17) |
| | 200 | 1324.53 | $\surd$ | $\surd$ (23) |
| | 250 | 1474.53 | $\surd$ | $\surd$ (13) |
| 0.4 | 100 | 1127.09 | $\surd$ | $\surd$ (9) |
| | 150 | 1297.76 | $\surd$ | $\surd$ (15) |
| | 200 | 1442.56 | $\surd$ | $\surd$ (18) |
| | 250 | 1570.91 | $\surd$ | $\surd$ (27) |
| 0.6 | 100 | 1269.15 | $\surd$ | $\surd$ (3) |
| | 150 | 1406.04 | $\surd$ | $\surd$ (17) |
| | 200 | 1506.04 | $\surd$ | $\surd$ (24) |
| | 250 | 1701.20 | $\surd$ | $\surd$ (7) |
| 0.8 | 100 | 1369.52 | $\surd$ | $\surd$ (9) |
| | 150 | 1469.52 | $\surd$ | $\surd$ (15) |
| | 200 | 1520.91 | $\surd$ | $\surd$ (14) |
| | 250 | 1570.91 | $\surd$ | $\surd$ (10) |
| 1.0 | 100 | 1410.07 | $\surd$ | $\surd$ (15) |
| | 150 | 1470.91 | $\surd$ | $\surd$ (7) |
| | 200 | 1520.91 | $\surd$ | $\surd$ (5) |
| | 250 | 1570.91 | $\surd$ | $\surd$ (11) |

## REFERENCES


[1] T. Aykin, "Networking policies for hub-and-spoke systems with application to the air transportation system," *Transportation Science*, vol. 29, no. 3, pp. 201 – 221, 8 1995.

[2] N. Bania, P. Bauer, and T. J. Zlatoper, "U.s. air passenger service: a taxonomy of route networks, hub locations, and competition," Federal Reserve Bank of Cleveland, Working Paper 9216, 1992.

[3] D. Skorin-Kapov and J. Skorin-Kapov, "On tabu search for the location of interacting hub facilities," *Euro. J. of Oper. Res.*, vol. 73, no. 3, pp. 502–509, March 1994.

[4] M. J. Kuby and R. G. Gray, "The hub network design problem with stopovers and feeders: The case of federal express," *Transportation Research Part A: Policy and Practice*, vol. 27, no. 1, pp. 1–12, January 1993.

TABLE III

COMPARISON WITH OTHER METHODS, AP DATA. A $\surd$ INDICATES THE OPTIMUM WAS FOUND. THE NUMBERS IN BRACKETS INDICATE HOW MANY TIMES THE OPTIMUM WAS FOUND.

| Problem | Known-Best | GA-MCEC [12] | MSTS-3 [25] | HubTS [25] | DPSO |
|---|---|---|---|---|---|
| 10L | 224250.05 | $\surd$ | $\surd$ | $\surd$ | $\surd$ (50) |
| 20L | 234690.95 | $\surd$ | $\surd$ | $\surd$ | $\surd$ (38) |
| 25L | 236650.62 | $\surd$ | $\surd$ | $\surd$ | $\surd$ (36) |
| 40L | 240986.23 | $\surd$ | $\surd$ | $\surd$ | $\surd$ (5) |
| 50L | 237421.98 | $\surd$ | $\surd$ | $\surd$ | $\surd$ (11) |
| 100L | 238016.28 | $\surd$ | $\surd$ | $\surd$ | $\surd$ (2) |
| 200L | 233802.97 | $\surd$ | 233803.02 | 233803.02 | $\surd$ (1) |
| 10T | 263399.94 | $\surd$ | $\surd$ | $\surd$ | $\surd$ (50) |
| 20T | 271128.18 | $\surd$ | $\surd$ | $\surd$ | $\surd$ (37) |
| 25T | 295667.84 | $\surd$ | $\surd$ | $\surd$ | $\surd$ (32) |
| 40T | 293164.83 | $\surd$ | $\surd$ | $\surd$ | $\surd$ (7) |
| 50T | 300420.98 | $\surd$ | $\surd$ | $\surd$ | $\surd$ (5) |
| 100T | 305097.96 | $\surd$ | $\surd$ | $\surd$ | $\surd$ (2) |
| 200T | 272188.11 | $\surd$ | 27237.78 | 27237.78 | $\surd$ (1) |

TABLE VI

COMPARISON WITH OTHER METHODS, $n = 15$ CAB DATA. A $\surd$ INDICATES THE OPTIMUM WAS FOUND. THE NUMBERS IN BRACKETS INDICATE HOW MANY TIMES THE OPTIMUM WAS FOUND.

| $\alpha$ | $f$ | Optimal Cost | Other Methods [13] [14] [25] [11] | DPSO |
|---|---|---|---|---|
| 0.2 | 100 | 1030.07 | $\surd$ | $\surd$ (29) |
| | 150 | 1239.77 | $\surd$ | $\surd$ (44) |
| | 200 | 1381.28 | $\surd$ | $\surd$ (41) |
| | 250 | 1481.28 | $\surd$ | $\surd$ (43) |
| 0.4 | 100 | 1179.71 | $\surd$ | $\surd$ (28) |
| | 150 | 1355.09 | $\surd$ | $\surd$ (49) |
| | 200 | 1462.62 | $\surd$ | $\surd$ (46) |
| | 250 | 1556.66 | $\surd$ | $\surd$ (49) |
| 0.6 | 100 | 1309.92 | $\surd$ | $\surd$ (27) |
| | 150 | 1443.97 | $\surd$ | $\surd$ (31) |
| | 200 | 1506.66 | $\surd$ | $\surd$ (34) |
| | 250 | 1556.66 | $\surd$ | $\surd$ (37) |
| 0.8 | 100 | 1390.76 | $\surd$ | $\surd$ (25) |
| | 150 | 1456.66 | $\surd$ | $\surd$ (14) |
| | 200 | 1506.66 | $\surd$ | $\surd$ (30) |
| | 250 | 1556.66 | $\surd$ | $\surd$ (34) |
| 1.0 | 100 | 1406.66 | $\surd$ | $\surd$ (30) |
| | 150 | 1456.66 | $\surd$ | $\surd$ (30) |
| | 200 | 1506.66 | $\surd$ | $\surd$ (45) |
| | 250 | 1556.66 | $\surd$ | $\surd$ (40) |


[5] J. G. Klincewicz, "Hub location in backbone/tributary network design: a review," *Location Science*, vol. 6, no. 1, pp. 307–335, May 1998.

[6] D. T, S. Harit, J. English, and G. Whisker, "Hub and spoke networks in truckload trucking: Configuration, testing, and operational concerns," *Logistics and Transportation*, vol. 31, pp. 209–237, 1995.

[7] Y. Lee, B. H. Lim, and J. S. Park, "A hub location problem in designing digital data service networks: Lagrangian relaxation approach," *Location Science*, vol. 4, no. 3, pp. 185–194, 1996.

[8] S. Abdinnour-Helm, "Using simulated annealing to solve the p-hub median problem," *International Journal of Physical Distribution & Logistics Management*, vol. 31, no. 3, pp. 203 – 220, 2001.

[9] K. B. Amur Sibel, "Network hub location problem: State of the art," *Euro. J. of Oper. Res.*, vol. 190, pp. 447–58, 2001.

[10] M. E. O'kelly, "A quadratic integer program for the location of interacting hub facilities," *Euro. J. of Oper. Res.*, vol. 32, no. 3, pp. 393–404, December 1987.

[11] H. Topcuoglu, F. Corut, M. Ermis, and G. Yilmaz, "Solving the uncapacitated hub location problem using genetic algorithms," *Comput. Oper. Res.*, vol. 32, no. 4, pp. 967–984, 2005.

[12] N. Mohammad and B. M. Ombuki-Berman, "An efficient genetic algorithm for the uncapacitated single allocation hub location problem," in *IEEE CEC 2010: Proceedings of The IEEE World Congress on Computational Intelligence*, 2010, pp. 941–948.

[13] S. Abdinnour-Helm, "A hybrid heuristic for the uncapacitated hub location problem," *Euro. J. of Oper. Res.*, vol. 106, no. 2-3, pp. 489–499, April 1998.

[14] J.-F. Chen, "A hybrid heuristic for the uncapacitated single allocation hub location problem," *Omega*, vol. 35, no. 2, pp. 211 – 220, 2007.

[15] A. Ernst and M. Krishnamoorthy, "Solution algorithms for the capacitated single allocation hub location problem," *Annals of Operations Research*, vol. 86, pp. 141 – 159, 1 1999.

[16] M. Randall, "Solution approaches for the capacitated single allocation hub location problem using ant colony optimisation," *Comput. Optim. Appl.*, vol. 39, no. 2, pp. 239–261, 2008.

[17] R. Eberhart, Y. Shi, and J. Kennedy, *Swarm Intelligence*, ser. The Morgan Kaufmann Series in Evolutionary Computation. Elsevier Science, 2001.

[18] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd ed. Wiley Publishing, 2007.

[19] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. part i: background and development," *Natural Computing*, vol. 6, pp. 467–484, 2007.

[20] J. Castro, D. Landa-Silva, and J. Prez, "Exploring feasible and infeasible regions in the vehicle routing problem with time windows using a multi-objective particle swarm optimization approach," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)*, ser. Studies in Computational Intelligence, N. Krasnogor, M. Melin-Batista, J. Prez, J. Moreno-Vega, and D. Pelta, Eds. Springer, 2009, vol. 236, pp. 103–114.

[21] A. R. Guner and M. Sevkli, "A discrete particle swarm optimization algorithm for uncapacitated facility location problem," *J. Artif. Evol. App.*, vol. 2008, pp. 1–9, 2008.

[22] S. Consoli, J. A. Moreno-Pérez, K. Darby-Dowman, and N. Mladenović, "Discrete particle swarm optimization for the minimum labelling steiner tree problem," *Natural Computing: an international journal*, vol. 9, no. 1, pp. 29–46, 2010.

[23] M.-P. JA, C.-G. JP, M.-G. FJ, M. B, M.-V. JM, and R. J, "Discrete particle swarm optimization for the p-median problem," in *In: Proceedings of the 7th metaheuristics international conference*, 2007.

[24] M. E. O'Kelly, "A quadratic integer program for the location of interacting hub facilities," *Euro. J. of Oper. Res.*, vol. 32, no. 3, pp. 393–404, 1987.

[25] M. R. Silva and C. B. Cunha, "New simple and efficient heuristics for the uncapacitated single allocation hub location problem," *Comput. Oper. Res.*, vol. 36, no. 12, pp. 3152–3165, 2009.