International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 4 (2012) pp. 578- 588

Discrete Particle Swarm Optimization with a Search Decomposition and Random Selection for the Shortest Path Problem

Marina Yusoff[1], JunaidahAriffin[2], Azlinah Mohamed[1]

[1]Intelligent System Group
Faculty of Computer and Mathematical Sciences
UniversitiTeknologi MARA
40450 Shah Alam, Selangor
Malaysia
marinay, azlinah}@tmsk.uitm.edu.my

[2]Institute for Infrastructure Engineering & Sustainable Management,
Flood-Marine Excellence Centre,
Faculty of Civil Engineering
UniversitiTeknologi MARA
40450 Shah Alam, Selangor,
Malaysia
junaidahariffin@yahoo.com

Abstract: **This paper proposes a discrete particle swarm optimization (DPSO) for the solution of the shortest path problem (SPP). The proposed DPSO termed as DPSO_SPP adopts a new solution representation which incorporates a search decomposition procedure and random selection of priority value. The purpose of this representation is to reduce the searching space of the particles, leading to a better solution. Detailed descriptions of the new solution and the DPSO_SPP algorithm are elaborated. Computational experiments involve an SPP dataset from previous research and road network datasets. The DPSO_SPP is compared with a genetic algorithm (GA) using naive and new solution representation. The results indicate that the proposed DPSO_SPP is highly competitive and shows good performance in both frequency of obtaining an optimal solution and rate of convergence in comparison with the GA_SPP, PSO, and GA. In particular, DPSO_SPP with the use of inertia weight had shown better solution to SPP compared to constriction coefficient (CF). The quality of the solution achieved through DPSO_SPP for all datasets indicated higher potential in achieving the optimum results for SPP, serving as a good ground to further test the algorithm on larger datasets.**

Keywords: discrete particle swarm optimization, genetic algorithm, random selection, search decomposition, priority value, shortest path problem.

I. Introduction

The shortest path problem (SPP) has been widely discussed over several decades. Although extensive research has been done in variants of SPP with the development of an exact algorithm [1][2] and heuristic algorithms [3][4][5], a faster solution is still required. This solution is crucial because SPP is a core problem in several domains such as transportation [6] and vehicle routing [7][8]. Some research [6][9][10][11][12], committed to the improvement of SPP solutions and computer memory usage, enhanced the well-known Dijkstra algorithm and k-Shortest path algorithms. Even though they can yield optimal or near optimal solutions, the algorithmic approach has still required an exhaustive search and consumed more processing time, due to the nature of algorithms.

Taking a different approach, Gen and Lin [13] introduced random key-based GA using a real value priority-based encoding for solving SPP. Their approach improved the ability to reach an optimal solution as well as processing time. However, a method using particle swarm optimization (PSO) outperformed the GA. A range of random discrete priority values (PVs) representing all of the nodes in a network graph offered 95% optimal solution for PSO, but this method required between 100 and 3000 iterations using 30 populations of particles. Another PSO implementation [4] on a stochastic SPP required more than 50 populations of particles for 100% convergence (traversing until destination node) for 10 nodes but less than 80% of convergence for 15 nodes using 150 populations. This work relied on the use of probability distribution of the cost for each edge using real value.

The drawbacks of PSO implementation in these methods inspired us to further the work of Mohemmed et al [5], recognizing the usefulness of discrete value representing the position of the particles and its potential to obtain fast optimal

solutions. Hence, this paper investigated a new solution to SPP with the objective function to find the minimum distance. A single source and a single destination for the SPP are considered. The linear SPP is adopted from Santos et al [10] and Gen and Lin [13] is defined as follows:

Let $G = (N, E)$ be a weighted directed graph where $N = \{N_0, N_1, N_2,...,N_n\}$ represents the set of nodes. N_0 is the source node (the vehicle location), while N_n is the destination node (PFA). E is the set of edges, where for each edge $(i, j) \in E$, a non-negative value C_{ij} represents the length of edge (i, j), or equivalently, the distance between node i and node j. The objective is to find the path from the vehicle location to the PFA with minimum distance. The SPP is mathematically formulated as follows:

$$Minimize \sum_{i=0,j=1}^{n} C_{ij}X_{ij} \quad (1)$$

subject to:

$$\sum_{j}^{n} X_{ij} - \sum_{k=1}^{n} X_{ki} = \begin{cases} 1 & if\ i = 0 \\ 0 & if\ i = 1,\dots,n-1 \\ -1 & if\ i = n \end{cases} \quad (2)$$

$$X_{ij} \in \{0,1\}, (i,j) \in E \quad (3)$$

where

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

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

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

Constraints 2 ensure that the path starts at N_0, end at N_n, and either pass through or avoid every other node j. Constraint 3 indicates the binary decision variables of the problem. The objective function 1 calculates the distance of the shortest path. This paper addresses this problem with the aim to improve the processing time of the algorithm and its solution.

The remainder of this paper is organized as follows. Section II discusses PSO and Section III explains both the naive and new solution representations. Section IV explains the proposed DPSO with search decomposition and random selection. Section V discusses the computational experiment and discussion. Section VI presents the conclusions and recommendation for future research.

II. Particle Swarm Optimization

PSO is a population-based stochastic approach, categorized under swarm intelligence [14], for the solution of continuous and discrete problems. This method was initiated by Kennedy and Eberhart in the mid 1990s [15]. Basically, PSO indicates the velocity and position of particles in a multi-dimensional space. By updating both velocity and position, the global best (*Gbest*) is derived from the simulated social behavior of a group of particles [16]. PSO is able to explore regions of the search space and exploit the search to refine feasible solutions. The ability of searching strategies is based on parameters; acceleration constants and inertia weight [17][18] that have been applied in the PSO algorithm.

A considerable amount of research has been directed toward the modification of canonical PSO to solve several types of continuous problem. The inventors of PSO explored discrete binary PSO with special attention to discrete problems, leading to a new way of updating the position of particles [19] to accommodate discrete binary problems. This was further improved by several studies using a benchmark dataset [20][21] and a real world problem [22][23] of discrete nature. The use of DPSO with multiple (rather than binary) discrete values to solve combinatorial problems in SPP were explored by [5].

In this work, equation 4, 5, and 6 were employed for velocity and position updates. Equation 4 shows the constriction coefficient (CF) value derived from acceleration constant parameters. Equation 5 performs the calculation of new velocity while Equation 6 executes the position updates. Both velocity and position are in positive integers.

$$CF = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|} \quad (4)$$

where $\varphi = c_1 + c_2$.

$$V_{id(new)} = CF * V_{id(old)} + C_1 * r_1 * (Pbest_{(id)} - X_{id(old)} + C_2 * r_2 * (Gbest_{(id)} - X_{id(old)}) \quad (5)$$

$$X_{id(new)} = X_{id} + V_{id(new)} \quad (6)$$

The discrete multi value for particle representation has demonstrated some decrease in the search space that encourages fast convergence and optimal solution. Due to the high number of iterations in the work of Mohemmed et al [5], a modification is needed particularly in the selection of particles. The movement of particles are in the n-dimension, where n is the total number of nodes. The movement of particles can be reduced with the embedded decomposition procedure [24]. A detailed explanation of the naive and new solutions is given in Section IV.

III. Solution Representation

A. Naive solution representation

In the naive solution representation the position values of the particles are represented with PVs; each node has its own PV, and PVs were randomly initialized. Figure 1 demonstrates that all nodes from N_0 to N_n are represented using multi discrete values of *PVs*, in which a row of *PVs* corresponds to one particle. PVs are randomly assigned to each node in a specific range, i.e, [-100, 100] and selected according to its maximum value. It does not matter whether PV values go from minimum to maximum value or from maximum to minimum; the essential is that the PVs be sorted according to this method. The details of the selection process can be found in [5].

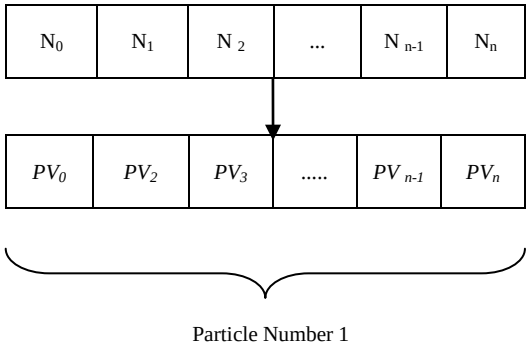


Figure 1. Solution representation [5]

This solution**,** however, required more than 100 iterations for a small-scale SPP, with a number of nodes between 15 and 70. It can be seen that the movement of particles depends on the number of nodes represented in the *n*-dimension's space of search. However, since the values of PVs in the form of multi discrete values present some limitations to the movement of particle positions in the search space, the iteration number should be reduced for faster convergence and to obtain an optimal solution. Therefore, we propose a means of reducing the search space by emphasizing a new way to select PVs and to maintain the use of discrete values for PVs. This solution is explained in the next section.

B. New solution representation

The idea of representing a new particle position came from the fundamental concept of the search process in a graph [25]. We propose a bounded decision for PV selection, using a search decomposition procedure. The procedure, adapted from the work of Mohemmed et al [5], is as shown in Figure 2. This procedure enables some limitation of the search space in the movement of particles. The search decomposition imposes the expansion of nodes from source node/parent node, as illustrated in Figure 3. At the Level 1, a source node is selected, N_0. As shown in Figure 4, N_0 is expanded into two paths/branches, N_0-N_1, and N_0-N_2. This level designates sub-particle number 1. Then, an array of two PVs is considered. The PVs that represent each of nodes, N_1 and N_2, are randomly selected. The selected PV of the particular node is assigned to a minimum value of PVs, for example, PV_2 in Figure 5 is assigned to -100. Then, a path of at Level 2 is expanded, showing the number of PVs and the assignment of a minimum PV for the selected node/ path, respectively. At the Level 1, node N_2 is selected to expand, while at the Level 2, N_5 is selected and finally, at the third level, only one PV value is taken into account, and finally the traversing arrives the destination node, D_1. From this illustration the total distance (fitness value) is calculated from the generated route of N_0 - N_2- N_5 -D_1.

```
1:  Do
2:    If there is no leaf
3:    Failure
4:    Else
5:  Expand nodes
6:  Choose only one path in a random selection
7:     Assign the PV of the selected path with PV_min
8:    Calculate distance of the selected path
9:  While (all nodes expanded or there is no leaf to expand)
```

Figure 2. A search decomposition procedure

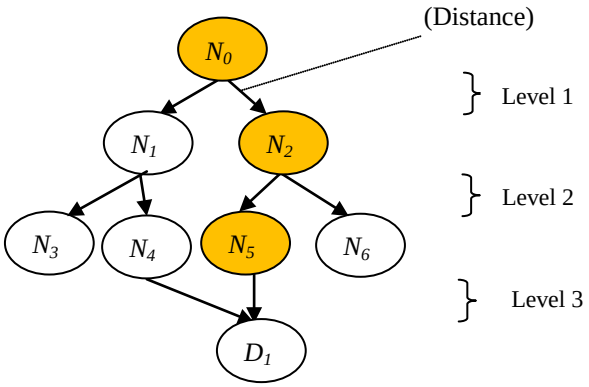


Figure 3. The illustration of search decomposition into branches

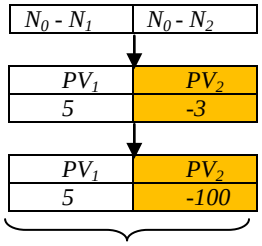


Figure 4. Illustration of a sub-particle for the Level 1

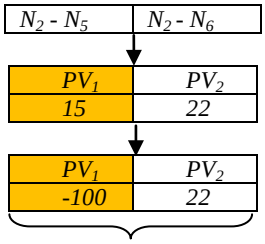


Figure 5. Illustration of a sub particle represented by PVs at the Level 2

In summary, the initialization of PVs depends on the number of nodes, and the path selection is based on the random selection of PVs during the expansion of the graph. Theoretically, this idea would reduce the search space compared to that obtained in the naive solution for SPP. The implementation of this solution representation in DPSO algorithm is discussed in the next section.

IV. DPSO with Search Decomposition Procedure

The new solution representation discussed above is therefore implemented in a modified PSO named as DPSO_SPP, as shown in Figure 6. The algorithm starts with the normal process of PSO initialization, which initializes the number of populations, coefficient values, and then C_1 and C_2 in step 2 and step 3, respectively. Step 4 is the initialization PV and velocities. Step 5 performs the steps for the search decomposition with random selection. In this step, only one path is selected and node with PV_{min} is selected upon selection of the path as demonstrated in Figure 6. After all the 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 proceeds from step 7 until step 20 until a maximum iteration is achieved. In this iteration, each particle is respectively updated with a new velocity update and position. The new velocity and new position value are in the form of positive integers. Then, PV for all sub-particles is updated using step 11. Step 12 performs the decomposition and random selection of PV. $Pbest_{(new)}$ and $Gbest_{(new)}$ are calculated in step 13 and 14, respectively. Steps 15 through 19 are the conditions for the selection of the best current fitness for each iteration.

1: Begin
2: Initialize number of population
3: Declare C_1 and C_2
4: Initialize PV, V_{min} and V_{max} for all particles in random
5: Perform search decomposition procedure
6: Calculate Pbest and Gbest
7: Do
8: For each particle
9: Calculate $V_{(new)}$using equation (2.1)
10: Calculate $PV_{(new)}$using equation (2.2)
11: Update PV for all sub particles
12: Perform step 5
13: Calculate $Pbest_{(new)}$
14: Calculate $Gbest_{(new)}$
15: If ($Gbest_{(new)} > Gbest$)
16: Assign $Gbest_{(old)}$ as the best current fitness
17: If ($Gbest_{(new)} =< Gbest$)
18: $Gbest_{(old)} = Gbest_{(new)}$
19: Assign $Gbest_{(new)}$ as the best current fitness
20: While (maximum iteration is achieved)
21: End

Figure 6. DPSO_SPP algorithm

V. Computational Experiment and Discussion

This section discusses the results of implementing the new solution compared with the naive solution for SPP. A comparative study was done comparing the results of the proposed DPSO (DPSO_SPP), GA using the new solution (GA_SPP), PSO using the naive solution, and GA using the naive solution. The main aim was to observe any improvement in solutions when compared to the previous solution.

A. Experimental setup

Parameters as suggested in [5] are applied as shown in Table 1. The value of inertia weight is selected from the range of this parameter suggested by Shi & Eberhart [26]. Routes for the road network datasets were obtained from GIS MAP software. These routes were transformed into a graphic abstraction. This graph was then transformed into an adjacency matrix comprising a distance value for the valid node, with a value of 0 representing the invalid node. All experiments were conducted on PCs with Intel(R) Core(TM) 2 Duo E8400 3.00 GHz and 2.00 GB RAM under Windows Vista. The algorithms were applied using Java language. The performance of the DPSO_SPP, GA_SPP, PSO and GA using a SPP0 dataset from [5] and SPP1 until SPP13 are datasets from a road network dataset during flash flood evacuation in Kota Tinggi. Table 2 is the list of SPP datasets. Routes for the road network datasets were obtained from GIS MAP software. These routes were transformed into a graphic abstraction. The graph was then transformed into an adjacency matrix comprising a distance value for the valid node, with a value of 0 representing the invalid node.

Table 1. List of parameters

Parameter	Value
PV_{max}	-100
PV_{min}	100
C_1	2.05
C_2	2.05
Initial V_{min}	-10
Initial V_{max}	10
Initial weight, *w*	1.2 [26]

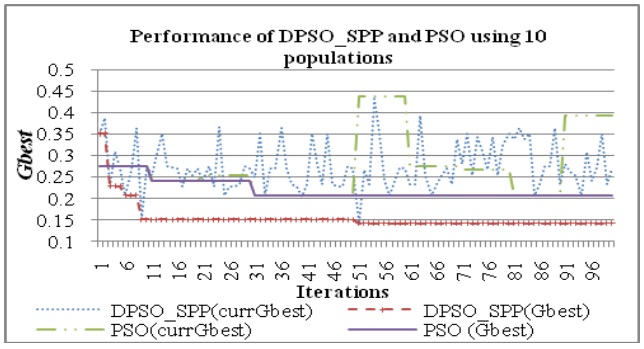
Table 2. List of datasets for SPP

Dataset	Number of node	Dataset	Number of node
SPP0 [5]	20	SPP7	30
SPP1	13	SPP8	22
SPP2	36	SPP9	29
SPP3	13	SPP10	33
SPP4	22	SPP11	26
SPP5	26	SPP12	19
SPP6	37	SPP13	38

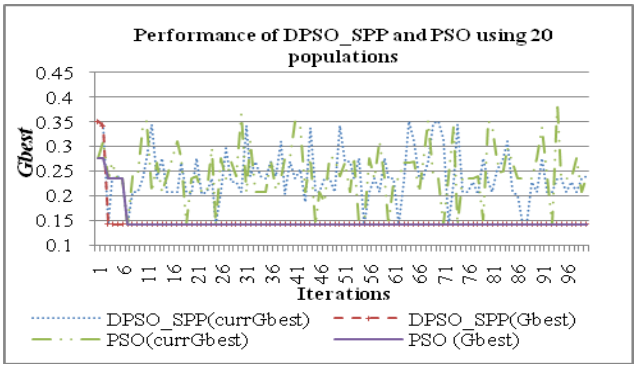
B. Performance of DPSO_SPP in terms of obtaining an optimal solution

The results tabulated in this section are based on the application of CF to DPSO_SPP as recommended from [5]. To illustrate the performance of the new solution representation, all of the algorithms were tested using two population numbers, i.e.10 and 20 populations. Figure 7(a) and 7(b) show the results of DPSO_SPP and PSO with 100 iterations for 10 populations and 20 populations, respectively. The graph in Figure 7 shows the current *Gbest* value (*currGbest*) and the best current fitness (*Gbest*). Figure 7(a) demonstrates near-optimal solution achievement at each of iteration, DPSO_SPP obtained optimal solution (*Gbest*=1.42) at 53th iteration, since in this method the movement of particles is bounded in a limited range with the implementation of a decomposition procedure.

The proof of this implementation can be seen in Figure 7(b), which shows that DPSO_SPP with 20 populations obtained a fast optimal solution at the 3rd iteration whereas PSO obtained the optimal solution at the 8th iteration. It is interesting to note that DPSO_SPP has shown potential in obtaining a higher rate of optimal solutions for SPP. The results recorded in Figure 7(a) and 7(b) indicate that the size of the population had influenced the solutions. It is reasonable to expect that with more random particles moving in the search space, the possibility of finding the best fitness particles is higher.



(a)



(b)

Figure 7. Comparisons of DPSO_SPP and PSO using SPP0 dataset (a) 10 populations, (b) 20 populations

Furthermore, all algorithms were tested to see the frequency of obtaining an optimal solution based on 30 experiments using the SPP0 dataset and three different sized populations (Table 3), giving attention to the iteration numbers. Overall, the new solution representation showed a higher percentage of optimal solutions. DPSO_SPP and GA_SPP obtained 25% to 100% for the optimal solution compared to PSO and GA with less than 20%.

Table 3. Performance of algorithms based on frequency of obtaining an optimal solution

Number of Iterations		Frequency of obtaining optimal solution (%)					
		10	**20**	**30**	**40**	**50**	**100**
Algorithm	**Population size**						
DPSO_SPP	10	26.7	50	80	80	26	**100**
	20	66.7	80	93.3	**100**	**100**	**100**
	30	86.7	90	**100**	**100**	**100**	**100**
GA_SPP	10	33.3	53.3	40	93.3	29	**100**
	20	46.7	83.3	93.3	96.7	**100**	96.7
	30	26	93.3	96.7	**100**	**100**	**100**
PSO	10	3.3	3.3	0	3.3	0	0
	20	3.3	6.7	10	0	13.3	3.3
	30	13.3	3.3	16.7	20	10	10
GA	10	0	6.7	6.7	10	0	6.7
	20	10	3.3	10	6.7	0	3.3
	30	10	0	6.7	3.3	3.3	13.3

It is somewhat surprising that DPSO_SPP at the 30th iteration using population size of 30 achieved 100% optimal solutions, while GA_SPP obtained 100% at the 40th iteration, using a similar sized population. This finding shows that the capability of DPSO_SPP is more promising with the incorporation of search decomposition with random selection. In addition, the results for GA_SPP support the implementation of a new solution representation, as demonstrated in Table 3.

In contrast to the solution representation applied in Mohemmed et al [5] that requires more than 100 iterations, the new solution representation embedded in DPSO_SPP requires only 30 iterations with 30 populations to reach an optimal solution for all of the 30 experiments. The results were competitive to GA_SPP where it requires 40 iterations to successfully obtain an optimal solution for a similar number of experiments. Although the experiments involved a small dataset, based on the solution quality that was achieved using DPSO_SPP, it indicates that DPSO_SPP has a higher potential for finding a fast optimal solution for SPP. The above results indicate that the proposed solution representation and DPSO_SPP can be considered for different sizes of SPP datasets.

However, further results to confirm this solution is demonstrated using the 13 road network dataset. The similar parameters were used; 30 populations, 30 experiments with a

maximum of 30 iteration. The comparison results of using CF and inertia weight = 1.2 for DPSO_SPP and PSO are based on percentage of failure rate of convergence and the percentage of obtaining optimal solutions. The results are shown in Table 4. For SPP1, neither DPSO_SPP nor GA_SPP datasets had a failure rate, while DPSO_SPP obtained 100% optimal solution. However, other algorithms have failed to achieve an optimal solution with 96.67% of convergence using CF. In contrast PSO failed to generate any result using inertia weight. In this case, CF and inertia weight influenced the results of PSO.

The performance of DPSO_SPP and GA_SPP for SPP2, SPP3, SPP5, and SPP11 proved better than PSO and GA. DPSO_SPP and GA_SPP may have a good sequence of PV in finding valid nodes with a minimum distance. Subsequently, it can be seen that the branching procedure with random selection of PV plays a significant role in obtaining an optimal solution.

For SPP4, the results for DPSO_SPP was found to perform better than the other algorithms using inertia weight, providing 33.33% frequency of getting optimal solutions whereas GA_SPP had only obtained 16.67%. So far, the findings of SPP0, SPP1, SPP2, SPP3, SPP4, SPP5, and SPP11 seemed consistent, with DPSO_SPP and GA_SPP revealing good performance with a new solution representation embedded in them.

However, the results of SPP6 provide 20% frequency of optimal solution for DPSO_SPP using CF. The same results obtained by GA_SPP. In contrast, DPSO_SPP provides an approximate 76.67% (using CF) for the failure rate of convergence while GA_SPP with 80% failure rate. The use of inertia weight for DPSO_SPP has reduced the percentage of failure rate while the percentage of getting the optimal solution is improved to 23.33%. The use of inertia weight seems to give better result when compared to CF for this dataset.

Table 4. Comparison results using road network dataset

Dataset	DPSO_SPP				GA_SPP		PSO [5]				GA [5]	
	CF		Inertia weight				CF		Inertia weight			
	FR (%)	F (%)	FR (%)	F (%)	FR (%)	F (%)	FR (%)	F (%)	FR (%)	F (%)	FR (%)	F (%)
SPP1	**0**	100	0	**100**	0	0	96.67	0	-	-	96.67	0
SPP2	0	100	0	**100**	0	100	-	-	-	-	-	-
SPP3	0	100	0	**100**	0	100	-	-	-	-	-	-
SPP4	0	3.33	**0**	**33.33**	3.33	16.67	96.7	0	-	-	96.7	0
SPP5	0	100	0	**100**	0	100	-	-	-	-	-	-
SPP6	76.67	20	**66.67**	**23.33**	80	20	-	-	-	-	-	-
SPP7	16.67	83.33	10	**90**	10	90	-	-	-	-	-	-
SPP8	6.67	93.33	**0**	**96.67**	86.67	0	-	-	-	-	-	-
SPP9	6.67	93.33	**0**	**93.33**	13.33	0	-	-	-	-	-	-
SPP10	10	60	**3.33**	**56.67**	5	0	-	-	-	-		
SPP11	0	100	0	**100**	0	100	-	-	-	-	-	-
SPP12	0	100	**0**	**100**	0	100	0	73.33	0	73.33	0	83.33
SPP13	6.67	90	**0**	**100**	0	73.33	26.67	73.33	30	70	23.33	70

* FR- Failure rate of convergence (%), F - frequency of obtaining optimal solution (%)

For SPP7, DPSO_SPP with an inertia weight outperformed DPSO_SPP with CF. GA_SPP offered similar results to DPSO_SPP. This may be resulted from fewer branches generated from this dataset for during search decomposition. In contrast, both PSO and GA did not give any results. The frequency of obtaining optimal solution has increased about 3.34% compared to the use of CF for DPSO_SPP with inertia weight for SPP8. The improvement is also in its convergence rate. The DPSO_SPP with inertia weight has outperformed other algorithm. The results for DPSO_SPP with inertia weight for SPP9, SPP11, and SPP12 have shown a similar performance.

The above results had indicated that DPSO_SPP with weight perform better that DPSO_SPP with CF in frequency of obtaining an optimal solution. However, results for DPSO_SPP with inertia weight for SPP10 provides a decrease of 3.33% compared to DPSO_SPP using CF. On the other hand, it provides 3.33% of failure of convergence which is slightly lower than with CF. Compared to GA_SPP there was no optimal solution obtained. Overall, DPSO_SPP with the use of inertia weight = 1.2 had shown better solution to SPP compared to CF. Thus, a further comparison result for the four algorithms is shown in Appendix 1, 2, and 3. The comparison results is based on the use of inertia weight = 1.2 using 30 experiments and 30 populations with the maximum of 200 iterations. On the whole, DPSO_SPP algorithm has shown better performance compared to GA_SPP, PSO, and GA in fitness value and processing time.

C. Discussion

This paper enhanced of the work of Mohemmed et al [5] due to the use of discrete value (positive integer) that represents the position of the particles in PSO. It is somewhat surprising that DPSO_SPP with random selection as the new solution representation for SPP solution gave significant results for all road network dataset in its failure rate and frequency of obtaining an optimal solution. These results are due to the search decomposition procedure with random selection of the PV that was embedded in the new solution provides a boundary to the searching space. Consequently, the probability of getting optimal solution is higher and has reduced the failure rate of convergence compared to the output given by the solution of [5]. Compared to the solution of [5], which required more than 100 iterations, the new solution representation embedded in DPSO_SPP required only 30 iterations using 30 populations to obtain 100% optimal solution for 30 experiments for dataset SPP0, SPP2, SPP3, SPP5, SPP11, SPP12 and SPP13.

Although GA_SPP was shown to be competitive with DPSO_SPP in terms of failure rate and percentage of optimal solution for some datasets, the failure rate obtained for GA_SPP is higher than that of DPSO_SPP, showing that GA_SPP achieved a lower quality solution compared to that of DPSO_SPP. This is because the formula for new velocity and new position adopted from the canonical PSO played an important role in finding a better solution for DPSO_SPP, whereas GA_SPP has no updating formula for PV. GA_SPP only perform its crossover to produce new off-spring.

Even though the experiments involve a small dataset, the quality of the solution achieved through DPSO_SPP for all datasets has indicated higher potential in achieving the optimum results for spp. In addition, the use of inertia weight has shown better performance to DPSO_SPP compared to the use of cf parameter in velocity updates. the finding indicate that the use of inertia weight (inertia = 1.2) that was suggested in Shi and Eberhart [26], provides good results compared to CF (CF > 4) as suggested in Mohammed et al [5].

VI. Conclusion and Recommendation

This study demonstrates the improvement of the solutions for SPP when using a DPSO with search decomposition and random selection of PV. Compared to the naive solution representation, which required more than 100 iterations, the new solution representation embedded in DPSO_SPP required only 30 iterations using 30 populations to obtain an optimal solution for all of the 30 experiments for SPP0. The results for DPSO_SPP are competitive with those for GA_SPP which require 40 iterations to successfully obtain optimal solution for a similar number of experiments and similar dataset. Although the experiments involve a small dataset, the quality of the solution achieved through DPSO_SPP for all datasets indicated higher potential in achieving the optimum results for SPP, serving as a good ground to further test the algorithm on larger datasets.

Acknowledgment

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

References

[1] R. K. Ahuja, K. Mehlhorn, J. B. Orlin, and R. E. Tarjan. "Faster algorithms for the shortest path problem", *Journal of Association for Computing Machinery*, 37(2), pp. 213-223, 1990.

[2] R. Seidel."On the all-pairs-shortest-path problem" In *Proceeding of 24th Annual ACM STOC*, pp. 745-749, 1992.

[3] L. Fu, D. Sun, and L. R. Rilett. "Heuristic shortest path algorithms for transportation applications: State of the art", *Journal of Computers and Operation Research*, 33, pp. 3324-3343, 2006.

[4] S. Momtazi, S. Kafi, and H. Beigy. "Solving stochastic path problem: particle swarm optimization approach", *Lecture Notes of Artificial Intelligence*, 5027, pp. 590-600, 2008.

[5] A. W. Mohemmed, N. C. Sahoo, and T. K. Geok. "Solving shortest path problem using particle swarm optimization", *Journal of Applied Soft Computing*, 8(4), pp. 1643-1653, 2008.

[6] R. H. Möhring, H. Schilling, B. Schutz, D. Wagner, and T. Willhalm. "Partitioning graphs to speedup Dijkstra's algorithm", *ACM Journal of Experimental Algorithmics*, 11(28), pp. 1-29, 2006.

[7] S.W. Lin, K. C.Ying, Z.-J. Lee, and H. S. Chen. "Vehicle routing problems with time windows using simulated annealing", In *Proceeding of IEEE International*

Conference on Systems, Man and Cybernetics.(SMC '06), 2006, pp. 645-650, 2006.

[8] T. J. Ai, and V. Kachitvichyanukul. "A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery", *Journal of Computers and Operations Research*, 36(5), pp.1693-1702, 2009.

[9] M. H. Xu, Y. Q. Liu, Q. L. Huang, Y. X. Zhang, and G. F. Luan. "An improved Dijkstra's shortest path algorithm for sparse network", *Journal of Applied Mathematics and Computation*, 185, pp. 247-254, 2007.

[10] L. Santos, J. Coutinho-Rodrigues, and J.R. Current."An improved solution algorithm for the constrained shortest path problem", *Journal of Transportation Research Part B 41*, pp.756-771, 2007.

[11] A. Sedeño-Noda, and C. González-Martín. "On the K shortest path tree problem", *European Journal of Operational Resear*ch, 202, pp.628-635, 2010.

[12] J. B. Orlin, K. Madduri, K. Subramani, and M. Williamson." A faster algorithm for the single source shortest path problem with few distinct positive lengths", *Journal Discrete Algorithms*, 8, pp.189-198, 2010.

[13] M. Gen, and L. Lin."A new approach for shortest path problem by random key-based GA", In *Proceeding of Genetic and Evolutionary Computation (GECCO)*, pp. 1411-1412, 2006.

[14] A. P. Engelbrecht. *Computational Inteligence: An Introduction*, John Wiley & Sons, pp. 10, 2002.

[15] R. C. Eberhart, and J. Kennedy. "A new optimizer using particle swarm theory", In *Proceeding of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39 - 43, 1995.

[16] A. R. Guner, and M. Sevkli. "A discrete particle swarm optimization algorithm for uncapacitated facility location problem", *Journal of Artificial Evolution and Applications*, 2008(10), pp. 1-9, 2008.

[17] A. P. Engelbrecht, Computational inteligence: An introduction: John Wiley & Sons, pp. 187-195, 2002.

[18] Y. Shi and R. C. Eberhart. "Empirical study of particle swarm optimization", In *Proceedings of the 1999 Congress on Evolutionary Computation*, 1999.

[19] J. Kennedy and R. C. Eberhart. "A discrete binary version of the particle swarm algorithm", In *Proceeding of IEEE International Conference on Systems, Man, and Cybernetics*, pp. 4104-4108, 1997.

[20] R. C. Eberhart and Y. Shi. "Comparing inertia weights and constriction factors in particle swarm optimization", In *Proceeding of Congress on Evolutionary Computation*, pp. 84 - 88, 2000.

[22] L. T. Bui, O. Soliman, and H. A. Abbass. "A modified strategy for the constriction factor in particle swarm optimization", *Lecture Notes in Computer Science*, 4828, Springer-Verlag Berlin / Heidelberg, pp. 333-344, 2007.

[23] T. Gong and A. L. Tuson. "Particle swarm optimization for quadratic assignment problems-a forma analysis approach", *Journal of Computational Intelligence Research*, 4, pp. 177-185, 2008.

[24] L. Fu, D.Sun, and L. R. Rilett." Heuristic shortest path algorithms for transportation applications: State of the art", *Journal of Computers and Operation Research*, 33, pp.3324-3343, 2006.

[25] S. Russell, and P. Norvig. *Artificial intelligence: A modern approach*, 3rd ed., Pearson Education, 2003.

[26] Y. Shi and R. C. Eberhart. "A Modified Particle Swarm Optimizer", In *Proceeding of IEEE International Conference on Evolutionary Computation*, pp. 69 - 73, 1998.

Author Biographies

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.

Appendix 1: Comparison results for SPP0 until SPP4 using inertia weight

Dataset		PSO [5]			GA [5]			DPSO_SPP			GA_SPP		
		Total Distance (KM)	PT (s)	Iter	Total Distance (KM)	PT (s)	Iter	Total Distance (KM)	PT (s)	Iter	Total Distance (KM)	PT (s)	Iter
SPP0	Avg	0.227	0.008	1	0.221	0.010	1	**0.217**	**0.005**	1	0.218	0.009	1
	Min	0.142	0.000	1	0.142	0.000	1	0.142	0.000	1	0.142	0.000	1
	Max	0.351	0.016	1	0.310	0.031	1	**0.275**	0.016	1	0.349	0.032	1
	Std. Dev	0.045	0.008	0	0.040	0.009	0	0.035	0.008	0	0.048	0.010	0
SPP1	Avg	-	-	200	-	-	200	**1.248**	**0.012**	1	1.254	0.014	1
	Min	-	-	200	-	-	200	1.219	0.000	1	1.219	0.000	1
	Max	-	-	200	-	-	200	**1.266**	0.016	1	1.270	0.063	1
	Std. Dev	-	-	0	-	-	0	0.023	0.007	0	0.021	0.011	0
SSP2	Avg	-	-	200	-	-	200	38.550	**0.015**	1	38.550	0.017	1
	Min	-	-	200	-	-	200	38.549	**0.005**	1	38.549	0.009	1
	Max	-	-	200	-	-	200	38.580	**0.023**	1	38.580	0.030	1
	Std. Dev	-	-	0	-	-	0	0.006	**0.003**	0	0.006	0.003	0
SSP3	Avg	-	-	200	-	-	200	**3.565**	**0.015**	1	3.586	0.019	1
	Min	-	-	200	-	-	200	3.507	0.000	1	3.507	0.000	1
	Max	-	-	200	-	-	200	**4.521**	0.031	1	4.521	0.032	1
	Std. Dev	-	-	0	-	-	0	0.202	0.008	0	0.261	0.015	0
SSP4	Avg	-	-	200	-	-	200	**2.410**	**0.003**	17	2.447	0.004	11
	Min	-	-	200	-	-	200	**1.794**	0.000	1	2.380	0.000	1
	Max	-	-	200	-	-	200	**2.694**	0.031	91	2.879	0.031	40
	Std. Dev	-	-	0	-	-	0	0.184	0.008	20	0.147	0.008	12

* PT - processing time (second), iter - number of iteration

Appendix 2: Comparison results for SPP5 until SPP9 using inertia weight

Datasets		PSO [5]			GA [5]			DPSO_SPP			GA_SPP		
		Total Distance (KM)	PT (s)	Iter	Total Distance (KM)	PT (s)	Iter	Total Distance (KM)	PT (s)	Iter	Total Distance (KM)	PT (s)	Iter
SPP5	Avg	-	-	200	-	-	200	**1.979**	0.011	**2**	2.243	0.010	1
	Min	-	-	200	-	-	200	1.857	0.000	1	1.857	0.000	1
	Max	-	-	200	-	-	200	**2.744**	0.031	5	3.133	0.031	2
	Std. Dev	-	-	0	-	-	0	0.278	0.011	1	0.400	0.009	0
SPP6	Avg	-	-	200	-	-	200	**4.068**	**0.003**	**20**	4.090	0.006	30
	Min	-	-	200	-	-	200	4.047	0.000	1	4.047	0.000	1
	Max	-	-	200	-	-	200	4.147	0.016	42	4.147	0.016	103
	Std. Dev	-	-	0	-	-	0	0.041	0.006	12	0.050	0.008	28
SPP7	Avg	-	-	200	-	-	200	**2.546**	**0.008**	1	2.553	0.011	1
	Min	-	-	200	-	-	200	2.538	0.000	1	2.538	0.000	1
	Max	-	-	200	-	-	200	2.770	0.016	2	2.770	0.031	3
	Std. Dev	-	-	0	-	-	0	0.043	0.008	0	0.059	0.011	0
SPP8	Avg	-	-	200	-	-	200	**2.421**	**0.006**	1	2.456	0.011	1
	Min	-	-	200	-	-	200	2.288	0.000	1	2.456	0.000	1
	Max	-	-	200	-	-	200	2.456	**0.016**	3	2.456	0.078	3
	Std. Dev	-	-	0	-	-	0	0.069	0.008	0	0.000	0.018	0
SPP9	Avg	-	-	200	-	-	200	**1.727**	0.011	1	1.769	0.009	1
	Min	-	-	200	-	-	200	1.719	0.000	1	1.719	0.000	1
	Max	-	-	200	-	-	200	1.951	0.031	2	2.299	0.016	2
	Std. Dev	-	-	0	-	-	0	0.043	0.010	0	0.128	0.008	0

Appendix 3: Comparison results for SPP10 until SPP13 using inertia weight

Dataset		PSO [5]			GA [5]			DPSO_SPP			GA_SPP		
		Total Distance (KM)	PT (s)	Iter	Total Distance (KM)	PT (s)	Iter	Total Distance (KM)	PT (s)	Iter	Total Distance (KM)	PT (s)	Iter
SPP10	Avg	-	-	200	-	-	200	**3.458**	0.011	1	3.474	0.009	1
	Min	-	-	200	-	-	200	3.285	0.000	1	3.453	0.000	1
	Max	-	-	200	-	-	200	3.548	0.031	2	3.548	0.032	2
	Std. Dev	-	-	0	-	-	0	0.061	0.010	0	0.028	0.010	0
SPP11	Avg	-	-	200	-	-	200	**1.720**	0.004	1	1.725	0.013	1
	Min	-	-	200	-	-	200	1.720	0.000	1	1.720	0.000	1
	Max	-	-	200	-	-	200	1.720	0.016	1	1.877	0.031	1
	Std. Dev	-	-	0	-	-	0	0.000	0.007	0	0.029	0.009	0
SPP12	Avg	0.779	0.088	1	0.773	0.008	1	**0.771**	**0.006**	1	0.771	0.013	1
	Min	0.771	0.000	1	0.771	0.000	1	0.771	0.000	1	0.771	0.000	1
	Max	0.937	0.031	1	0.782	0.032	1	0.771	0.032	1	0.771	0.032	1
	Std. Dev	0.030	0.009	0	0.004	0.010	0	0.000	0.009	0	0.000	0.010	0
SPP13	Avg		-	200	-	-	200	**0.603**	**0.005**	1	0.930	0.010	1
	Min	-	-	200	-	-	200	0.603	0.000	1	0.603	0.000	1
	Max	-	-	200	-	-	200	0.603	0.016	1	2.645	0.032	1
	Std. Dev	-	-	0	-	-	0	0	0.007	0	0.697	0.012	0