A Modified Discrete Particle Swarm Optimization for Solving Flash Floods Evacuation Operation

[1] Marina Yusoff, [2]Junaidah Ariffin, [1]Azlinah Mohamed

***Abstract*—**Evacuation operation, which is a process of evacuating residents from any dangerous sites to safer destination in the shortest possible time, is of prime importance in emergency management. Untimely assistance and poor coordination at the operation level have always been the major problem in evacuation process during flash floods. This paper focuses on evacuation vehicle routing solution using a modification of a discrete particle swarm optimization (DPSO) with a new search decomposition procedure. Comparative analysis of this algorithm and a genetic algorithm (GA) using the severe flash floods events datasets is performed. The findings indicate that the DPSO provides better performance in both solution quality and processing time. Further experimental analysis for a large evacuation dataset can be considered to confirm the performance of a modified DPSO.

***Keywords*—**Discrete particle swarm optimization, Evacuation operation, Flash floods, Genetic algorithm, Search decomposition.

I. INTRODUCTION

Disasters have made news as catastrophic events had affected people and had incurred losses due to infrastructure damage. It had resulted in loss of life, properties and suffering from serious psychological stress which eventually leads to critical health condition [1]. In December 2006, 15 people had been reported dead and more than 100,000 people had to flee their homes in Johor state [2]. These circumstances have resulted from flash floods. The occurrence of flash flood is due to heavy rainfall that is associated with thunderstorms [3-4]. Table 1 shows the number of people affected of the flash flooded districts in Johor state for December, 2006 and January, 2007**.**

Table 1 Flash flood figures for districts in Johor state [2]

Districts	**Number of people affected**	
	December 2006	**January 2007**
Johor Bharu	20,530	15,229
Kota Tinggi	14,864	15,660
Kluang	19,091	19,210
Muar	34, 253	4,233
Batu Pahat	30, 619	55,259
Pontian	5,978	5,583
Segamat	15,148	8,784
Mersing	1,329	4,517
Total	**111,193**	**128,475**

Heavy rains and overflowing rivers have flooded towns and villages. Malaysia Metrological Department [3] has reported that Kota Tinggi district in Johor faced the most severe flash flood compared to other districts in Johor. This is due to the unusual heavy rainfall and the physiographic of the basin in Kota Tinggi. The seriousness of the situation leads this paper to concentrate on the flash flood evacuation operation in the district of Kota Tinggi. Evacuation datasets of these flash floods event were acquired and computationally experimented. The dataset is shown in Table 2.

Evacuation operation, which is a process of evacuating residents from any dangerous sites to safer destination in the shortest possible time, is of prime importance in emergency management. Untimely assistance and poor coordination at the operation level have always been the major problem in evacuation process during flash floods. A lot of people have to be safely evacuated at the shortest possible time to avoid loss of lives during disaster. Much effort has been done in producing manual evacuation guideline [5-7], developing evacuation system [8], developing simulation [9-10], and developing a wide variety of algorithms [10-12] to facilitate the evacuation operation for different types of disasters.

During the evacuation process, the most challenging task is to move people to safer locations. As time is the decision factor in the evacuation process, urgent and firmly decisions are required. An evacuation plan should be efficiently constructed by taking into consideration the routes of vehicle. This paper addresses the evacuation vehicle routing problem (EVRP) that considers the routing of capacitated vehicles from a vehicle location (single source) to various numbers of potential flooded areas (PFA). This problem is seemed similar to vehicle routing problem (VRP).

EVRP is associated with the routing of vehicles to destinations with limitations of vehicle capacity and depends on the standard travelling speed of each vehicle. Commonly, EVRP relies on the same core problem of the vehicle routing problem (VRP). The VRP which was introduced by Dantzig and Ramser in 1959 [13] is customer based oriented and involves the routing of an unlimited number of vehicles from the depot to customer locations and return to depot with the shortest travelling cost [14-15].

However, EVRP deals with different routing processes, vehicles from vehicle location travelled and picking up people at each PFA and then route them to relief centers. During a pickup, each of vehicles is allocated with a number of people. The allocation is generated from [10]. It has been observed that EVRP is closely related to the capacitated vehicle routing problem (CVRP), primarily in its handling capacity constraints [16-19]. In particular, EVRP deals with routing of a number of vehicles to PFA, whereas CVRP deals with the delivery of goods to customers. Like EVRP, CVRP assumes that each customer is served by exactly one vehicle without exceeding the capacity constraints of each vehicle. Several optimization algorithms have been employed [13][15-17] for solving CVRP. For example, GA with local search is applied in [16] and DPSO with binary position and a hybrid of DPSO-SA in [17]. In general, they obtained an effective result in terms of processing time with no assurance for optimal results.

The recent solution for the EVRP [18] had shown good performance for DPSO compared to GA for a single vehicle location to a single PFA. This paper concentrates on the performance of these algorithms considering the routing the capacitated vehicles from a single vehicle location to multiple PFA with the introduction of a search decomposition procedure in its solution representation.

This paper is organized as follows. Section 2 reviews the PSO algorithm. Section 3 presents the problem formulation. Section 4 presents the EVRP solution and the modified DPSO algorithm. Section 5 explains the computational results and discussion. Finally, Section 6 concludes the paper and addresses some future work.

II. Particle Swarm Optimization

PSO was introduced by Kennedy and Eberhart in the mid-1990s. It is a population-based stochastic approach which has been grouped under swarm intelligence [19-20] and evolutionary computation [22]. PSO can be used to solve continuous and discrete problems. PSO was derived from a concept of a flock of birds which fly everywhere to find food. Each bird is illustrated as a particle. Each particle moves stochastically in search space for a feasible solution. Each of the particles has its own velocity and position. PSO can indicate the velocity and position of particles in a multi-dimensional space. By updating both velocity and position, a feasible solution can be achieved. The fitness values comprise of global (*Gbest)* and personal best (*Pbest)* derived from the simulated behavior of a group of particles [23]. *Pbest* is the solution offered by each of the particle while the *Gbest* is the best solution obtained from all particles.

PSO algorithm has been used to solve continuous problem**.** The algorithm is shown in Algorithm 1 [20]. The algorithm starts with the initialization of the population of particles or swarm size, followed by the initialization of inertia weight (*W*) and acceleration constants (C_1 and C_2). Step 4 and 5 initialize the minimum value ($V_{initialize(min)}$) and maximum value of velocity ($V_{initialize(max)}$) and minimum position (D_{min}) and maximum value of position (D_{max}), respectively. Next is the calculation of *Pbest* and *Gbest* value for each particle. Step 9 calculates the new velocity value for each particle using equation 1. Step 10 updates the new position, $D_{(new)}$ using Equation 2. Finally, *Pbest* $_{(new)}$ and *Gbest* $_{(new)}$ are determined based on the fitness value set for the problem. Iteration starts from step 7 until step 13 to update the current velocity and position of each particle. This iteration will be done until it satisfies the stopping condition.

Algorithm 1 PSO

```
1.  Begin
2.      Initialize number of particles and populations
3.      Declare W, $C_1$ and $C_2$
4.      Initialize $V_{initialize(min)}$ and $V_{initialize(max)}$
5.      Initialize $D_{min}$ and $D_{max}$
6.      Calculate  Pbest and Gbest value for each
        particle
7.      Do
8.        For each particle
9.          Calculate new velocity value, $V_{(new)}$
10.         Calculate new position, $D_{(new)}$
11.         Calculate Pbest $_{(new)}$
12.       Calculate Gbest $_{(new)}$
13.     While (stopping condition is  reached)
14. End
```

PSO has the capability to explore regions of the search space and exploit the search to refine a feasible solution. These search strategies are influenced by the parameters; acceleration constants (C_1 and C_1) and inertia weight [20][24] that has been applied in the PSO algorithm. Equation 1 and 2 present the velocity and position formulas for the canonical PSO, respectively.

$$V_{id(new)} = W * V_{id} + C_1 * r_1 * (Pbest_{(id)} - X_{id} + C_2 * r_2 * (Gbest_{(id)} - X_{id}) \quad (1)$$

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

where:

$V_{id(new)}$	= new velocity of the i^{th} particle in d^{th} dimension
V_{id}	= current velocity of the i^{th} particle in d^{th} dimension
X_{id}	= current position of the i^{th} particle in d^{th} dimension
$X_{id(new)}$	= new position of the i^{th} particle in d^{th} dimension
W	= inertia weight
C_1 and C_2	= acceleration coefficient
r_1 and r_2	= random function in the range of [0,1]
$Pbest_{(id)}$	= position of the personal best of the i^{th} particle in d^{th} dimension
$Gbest_{(id)}$	=position of the global best derived from all particles in the swarm.

A considerable amount of research has been directed towards the modification of canonical PSO to solve several types of continuous problems. The inventors of PSO explored discrete binary PSO with special attention to discrete problems, leading to a new means of updating the position of particles [25] to accommodate discrete binary problems. The particle position is determined based on sigmoid function [25]. The use of DPSO with multiple discrete values (rather than binary) was also explored. Mohemmed et al employed a discrete particle position to represent nodes in the shortest path problem [26].

Izakian et al mapped a grid job scheduling problem to direct and indirect representation for the particle position [27]. Direct representation used binary value while indirect representation used multi discrete value. Direct representation is shown faster than indirect representation. For the direct representation, the particle position is calculated based on the new formulation for the velocity which only considers the *Pbest.*

III. Problem Formulation

The EVRP involves a static routing of a number of vehicles from vehicle location to a single and multiple PFA. EVRP addresses the objective function to find the minimum total travelling time for all capacitated vehicles from vehicle location to the PFA. This problem is mathematically formulated based on the SPP formulation [28]. The problem can be formally defined as follows: Let $G = (N, E)$ be a weighted directed graph. Define $N = \{N_0, N_1, \dots, N_n\}$. N_0 represents the vehicle location and N_n is the destination node (PFA). E is the set of edges. t_{ij} represents the travelling cost of traversing from i to j. For each edge $(i, j) \in E$, travel time $t_{ij} \geq 0$, is a non negative integers. $H = \{H_1, H_2, \dots, H_k\}$ is the set of all vehicles that are able to move from node i and j. The objective function is to find the minimum total travelling time for all vehicles from N_0 to N_n. The EVRP is mathematically formulated as shown below:

$$\text{Minimize} \sum_i^n \sum_j^n \sum_k^m T_{ijk} X_{ij} \tag{3}$$

Subject to:

$$\sum_j^n X_{ij} - \sum_{k=1}^n X_{ki} = \begin{cases} 1 & if\ i = 0 \\ 0 & if\ i = 1, \dots, n-1 \\ -1 & if\ i = n \end{cases} \tag{4}$$

$$X_{ij} \in \{0,1\}, (i,j) \in E \tag{5}$$

where:

i = index of nodes, $i \in N$

j = index of nodes, $j \in N$

k = index of vehicle H, $k \in H$

T_{ijk}= represents the travelling time of vehicle k traversing from i to j.

X_{ij} = is a binary variable which is 1 if the node i to node j *is* traversed, otherwise it is 0.

Constraints 4 ensure that the path starts at N_0, end at N_n, and either pass through or avoid every other node j. Constraint 5 is the set of bound decision variables.

IV. EVRP Solution

The solution of EVRP has adopted the similar process of nodes expansion and the random selection of PV as discussed in [28]. The calculation of travelling time, total travelling time for each of the vehicle travelled through valid path, and the total travelling time for all vehicles that travelled through all the valid paths can be referred to [18]. In EVRP, each array of PV is assigned to each of the vehicles where the number of PV at each level comprises of a number of child nodes (C_n) multiples with the number of vehicles (V_m), or C_n x V_m for each level of the expansion of the graph as illustrated in Fig. 1 and 2.

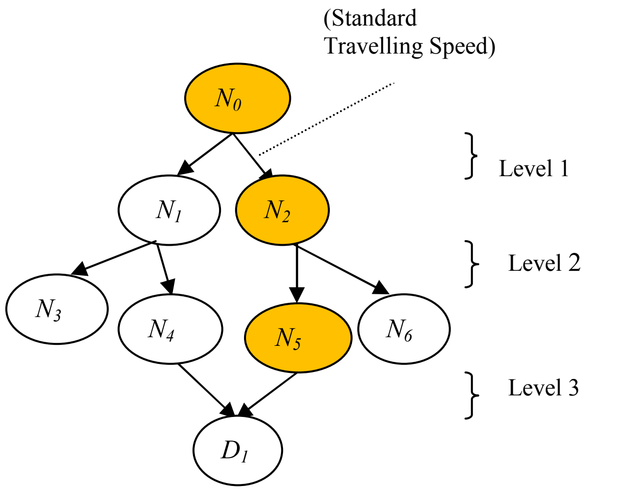

Fig. 1 The illustration of search decomposition procedure into branches

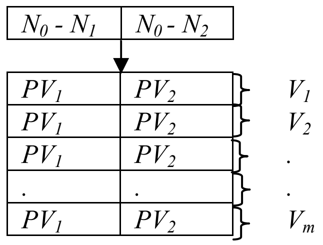

Fig 2. Illustration of a sub-particle for the Level 1

Each particle is represented in 3-dimensions, which consist of number of search decomposition (D_t), C_n, and V_m, or D_t x C_n x V_m. The procedure of the search decomposition is shown in Fig. 3. The decomposition imposes the expansion of nodes from source node. This procedure would enable some limitation on the search space for the movement of particles as illustrated in Fig. 1.

```
1:  Begin
2:   Initialize the search tree
3:   Do
4:     If there is no leaf
5:       Failure
6:     Else
7:       Expand node
8:       Choose one path in a random selection and assigned
         $PV_{min}$ to this path
9:       Calculate traveling time for each vehicle
10:  While (all nodes expanded or there no leaf to expand)
11: End
```

Fig 3. A search decomposition procedure

The new solution representation for EVRP that is discussed above is implemented in DPSO as shown in Algorithm 2. The algorithm starts with the normal process of PSO. Step 2 and 3 initialize the number of population and the coefficient values C_1 and C_2, respectively. Step 4 performs the initialization of PV and velocities. Step 5 retrieves vehicle's information which includes the vehicle id, vehicle capacity, and its standard travelling speed. Step 6 and 7 perform the search decomposition procedure for each of the vehicle. In this step, only one path is selected and the selected node is assigned with PV_{min} upon selection of the path as demonstrated in Fig. 3 After all nodes are expanded, the *Pbest* and *Gbest* of each particle are calculated. *Pbest* is the total distance for each particle, whereas *Gbest* is the minimum total distance obtained from all particles. The iteration process starts at step 9 through 22 until a maximum iteration is achieved. In this iteration, each particle is updated with a new velocity and new position value (PV) at step 10 until 12. The new velocity and position value are in the form of positive integer. Then, PV for all sub particles is updated using step 13. Step 14 performs the decomposition procedure of PV. $Pbest_{(new)}$ and $Gbest_{(new)}$ are calculated at step 15 and 16, respectively. Finally, steps 17 through 21 are the conditions for the selection of the best current fitness for each of the iteration.

Algorithm 2 Modified DPSO

```
1:  Begin
2:   Initialize number of population
3:   Declare $C_1$ and $C_2$
4:   Initialize PV, $V_{intialize(min)}$ and $V_{initialize(max)}$ for all
      particles in random
5:   Retrieve vehicle's information from[10]
6:   For each vehicles
7:     **Perform search decomposition procedure**
8:     Calculate Pbest and Gbest
9:    Do
10:     For each particle
11:       Calculate $V_{(new)}$
12:       Calculate $PV_{(new)}$
13:       Update PV for all sub particles
14:       Perform step 6 and 7
15:       Calculate Pbest $_{(new)}$
16:       Calculate Gbest $_{(new)}$
17:       If (Gbest $_{(new)} >$ Gbest)
18:         Assign $G_{best}$ as the best current fitness
19:      If (Gbest $_{(new)} =<$ Gbest)
20:        Gbest= Gbest $_{(new)}$
21:       Assign $Gbest_{(new)}$ as the best current fitness
22:   While (maximum iteration is achieved)
23: End
```

V. Computational Results and Discussion

The performances of these algorithms are analyzed based on the objective function to find the minimum total travelling time for all the capacitated vehicles from vehicle location to PFA. The comparison involves two aspects: total travelling time for all vehicles from vehicle location to PFA, and the processing time. The selection of parameters was selected based on previous research [13]. DPSO is compared to GA with one point crossover using the same solution representation. Datasets for the computational experiment are from flash flood evacuation in Malaysia starting from VR1_PFAs_06 until VR5_PFAs_07 with various number of destinations (multiple PFA) are shown in Table 2.

Table 2 EVRP datasets

Dataset	Number of PFA	Number of nodes	Number people	Number of vehicles generated from [10]
VR1_PFAs_06	2	47	1416	174
VR2_PFAs_06	3	59	2631	260
VR3_PFAs_06	4	83	3155	370
VR4_PFAs_06	5	119	4584	508
VR5_PFAs_06	6	140	5032	612
VR6_PFAs_06	2	47	7269	750
VR7_PFAs_06	3	59	8484	1316
VR8_PFAs_06	4	83	9008	1366
VR1_PFAs_07	2	49	1566	238
VR2_PFAs_07	3	61	3106	374
VR3_PFAs_07	4	88	3180	355
VR4_PFAs_07	5	109	3800	496
VR5_PFAs_07	6	133	3996	516

A. Performance of a modified DPSO

This section discusses the results for the datasets involved with more than one PFA. The experiments for those datasets used 30 population of particles, 30 experiments and based on the iteration up to 200 or until all vehicles arrived at PFA. The performance of the algorithms is based on the total travelling time (fitness value) obtained by all the travelled vehicles and processing time. Average of the total travelling time and processing time is calculated based on 30 experiments. The results are tabulated in Table 3 until Table 15. Table 3 compares the results of the travelled vehicles from vehicle location to the two PFA. It is apparent that DPSO outperformed the other algorithms at average of 2.699 KM. As can be seen in the table, although, average of processing time is 0.469 seconds for DPSO which is of about 0.010 second more than the average of processing time for GA. This shows that there is very little difference in terms of processing time.

Table 3 Performance of DPSO and GA using dataset VR1_PFAs_06

	DPSO			GA		
	tt_{vs}	PT (s)	iter	tt_{vs}	PT (s)	iter
Avg	2.699	0.469	1	2.702	0.459	1
Min	2.654	0.421	1	2.654	0.421	1
Max	2.750	0.605	1	2.750	0.577	1
Std Dev	0.049	0.047	0	0.049	0.034	0

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

Table 4 shows the results of VR2_PFAs_06. It clearly shows from the table that DPSO gave better result compared to the other three algorithms whereas GA failed to obtain any results for 200 iterations.

Table 4 Performance of DPSO and GA using dataset VR2_PFAs_06

	DPSO			GA		
	tt_{vs}	PT (s)	iter	tt_{vs}	PT (s)	iter
Avg	6.618	1.809	1	6.783	1.946	1
Min	2.654	0.453	1	6.727	1.794	1
Max	6.822	2.153	1	6.913	2.278	1
Std Dev	0.750	0.288	0	0.056	0.096	0

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

As shown in Table 5, on average, DPSO has shown a better fitness value but requires about 0.009 seconds more for the average of processing time when compared to GA. Although GA gives less processing time, the average of total travelling time is slightly higher than DPSO. What is interesting here is the achievement of DPSO in terms of total travelling time. Hence, this algorithm with the new solution representation provides better performance in finding solutions to EVRP focusing on the multiple PFA. This is supported by the results of VR1_PFAs_06 and VR2_PFAs_06, VR4_PFAs_06, VR5_PFAs_06, VR6_PFAs_06, VR7_PFAs_06, and VR8_PFAs_06.

Table 5 Performance of DPSOs and GAs using dataset VR3_PFAs_06

	DPSO			GA		
	tt_{vs}	PT (s)	iter	tt_{vs}	PT (s)	iter
Avg	10.518	14.254	1	10.573	14.147	1
Min	10.313	9.001	1	10.313	9.360	1
Max	11.137	40.498	1	11.137	16.863	1
Std Dev	0.245	5.190	0	0.267	1.247	0

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

The findings support that the embedded search decomposition procedure and random selection of PV brings a significant contribution to the solution of EVRP, in this case for multiple PFA. The random selection assist in obtaining fast convergence because of the limitations of search space required. Overall, DPSO gave a better solution quality (minimum total travelling time) than the other three algorithms for the multiple PFA and competitive to GA for some datasets.

Table 6 Performance of DPSOs and GAs using dataset VR4_PFAs_06

	DPSO			GA		
	tt_{vs}	PT (s)	iter	tt_{vs}	PT (s)	Iter
Avg	17.977	22.141	1	18.174	21.594	1.033
Min	16.405	13.930	1	16.405	13.400	1
Max	20.429	25.973	2	21.185	24.197	2
Std Dev	1.067	1.875	0.183	1.082	2.638	0.305

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

Table 7 Performance of DPSOs and GAs using dataset VR5_PFAs_06

	DPSO			GA		
	tt_{vs}	PT (s)	iter	tt_{vs}	PT (s)	Iter
Avg	23.897	55.378	1	24.230	55.352	1
Min	22.838	45.302	1	22.888	46.582	1
Max	25.611	67.205	1	25.793	99.372	1
Std Dev	0.787	5.927	0	1.028	9.700	0

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

Table 8 Performance of DPSOs and GAs using dataset VR6_PFAs_06

	DPSO			GA		
	tt_{vs}	PT (s)	iter	tt_{vs}	PT (s)	Iter
Avg	11.364	2.478	1	11.377	2.592	1
Min	11.101	1.809	1	11.101	1.856	1
Max	12.641	4.508	1	11.522	4.337	1
Std Dev	0.342	0.803	0	0.204	0.818	0

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

Table 9 Performance of DPSOs and GAs using dataset VR7_PFAs_06

	DPSO			GA		
	tt_{vs}	PT (s)	iter	tt_{vs}	PT (s)	Iter
Avg	24.047	6.083	1	24.110	6.510	1
Min	23.594	5.163	1	23.594	5.741	1
Max	24.305	7.909	1	24.852	6.942	1
Std Dev	0.303	0.426	0	0.311	0.212	0

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

Table 10 Performance of DPSOs and GAs using dataset VR8_PFAs_06

	DPSO			GA		
	tt_vs	***PT (s)***	***iter***	***tt_vs***	***PT (s)***	***Iter***
Avg	25.491	12.588	1	25.696	13.518	1
Min	24.916	11.123	1	24.916	11.700	1
Max	26.728	15.475	1	26.767	25.678	1
Std Dev	0.467	0.682	0	0.483	2.366	0

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

Results for DPSO performed better than the GA as shown in Table 11. The total travelling time for GA is a slightly lower than this algorithm.

Table 11 Performance of DPSOs and GAs using VR1_PFAs_07

	DPSO			GA		
	tt_vs	***PT (s)***	***iter***	***tt_vs***	***PT (s)***	***iter***
Avg	10.170	**3.314**	1	10.173	**3.452**	1
Min	10.167	0.920	1	10.167	1.482	1
Max	10.187	5.132	1	10.193	8.596	1
Std Dev	0.009	1.085	0	0.007	1.565	0

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

The next comparison highlights (Table 12) the results of the VR2_PFAs_07. So far, the proposed DPSO has shown good results with one iteration for convergence (all vehicles arrive at the assigned PFA). Although an average of processing time of DPSO is slightly higher than GA for VR2_PFAs_07, the total travelling time for this algorithm is 0.29% lower than GA, which is about 1.98 minutes. With a minimum total travelling time, all people can be picked-up by the assigned vehicles at each of the PFA at the shortest time.

Table 12 Performance of DPSOs and GAs using VR2_PFAs_07

	DPSO			GA		
	tt_vs	***PT (s)***	***iter***	***tt_vs***	***PT (s)***	***Iter***
Avg	**11.494**	**9.358**	1	11.527	**9.095**	1
Min	11.385	3.978	1	11.385	4.493	1
Max	11.723	22.433	1	13.583	21.419	1
Std Dev	0.090	3.826	0	0.391	4.141	0

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

As shown in Table 13, the result validates the employment of DPSO in solving EVRP. This algorithm provides the best results with less total processing time compared to GA to move all vehicles from vehicle location to four PFAs, using VR3_PFAs_07. This result again ensures all vehicles arrive at PFA at a minimum total travelling time, which is important in evacuation operation.

Table 13 Performance of DPSOs and GAs using VR3_PFAs_07

	DPSO			GA		
	tt_vs	***PT (s)***	***iter***	***tt_vs***	***PT (s)***	***iter***
Avg	**12.170**	**7.594**	2.300	12.174	**17.006**	2.400
Min	11.736	4.025	1.000	11.625	3.931	1.000
Max	13.466	22.479	5.000	13.257	309.620	10.00
Std Dev	0.535	3.836	1.343	0.552	55.332	1.905

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

DPSO outperformed GA for both of VR4_PFAs_07 and VR5_PFAs_07 as shown in Table 14 and 15 in its total travelling time and processing time. It is noted that the use of DPSO has successfully achieved the best performance among the other three algorithms. Thus, these results validate that this algorithm satisfy the objective function which is to find the minimum total travelling time.

Table 14 Performance of DPSOs and GAs using VR4_PFAs_07

	DPSO			GA		
	tt_vs	***PT (s)***	***iter***	***tt_vs***	***PT (s)***	***Iter***
Avg	**17.965**	**19.537**	1.100	18.029	**19.549**	1.259
Min	17.232	10.249	1.000	17.315	9.891	1.000
Max	19.943	31.731	2.000	19.991	60.707	2.000
Std Dev	0.573	5.987	0.305	0.716	9.662	0.447

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

Table 15 Performance of DPSOs and GAs using VR5_PFAs_07

	DPSO			GA		
	tt_vs	***PT (s)***	***iter***	***tt_vs***	***PT (s)***	***Iter***
Avg	20.426	**19.716**	1.5	20.429	**20.467**	1.6
Min	19.511	13.603	1.0	19.647	13.962	1.0
Max	22.616	28.797	5.0	23.027	29.578	4.0
Std Dev	0.693	4.106	0.9	0.900	5.104	0.9

** tt_{vs} – total travelling time (hour), PT - processing time (second), iter - number of iteration*

B. Discussion

This paper produced results which corroborate the solution of [28] for SPP. With some modification to the solution, the findings confirmed that DPSO proved to perform better than GA in getting the minimum total travelling time. Although the processing time obtained was statistically different between these algorithms, on average DPSO consumed less time. The use of multi-valued discrete particle position (PV) with the employment of search decomposition and random selection of

PV was observed to have successfully achieved good performance for a small number datasets.

The suggested mean of limiting the movement of particles in search space, using the search decomposition procedure and random selection support this algorithm. The decomposition of graph with random selection of PV depends on the number of expanded branches. Hence, the number of nodes traversed by each vehicle is dependent on the branch that was randomly selected. With this procedure at least one vehicle can traverse from vehicle location until PFA using a valid path because the selection of PV is limited to the number of branches. Based on the findings provided by DPSO, it can be illustrated that several valid paths were able to be determined using 30 populations of particles granted a higher possibility of using less travelling time for the vehicles travelled from vehicle location to multiple PFA. With the high possibility of getting a valid node, the best solution would become faster and lead to the fast convergence due to the less search space. This confirmed what was mentioned in the literature review that the DPSO has a capability of finding optimal solution and fast convergence compared to the GA.

The calculation of velocity involving exploitation, and exploration of particles in DPSO has contributed to the solution. The exploitation presents means of particles to perform a local search while the exploration is globally seeking the best solution, which was gathered from the selection of *Gbest*.

VI. Conclusion

This paper discusses the solution to evacuation process in achieving the objective function which is to find the minimum total travelling time for all the capacitated vehicles from vehicle location to the multiple PFA. A new solution representation incorporates the search decomposition procedure and random of PV selection were addressed. The new solution is embedded in DPSO. They were compared to GA in which using the same solution representation. Overall, it can be concluded that DPSO that was applied with a new solution representation provided better results compared to GA for multiple PFA. Further experiment can be done using dataset from PFA to relief centers for the EVRP incorporating the limitations of the capacity at the relief center and for large evacuation scenarios.

ACKNOWLEDGEMENT

This study has been made possible under the support of the Ministry of Science and Technology Malaysia through the Science Fund and University Technology MARA.

REFERENCES

[1] M. Omar, Evacuation Planning in Malaysia, Putrajaya, 2007.

[2] Social and Welfare Department. (2010, February 10). Laporan Bencana Banjir Tahun 2006 Hingga 2008. Available:http://www.jkmnj.gov.my/utama/index.php?option=com_content&view=article&id=54%3Alaporan-bencana-banjir-tahun-2006-hingga-2008&catid=1%3Alatest-news&Itemid=64.

[3] Malaysian Meteorological Department. (2011, May10). Laporan Hujan Lebat, Ribut Petir Serta Angin Kencang Yang Mengakibatkan Banjir Di Kelantan, Terengganu, Perlis Dan Kedah Pada 01~02 November 2010. Available:http://www.met.gov.my/

[4] A. Shafie, "A Case Study on Floods of 2006 and 2007 in Johor, Malaysia," Tech. Rep. Colorado State University, 2009.

[5] S. Hoffman, "Preparing for disaster: protecting the most vulnerable in emergencies," *UC Davis L. Rev.,* vol. 42, pp. 1491-1547, 2008.

[6] S. P. Simonovic and S. Ahmad, "Computer-based Model for Flood Evacuation Emergency Planning", *Natural Hazards,* vol. 34, pp. 25-51, 2005.

[7] D. Magiswary, R. Murali, M. Saravanan, and K. Maniam, "ICT and disaster preparedness in Malaysia: an exploratory study," *WSEAS Transactions on Information Science and Applications,* vol. 7, pp. 735-748, 2010.

[8] Y.-C. Chiu, "Traffic scheduling simulation and assignment for area-wide evacuation," in *Proc. 7th International IEEE Conference on Intelligent Transportation Systems*, 2004, pp. 537-542

[9] C. W. Johnson, "Applying the lessons of the attack on the world trade center, 11th September 2001, to the design and use of interactive evacuation simulations," In *Proc. Conference on Human Factors in Computing Systems,* 2005, pp. 651-660.

[10] M. Yusoff, J. Ariffin, and A. Mohamed, "Solving Vehicle Assignment Problem Using Evolutionary Computation," in *Proc. Advances in Swarm Intelligence*. vol 1, 2010, pp. 523-532.

[11] L. Özdamar and W. Yi, "Greedy neighborhood search for disaster relief and evacuation logistics," *Intelligent Systems, IEEE,* vol. 23, 2008, pp. 14-23.

[12] C. Xie and M. A. Turnquist, "Lane-based evacuation network optimization: An integrated Lagrangian relaxation and tabu search approach," *Transportation Research Part C: Emerging Technologies,* vol. 19, 2011, pp. 40-63.

[13] I. Brajevic, "Artificial Bee Colony Algorithm for the Capacitated Vehicle Routing Problem,"in **Proc.** *European Computing Conference*, Paris, 2011, pp. 239-244.

[14] T. J. Ai and V. Kachitvichyanukul, "A Particle Swarm Optimization for the Vehicle Routing Problem with Simultaneous Pickup and Delivery," *Computers and Operations Research,* vol. 36, 2009, pp. 1693-1702

[15] D. Ponce, "Bio-inspired metaheuristics for the vehicle routing problem," in *Proc. 9th WSEAS International Conference on Applied Computer Science,* 2009, pp. 80-84.

[16] S.-W. Lin, K.-C. Ying, Z.-J. Lee, and F.-H. Hsi, "Applying Simulated Annealing Approach for Capacitated Vehicle Routing Problems," in *Proc. IEEE*

International Conference on Systems, Man and Cybernetics. (SMC '06), Taipei, 2006, pp. 639-644.

[17] L. Zhishuo and C. Yueting, "A Hybrid Ant Colony Algorithm for Capacitated Vehicle Routing Problem," in *Proc. IEEE International Conference on Systems, Man and Cybernetics*, Taipei, Taiwan, 2006, pp. 3907-3911.

[18] M. Yusoff, J. Ariffin, and A. Mohamed, "A Multi-valued Discrete Particle Swarm Optimization for the Evacuation Vehicle Routing Problem," *Advances in Swarm Intelligence,* vol. LNCS 6728, 2011, pp. 182-193.

[19] J. Kennedy, "Swarm intelligence," in *Handbook of Nature-Inspired and Innovative Computing*, 2006, pp. 187-219.

[20] P. Engelbrecht, "Computational intelligence: An Introduction," 2nd ed. West Sussex: John Wiley & Son, 2007.

[21] K. E. Parsopoulos and M. N. Vrahatis, "Parameter selection and adaptation in unified particle swarm optimization," *Mathematical and Computer Modelling,* vol. 46, 2007, pp. 198-213.

[22] C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters,* vol. 85, 2003, pp. 317-325.

[23] R. Guner and M. Sevkli, "A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem," *Journal of Artificial Evolution and Applications,* vol. 2008, pp. 1-9, 2008.

[24] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," in *Proc. 1999 Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Service Center, 1999.

[25] J. Kennedy and R. C. Eberhart, "A Discrete Binary Version of the Particle Swarm Algorithm," in *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, Orlando, FL, USA, 1997, pp. 4104-4108.

[26] W. Mohemmed, N. C. Sahoo, and T. K. Geok, "Solving shortest path problem using particle swarm optimization," *Applied Soft Computing,* vol. 8, 2008, pp. 1643-1653.

[27] H. Izakian, B. T. Ladani, A. Abraham, and V. a. Sn´aˇsel, "A discrete particle swarm optimization approach for grid job scheduling," *International Journal of Innovative Computing,* vol. 6, 2010.

[28] M. Yusoff, J. Ariffin, and A. Mohamed, "A Discrete Particle Swarm Optimization with Random Selection Solution for the Shortest Path Problem," in *Proc. International Conference on Soft Computing and Pattern Recognition*, Paris, France, 2010, pp. 133 – 138.

Marina Yusoff is currently a PHD student in Universiti Teknologi MARA. Prior to this she was a lecturer in Universiti Teknologi MARA and worked as a senior executive of Information Technology in SIRIM Berhad, Malaysia. She holds a Bachelor Degree in Computer Science from the University of Science Malaysia, and MSC in Information Technology from Universiti Teknologi MARA. She is interested in the development of intelligent application, modification and enhancement artificial intelligence techniques include particle swarm optimization, neural network, genetic algorithm, and ant colony. She has presented her research in many conferences locally and internationally.

Junaidah Ariffin is currently a Professor of Civil Engineering and the Head of the Flood-Marine Excellence Centre, Universiti Teknologi MARA Shah Alam, Malaysia. She holds a PhD in water Resources Engineering from the University of Science Malaysia. She is responsible for research projects related to flood forecasting, operations and planning, inundation models, flood evacuations and sediment transport in rivers amounting to more than RM1 million. Her long list of publications on the above can be found from the university website. Currently she teaches the subject on erosion and sedimentation to the Masters graduates and fluid mechanics for the undergraduates. She is also the editor and reviewer of 3 international journals.

Azlinah Mohamed (MSc Artificial Intelligence, University of Bristol UK, PhD Universiti Kebangsaan Malaysia) is a Professor currently working in Universiti Teknologi MARA. Prior to this she was a tutor in University of Bristol and a Research Fellow in Universiti Kebangsaan Malaysia. Prof. Dr. Azlinah's current areas of interest are Hybrid Techniques, Pattern Recognition, and Web-based Decision Support Systems using intelligent agents in electronic government applications. She has presented her research in many conferences and published her work in journals internationally and locally. Besides that, she has also contributed as an examiner and reviewer to many conferences, journals and universities academic activities. In addition, she had also held administration post pertaining to academic development at the university level. Currently, she is the Special Officer on Academic Affairs and Development to the Vice Chancellor of the University.