Multi-Constrained Route Optimization for Electric Vehicles (EVs) using Particle Swarm Optimization (PSO)

Umair Farooq Siddiqi, Yoichi Shiraishi
Department of Production Science & Technology
Gunma University,
Ohta, Gunma, Japan.
Email: {umair,siraisi}@emb.cs.gunma-u.ac.jp

Sadiq M. Sait
Department of Computer Engineering
King Fahd University of Petroleum & Minerals
Dhahran, Saudi Arabia.
Email: sadiq@kfupm.edu.sa

***Abstract*—Route optimization (RO) is an important feature of the Electric Vehicles (EVs) which is responsible for finding optimized paths between any source and destination nodes in the road network. In this paper, the RO problem of EVs is solved by using the Multi Constrained Optimal Path (MCOP) approach. The proposed MCOP problem aims to minimize the length of the path and meets constraints on total travelling time, total time delay due to signals, total recharging time, and total recharging cost. The Penalty Function method is used to transform the MCOP problem into unconstrained optimization problem. The unconstrained optimization is performed by using a Particle Swarm Optimization (PSO) based algorithm. The proposed algorithm has innovative methods for finding the velocity of the particles and updating their positions. The performance of the proposed algorithm is compared with two previous heuristics: H_MCOP and Genetic Algorithm (GA). The time of optimization is varied between 1 second (s) and 5s. The proposed algorithm has obtained the minimum value of the objective function in at-least 9.375% more test instances than the GA and H_MCOP.**

***Index Terms*—Route Optimization, Simulated Evolution (SimE), Multi Constrained Optimal Path, Electric Vehicles (EVs).**

I. Introduction

The adoption of Electric Vehicles (EVs) is growing rapidly [1]. The rapid rechargeable batteries for the EVs are also available which have recharging time as low as 10 minutes [2], [3]. Route optimization is an important feature of EVs' navigation systems. Using the navigation system, the driver can select a road network and specify the source and destination nodes of his/her journey. The route optimization is responsible for finding optimum paths from the source to the destination node and also accepts the requirements of the driver like minimizing travelling time, recharging time, etc. The calculation of the recharging time and recharging cost is a new feature in EVs because they require recharging of their batteries after travelling a certain distance, for example, after every 100 km which is a much smaller distance as compared to the internal combustion engine based vehicles. The recharging stations for the EVs are located along the roads at appropriate positions to ensure that the majority of the EVs should find a recharging station before draining out their batteries. Many researchers have investigated and proposed policies for the appropriate positioning of the recharging stations and their important characteristics [4], [5]. The route optimization can be performed by two different approaches (i) Multi-Objective Optimization (MO), and (ii) Multi Constrained Optimal Path (MCOP). In the MO approach, multiple optimization objectives are simultaneously optimized and the user can also set priorities to different objectives. However, the MO approach has the drawbacks that (i) it does not allow the user to set constraints in terms of quantitative values, and (ii) it does not guarantee that the resulting path meets some criteria. In the MCOP approach, multiple requirements for the optimization are provided in quantitative terms. The MCOP approach aims to find solutions which minimizes a cost function and satisfy the set of constraints. When some information about the possible solutions are made known to the user then the user can conveniently selects the requirements or constraints in quantitative terms rather than just prioritizing the objectives like in the MO approach. Therefore, this work prefers to tackle the route optimization problem of EVs as a Multi Constrained Optimal Path (MCOP) problem. The MCOP is a well-known NP-complete problem [6], [7], [8], [9]. A slightly different version of the MCOP problem is Multi Constrained Path (MCP) problem that only aims at finding the paths which satisfy the constraints. MCP is also NP-complete when the number of constraints is more than two [8], [9]. EVs' route optimization using the MCOP is a new idea, because the route optimization for internal combustion engine based vehicles is usually performed by using the Multi Objective Optimization (MO) approach. The MCOP approach is mostly used to solve the Quality of Service (QoS) routing problem in computer networks. The proposed MCOP problem aims to minimize the distance and satisfy the requirements on: total travelling time, total time delay due to signals, total recharging time and total recharging cost. The Penalty Function (PF) method [10] is used to transform the MCOP problem into unconstrained optimization problem. The unconstrained optimization can yield feasible solutions which

can satisfy the MCOP problem. Particle Swarm Optimization (PSO) [11], [12] with innovative methods of finding the velocity and displacement of the particles is used to perform the unconstrained optimization. The PSO algorithm works on a number of solutions at a time which is like Genetic Algorithm (GA), however, it offers some advantages over the GA. The PSO is less computational. The computations performed by different particles or elements of the population are more independent in PSO. The PSO has only one operation which is the determination of the globally best position which should be determined by considering all the particles. The remaining operations which are finding the particles' own best positions and moving the particles are independent operations. In GA, the operations like selection of parents and crossover are dependent upon two or more elements of the population. The performance of the proposed algorithm is compared with two other heuristics: (1) H_MCOP which was proposed by Turgay Korkmaz et al. [6] to solve the MCOP problem, and (2) Genetic Algorithm (GA) [13] based algorithm which was proposed by Salman Yussof et al. [14]. The Java based implementations were used to obtain the comparison results. The simulations showed that, when the optimization time is 1 second (s), the proposed algorithm has minimized objective function more than GA and H_MCOP in 16.875% test cases. When the optimization time is 5s, then it has obtained minimum objective function in 9.375% and 19.678% more test cases than GA and H_MCOP respectively. This paper is organized as follows: The second section will describe the previous work in the related areas. The third section will define the problem and describe the proposed algorithms. Fourth section contains the simulation and comparison results. The last section contains the conclusion.

II. Previous Work

The problem of route optimization for the EVs is not previously solved. However, it shares many traits common with the problems of Quality of Service (QoS) routing in computer networks and route optimization in conventional vehicles. Some of the common traits are the use of optimization algorithms, formulation of the multi objective or multi constrained optimization problem, etc. Therefore, it is necessary to present a brief survey of the previous work in those areas. Korkmaz, et al. [6] proposed a heuristic for the MCOP problem that they named as H_MCOP. They used the Penalty Function approach to transform the MCOP problem into unconstrained optimization problem. The H_MCOP consists of two parts. In the first part, a feasible solution is obtained by using the Reverse Dijkstra's Algorithm. The second part uses the Look-Ahead Dijkstra's Algorithm to find another solution which should be equal to or better than the solution found in the first part. Salman Yussof & Ong Hang See proposed a Genetic Algorithm (GA) based heuristic [14] to solve the MCP problem of QoS routing in computer networks. The parents are selected by using the roulette wheel approach. In the crossover operation, a common point that exists in both parents is selected and the sub-paths from that point to the destination nodes are swapped. Mutation is performed by changing the sub-path from a randomly selected point to the destination. Xiao, et al. [7] proposed Limited Path Dijkstra's algorithm for the Multi Constrained Path (MCP) problem. It stores all non dominating paths that are found during the execution. The relaxation operation is extended to reject the new path if it is violating constraints or is dominated by any previously found path. At the end, the set of non-dominating paths is reduced by using several approaches of maintaining best or earliest found paths. Jian Li, et al. [15] proposed a PSO based algorithm for the multi constrained network routing problem. They merged the determination of the velocity and positions into a single crossover like operation. Dakuo, et al. [16] used Genetic Algorithm (GA) with feedback based penalty function for solving the MCOP problem. The penalty factors were determined by taking the product of the last iteration's fitness value and maximum value of the constraint violation. In their method, the constraint which was most violated was optimized first. Zhenjiang, et al. [17] proposed extended depth-first-search algorithm to solve the MCP problem. It used the type of edges (tree, back, or forward) to find and update the paths. It maintained a table structure in which descendants of nodes were stored. After the DFS search was completed, the paths were formed from the entries in the descendant's table.

III. Problem Description and Proposed Algorithms

A. Problem Description

Let us consider a road network $G(V, E)$, where V is the set of vertices and E is the set of edges. Each edge is represented by (i, j), where $i, j \in V$ and i is the starting node and j is the ending node. The paths should be found from the source s to the destination node d. Any path is represented by: $P = \{(s, i), (i, j), ..., (m, d)\}$, $(s, i)(i, j), \&(m, d) \in E$ and $s, i, j, m, d \in V$. In the proposed work, each edge (i, j) is associated with 4 parameters: f_1, f_2, f_3, and f_4. f_1 represents the average time taken by the EV to travel on the edge (i, j). f_2 stores the average time delay due to signals. f_3 stores the recharging time required by the EV and f_4 stores the recharging cost to travel on the edge (i, j). The positions of recharging stations, delay in the queues at the recharging stations and the recharging time on any edge is determined by using the results of the single highway model proposed by Zheng et al. [4]. They suggested that for every 100 km, 5 recharging points are sufficient. The number of recharging points for other lengths is determined by using the same ratio i.e., Number of recharging points:
$R_{points} = \frac{5}{100} \times l$.
In the above equation, l is the length of the edge. The recharging points are placed at equidistant from each other. When any edge has only one recharging point then that should be placed either in the middle or at the start of the edge. The delay due to waiting at any recharging station on the edge is given by:
$R_{wait} = \frac{T}{100} \times 13$ minutes
In the above equation, T is the traffic in terms of number

of EVs on the edge (i,j) and its value can vary between 0 and 100. The recharging time ($t_{charging}$) is considered as 10 minutes per vehicle [2], [4]. The cost of recharging EVs can be varying according to the load on the power system [5]. The average recharging cost (R_{cost}) has been determined as approximately 5 cents/kWh [5]. The capacity of battery is taken as $B_{capacity} = 167kWh$. The travel limit of EV (d_{limit}) is 100 km when the battery is 100% charged. If we assume linear relationship between the distance and the battery level (B_{level}) then the maximum distance that any EV can travel without recharging= $\frac{d_{limit}}{100} \times B_{level}$. Let us consider that (i,j) and (j,k) are any two consecutive edges in the current solution (P), $S_{max}(i,j)$ is the maximum speed of the EV on (i,j), $S_{traffic}(i,j)$ is the average speed of the EV in full traffic (i.e.,T=100) on (i,j), $W_s(i,j)$ is the average waiting time at any signal on (i,j), $N_s(i,j)$ is the number of signals on (i,j), $B_{initial}(i,j)$ is the battery level at the start of the edge (i,j) and its value can vary in [0,100]. The speed of the EV on (i,j) is given by:
$S = min(\frac{S_{Traffic} \times 100}{T}, S_{max})$
The values of functions f_1 to f_4 are determined as:
$f_1(i,j)$= $\frac{l(i,j)}{S(i,j)}$, (where $l(i,j)$= length of the edge)
$f_2(i,j)$= $W_s(i,j) \times N_s(i,j)$,
$[f_3(i,j) , f_4(i,j), B_{initial}(j,k)]$= $Re\_Time\_Cost(D_{(i,j)}, D_{(j,k)}, B_{initial}(i,j))$
The function $Re\_Time\_Cost()$ is shown in Fig. 1. The inputs to the function are $D_{(i,j)}$ and $D_{(j,k)}(1)$. Where $D_{(i,j)}$ stores the location of all (i.e. K) recharging points which exist on the edge (i,j) in terms of their distance from the intersection i. $D_{(j,k)}(1)$ contains the location of the first recharging station on the edge (j,k). The function first checks if the EV with the battery level at the start of the edge is sufficient that it can reach the first recharging station. If the EV cannot reach the first recharging station then it will return ∞ which means that the EV cannot travel on that edge. Otherwise it finds the recharging time (t) and recharging cost $(cost)$ for the EV to travel on the edge (i,j). It also finds the initial battery level $(B_{initial}(j,k))$ for the edge (j,k) which lies next to (i,j) in the current solution.

The MCOP problem for the route optimization in EVs can be defined as:
$Minimize(\sum_{(i,j)\in P} l(i,j))$, subject to: $\sum_{(i,j)\in P} f_k \leq c_k$, for $k = 1, 2, 3, \&4$.
c_k represents the maximum allowed values. The Penalty Function (PF) [11] is one approach to transform any multi constrained optimization problem into unconstrained optimization problem. The above MCOP problem is transformed into unconstrained optimization problem using the PF method and the objective function (Obj) after transformation is given by:
$Obj(P) = Minimize(\sum_{(i,j)\in P} l(i,j) + \sum_{k=1}^{4} r_k[(max(0, \sum_{(i,j)\in P} f_k(i,j) - c_k))^2])$.
The variables r_k for k= 1 to 4 are called as the penalty coefficients. r_k are assigned large values and should be greater than the length of any path (P) from the source to the destination node. The solutions of the unconstrained optimization problem which have $Obj(P) < min(r_1, r_2, r_3, r_4)$ are feasible solutions for the MCOP problem. The proposed algorithm finds an initial solution $(P_{initial})$ before starting the optimization process. Then values of $I_k = \sum_{(i,j)\in P_{initial}} f_k(i,j)$, for k=1 to 4 should be calculated. The user inputs are represented as: u_1, u_2, u_3, & u_4, and their values lie in the range [-100, 100]. The values of the actual constraints (c_1, c_2, c_3, c_4) will be obtained as: c_k = $\frac{u_k}{100} \times I_k + I_k$, for k= 1 to 4.
The proposed algorithm needs to find paths from the source to the destination node in initializing the particles and from any node to the destination node in moving the particles. A function named as $form\_path(u, v, N_{paths})$ is proposed which can find at most N_{paths} from nodes u to v. The function is shown in Fig. 2. The inputs to the algorithm are two nodes (u, v), variable N_{paths} which indicates the maximum limit on the number of paths to be found and variable K that is equal to the number of edges in the road network. In line 2, W is a linear array of K elements which acts as weights of the edges in the road network. The function $random(K)$ returns K random numbers in the range from 1 to 1000. The paths are obtained by using the Dijkstra's Algorithm [18]. In each iteration, the weights of the edges are set to different values which changes the shortest path in that iteration. At the end of the first loop, Q stores a number of distinct paths. In lines 5 to 8, the duplicate paths in Q are removed. Finally, the Q array is returned which contains N_{paths} or lesser number of paths from u to v.

Input: $B_{initial}(i,j)$:battery level at the start of the edge (i,j)
$D_{(i,j)}$= $\{d_1, ..., d_K\}$:locations of K recharging stations on (i,j) $D_{(j,k)}(1)$= d_{K+1}: first recharging station on (j,k)
Output: t: recharging time, $cost$: recharging cost, $B_{initial}(j,k)$: battery level at the start of the edge (j,k) Note: Edge (i,j) is followed by (j,k) in the current solution.

1: $D_{current}$=$\frac{d_{limit}}{100} \times B_{initial}(i,j)$, $t = 0$, $cost = 0$
2: **if** $D_{current} < d_1$ **then**
3: **return** (∞);
4: **else**
5: **for** i= 1 to K **do**
6: **if** $D_{current} \geq d_i$ and $D_{current} < d_{i+1}$ **then**
7: $t = t + t_{charging} + R_{wait}$
8: $B_{current} = \frac{(D_{current} - d_i)}{d_{limit}}$
9: $D_{current} = d_i + d_{limit}$
10: $cost = cost + R_{cost} \times B_{capacity}(1 - B_{current})$
11: **end if**
12: **end for**
13: $B_{initial}(j,k)$= $\frac{D_{current} - l(i,j) \times 100}{d_{limit}}$
14: **return** $(t, cost, B_{initial}(j,k))$
15: **end if**

Fig. 1. Function $Re\_Time\_Cost(D_{(i,j)}, D_{(j,k)}(1), B_{initial}(i,j))$

Input: nodes: u, & v, G=(V,E), N_{paths}, K= Number of edges in G
Output: Q: Array containing at most N_{paths} paths from u to v.

for i= 1 to N_{paths} **do**
  W= $random(K)$
  Q(i)= Apply Dijkstra's Algorithm (u, v)
end for
for i= 1 to $N_{paths} - 1$ **do**
  for j= i+1 to N_{paths} **do**
    if Q(i)== Q(j) **then**
      delete Q(j)
    end if
  end for
end for
return Q

Fig. 2. Function $form\_path(u, v, N_{paths})$.

Input: Particles: $I_1(x_0, x_1, .., x_{m-1})$ and $I_2(y_0, y_1, .., y_{m-1})$
Output: $I_1 - I_2 = d \in Integers$

$cnt = 0$, $d = 0$
for $i = 0; i < m - 1; i++$ **do**
  for $j = 0; j < m - 1; j++$ **do**
    if $x_i == y_j$ **then**
      $cnt = cnt + 1$
    end if
  end for
  if $cnt == 0$ **then**
    $d++$
  end if
end for
return d

Fig. 3. Method of finding difference between two positions: $Diff(I_1, I_2)$

B. Proposed Optimization Algorithm

Let us consider a Swarm of M particles which is represented by: $S = \{P_1, P_2, P_3, .., P_M\}$. Any $P_i \in S$, for i= 1 to M is similar to the P which was described in the previous subsection. However, the representation of P_j is different from P. Let us consider that $P_j = \{p_0, p_1, ..., p_{m-1}\}$, where the values of p_j, for j= 0 to $m - 1$ can be determined as follows:

$$p_j = \begin{cases} s \text{ if j=0} \\ n_x \text{ if } (p_{j-1}, n_x) \in E, n_x \in V \\ null \text{ if } p_{j-1} = d \text{ or } null \end{cases} \quad (1)$$

The proposed work assumes a m-dimensional search space which is represented by $A \in R^m$. For any particle P_i, the dimension R^j contains the possible values for the p_j element of the particle P_i. The objective function value of any particle (P_i) can be determined by using the formula: $Obj(P_i) = \sum_{j=0}^{m-2} l(p_j, p_{j+1}) + \sum_{k=1}^{4} r_k[(max(0, \sum_{j=1}^{m-2} f_k(p_j, p_{j+1}) - c_k))^2]$. (p_j, p_{j+1}) represents an edge $e_x \in E$ whose starting node is p_j and ending node is p_{j+1}. In PSO, the positions of particles change iteratively during the optimization process by adding the velocities into their current positions. The velocity of any particle (P_i) at time or iteration t is represented as $v_{P_i}(t)$. The particles' velocities are also updated iteratively. In PSO, the best position of any particle i.e., $pbest(P_i)$ for the particle P_i can be obtained as: $pbest(P_i(t)) = \arg\min_t(Obj(P_i(t)))$, where t represents the iteration count.

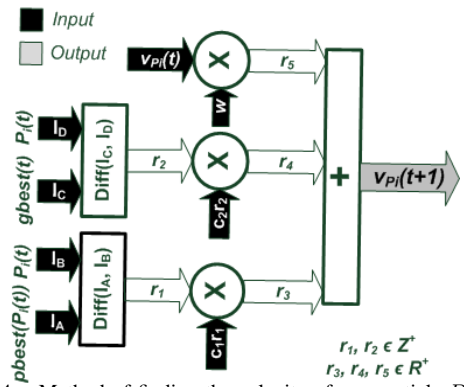

Fig. 4. Method of finding the velocity of any particle $P_i \in S$

The global best position $(gbest(t))$ can be obtained as: $gbest(t) = \arg\min_i(pbest(P_i(t)))$, for $i = 1, 2, .., M$. In the t^{th} iteration, the velocity of the particle for the next iteration i.e. $t + 1$ can be calculated as:

$$v_{P_i}(t+1) = wv_{P_i}(t) + c_1 r_1[pbest(P_i(t)) - P_i(t)] + c_2 r_2[gbest(t) - P_i(t)], for\ i = 1, 2, .., M \quad (2)$$

In (2), the c_1 and c_2 are also real numbers between [0,2]. r_1 and r_2 are randomly generated real numbers between $[0, 1]$. The method proposed to find the difference between any two particles: $I_1(x_0, x_1, .., x_{m-1})$ and $I_2(y_0, y_1, .., y_{m-1})$ is shown in Fig. 3. The idea behind the proposed method is that it returns the number of nodes which exist in the particle I_1 but do not exist in the particle I_2. The method to calculate the velocity using the proposed $Diff(I_1, I_2)$ method is shown in Fig. 4. After the velocities are determined, then they should be added into the particles' current positions in order to move the particles to their new positions. In Fig. 4, the output is the velocity of the particle for the next iteration which is represented by $v_{P_i}(t + 1)$ and is a positive real number. This work also proposes a new method of moving the particles which is shown in Fig. 5. The inputs are: $P_i(t)$, which is the position of the particle in the iteration t, $v_{P_i}(t+1)$ or just v_{P_i}, which is the velocity of the particle in the $t + 1^{th}$ iteration, $C \in Z^+$, which is usually set to 20, $Pb_r \in [0, 1] \in R$, which can be set to a high value like 0.6 and V_r, which is random real number between $[0, v_{P_i}(t + 1) - 1]$. The flow chart assumes that m' is the number of not null elements in any particle. The objective function is determined by calling the function $Obj()$ and $Obj(P) = \infty$ when $P = null$. The outermost loop in each iteration, randomly selects an element from $P_i(t)$ and store it in p_R. Then the function $form\_path()$ is called, to build at-most C number of paths from p_R to the destination node. The paths are stored in the array $Paths[]$. The inner loop is executed for each path in the array $Paths$. P_a contains the portion of the $P_i(t)$ from its first element to until the R^{th} position. The P_b contains the path which is at the j^{th} position in $Paths$. The paths in $Paths$, exists from the node p_R to the destination node. The path t_p is the concatenation of the P_a and P_b paths. The variable I_{min} stores the position (or path) which has minimum value of the objective function among

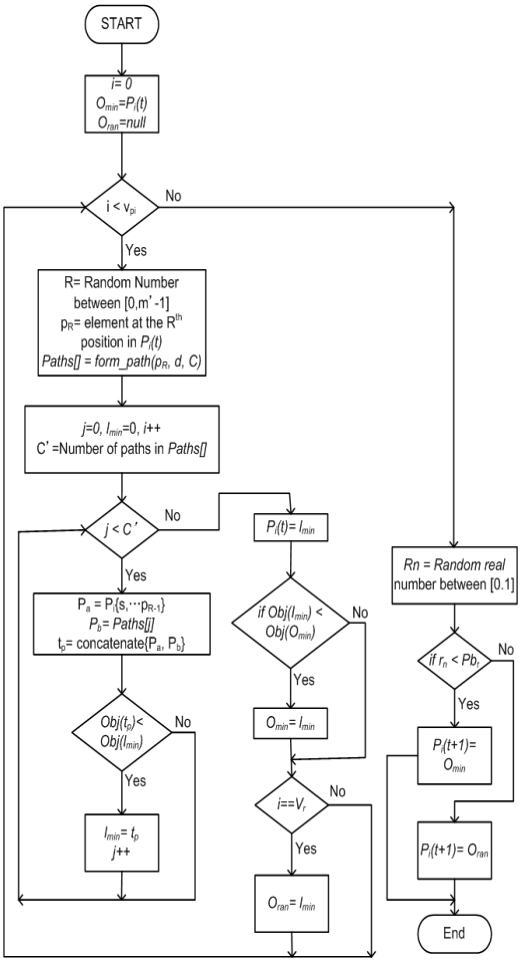

Fig. 5. Method of displacing the particles

all the positions build in the inner loop. The variable O_{min} stores the position which has minimum value of the objective function among all the positions found by the two loops. O_{ran} stores the position which has minimum value of the objective function in the inner loop when the outer loop has iteration count equal to V_r. The last portion of the flowchart returns the position of P_i for the $t+1^{th}$ iteration which is represented by $P_i(t+1)$.

IV. Simulation Results

The proposed algorithm is implemented in Java. The road network of the Dhahran city in Saudi Arabia, which has 108 nodes and 432 edges is considered as an input. The road lengths were extracted from the map by using the map meter. The traffic $(T(i,j))$ values to edges were assigned to integers randomly selected from [0,80]. Number of signals (N_s) values were also assigned to random integers between [0,3]. Average delay at any signal (W_s) was set to 3 minutes. The performance of the proposed algorithm is compared with (i) H_MCOP algorithm which was proposed by Turgay Korkmaz et al. [6] for solving the MCOP problems, and (ii) Genetic Algorithm (GA) based algorithm for solving the MCP problem which was proposed by Salman Yussof et al. [14]. The H_MCOP and GA were also implemented in Java. The algorithms were executed on Intel i5, 2.27 GHz processor based laptop computer with 3GB of RAM. The proposed algorithm was implemented with parameter: r_1 and r_2 were randomly generated between $[0,1]$. c_1, c_2 and w_1 values were set to 0.6, 0.8 and 0.5. The Swarm size is set to 10 and 20 for execution times 1 and 5 seconds respectively. The user input for the constraints, i.e., u_k, for $k = 1,2,3,4$ was set to -40% in all test cases. An instance of a problem consists of finding an optimum path between a source node which is randomly selected from the nodes [1,51] and a destination node which is selected randomly from the [52,108] nodes. The simulations are performed using four sets: set A, set B, set C, & set D. Set A contains 10 instances of the problem, set B contains 20 instances, set C contains 40 instances, and set D contains 80 instances. The GA was implemented with population size of 20 and iterations lasts for 1 sec or 5 sec. The parents in the crossover operation were selected using the roulette wheel approach. The results are shown in Table I. The first column mentions the name of the set, second column mentions the algorithm used. Third, fourth and fifth columns contains the percentage of instances in which the algorithm in column 2 has obtained objective function value which is better than or equal to the objective function value from the algorithm mentioned in the columns 3^{rd}, 4^{th}, and 5^{th}. For example, the first reading is: the proposed algorithm has obtained objective function value better than or equal to GA in 90% instances in set A. The 6^{th} column contains the sum of the two percentages as each algorithm is compared with two other algorithms. The higher the value of the sum the better the algorithm in 2^{nd} column has performed against the other algorithms. Seventh column contains the percentage of instances in the corresponding set in which the algorithm in the second column has obtained the objective function value which is equal to the minimum value as obtained by any algorithm. The last column mentions the number of unique feasible paths that were found by the algorithms in the 2^{nd} column in all instances in their sets. The table also shows the summary of all results which are obtained when execution time is 1 sec or 5 sec. The summary shows that when the execution time is 1 sec, the proposed algorithm has obtained the minimum objective function value in 72.5% instances whereas GA has obtained minimum value in 55.625% and H_MCOP has obtained minimum objective function value in 55.625% instances. The summary of the all results which were obtained at execution time equal to 5 sec shows that the proposed algorithm has obtained minimum objective value in 76.25% instances and GA and H_MCOP in 66.875% and 56.563% instances respectively. The values in the last column indicate that the GA has obtained maximum unique feasible paths than the proposed algorithm and H_MCOP finds the least number of distinct feasible paths. Even though the GA has found more number of feasible solutions but the solutions from the proposed algorithm are better in terms of their objective function values. Therefore, the performance of the proposed algorithm is still better than the GA. The

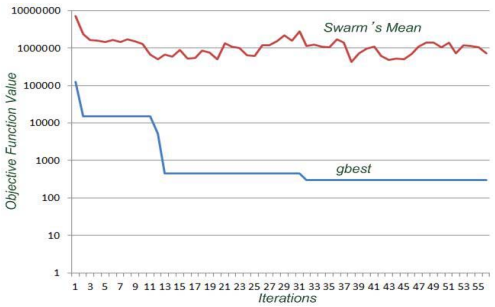

Fig. 6. Optimization curve of the proposed algorithm

optimization curve of the proposed algorithm is shown in Fig. 6, in which $x-axis$ contains the iterations and $y-axis$ contains the objective function values. The line labelled $gbest$ shows the value of the $gbest$ solution and the line labelled as Swarm's mean shows the mean objective function value of the whole swarm. The graph shows that both the $gbest$ and the Swarm's mean values decrease with iterations. The optimization loop was executed for 30 seconds.

TABLE I
SIMULATION AND COMPARISON RESULTS

	Algorithm	Obtains value $\leq$ than the other algorithms					# of Paths
execution time= 1 sec							
		Proposed	GA	H_MCOP	Sum	Min Value	
Set A	Proposed	-	90%	80%	170%*	80%	41
	GA	70%	-	50%	120%	60%	61
	H_MCOP	40%	50%	-	90%	40%	6
		Proposed	GA	H_MCOP	Sum	Min Value	
Set B	Proposed	-	100%	70%	170%*	70%	62
	GA	65%	-	65%	130%	55%	84
	H_MCOP	65%	65%	-	130%	65%	12
		Proposed	GA	H_MCOP	Sum	Min Value	
Set C	Proposed	-	100%	75%	175%*	75%	137
	GA	70%	-	72.5%	142.5%	62.5%	213
	H_MCOP	60%	62.5%	-	122.5%	60%	22
		Proposed	GA	H_MCOP	Sum	Min Value	
Set D	Proposed	-	96.25%	68.75%	165%	65%	273
	GA	63.75%	-	60%	123.75%	45%	428
	H_MCOP	57.5%	62.5%	-	120%	57.5%	48
Summary of Performance at execution time= 1 sec							
Proposed	It obtains minimum objective function value in 72.5% test instances						
GA	It obtains minimum objective function value in 55.625% test instances						
H_MCOP	It obtains minimum objective function value in 55.625% test instances						
execution time= 5 sec							
		Proposed	GA	H_MCOP	Sum	Min Value	
Set A	Proposed	-	100%	80%	180%*	80%	69
	GA	100%	-	80%	180%*	80%	112
	H_MCOP	80%	80%	-	160%	80%	7
		Proposed	GA	H_MCOP	Sum	Min Value	
Set B	Proposed	-	100%	65%	165%*	65%	81
	GA	0.8	-	65%	145%	55%	126
	H_MCOP	55%	55%	-	110%	55%	10
		Proposed	GA	H_MCOP	Sum	Min Value	
Set C	Proposed	-	97.5%	82.5%	180%*	80%	204
	GA	72.5%	-	75%	147.5%	60%	244
	H_MCOP	37.5%	42.5%	-	80%0.8	37.5%	20
		Proposed	GA	H_MCOP	Sum	Min Value	
Set D	Proposed	-	97.5%	81.25%	178, 75%*	80%	332
	GA	88.75%	-	53.75%	142.5%	72.5%	615
	H_MCOP	53.75%	55%	-	108.75%	53.75%	44
Summary of Performance at execution time= 5 sec							
Proposed	It obtains minimum objective function value in 76.25% test instances						
GA	It obtains minimum objective function value in 66.875% test instances						
H_MCOP	It obtains minimum objective function value in 56.563% test instances						

V. CONCLUSION

This paper uses the Multi Constrained Optimal Path (MCOP) approach to perform the route optimization in Electric Vehicles (EVs). The MCOP problem is transformed into unconstrained optimization problem by using the Penalty Function method. The optimization is performed through a Particle Swarm Optimization (PSO) based algorithm with innovative methods of finding the velocity and displacement of the particles. The performance of the proposed algorithm is compared with two previous heuristics: H_MCOP and GA. The simulation shows that when the optimization time was 1 sec, then the proposed algorithm has obtained minimum objective function value in 72.5% test instances which was 16.875% more than GA and H_MCOP. When the optimization time was 5 sec then the proposed algorithm has obtained minimum objective function value in 76.25% test instances which was 9.375% more than GA and 19.687% more than H_MCOP.

REFERENCES

[1] Ahmed Y. Saber & Ganesh K. Venayagamoorthy, "One Million Plug-in Electric Vehicles on the Road by 2015," Proceedings of the 12^{th} International IEEE Conference on Intelligent Transportation Systems, St. Lious, MO, USA; Oct. 3-7, 2009.

[2] Medhi E.-Amoli, Kent Choma, & Jason Stefani, "Rapid-Charge Electric-Vehicle Stations," IEEE Transactions of Power Delivery. Vol. 25, No. 3; July 2010.

[3] Power Energy Syst., "Lithium-ion titanate batteries," 2007. [On-Line] Available: http://www.b2i.cc/Documents/546/50Ah_Datasheet-012209.pdf

[4] Zheng Li, Zafer Sahinoglu, Zhifeng Tao, & K. H. Teo, "Electric Vehicles Network with Nomadic Portable Charging Stations," 2010 IEEE 72^{th} Conference on Vehicular Technology Fall. VTC-2010 Fall. Ottawa, ON, USA; Sept. 6-9, 2010.

[5] Olle Sundstorm & Carl Binding, "Planning Electric-Drive Vehicle Charging under Constrained Grid Conditions," 2010 Internal Conference on Power System Technology, (POWERCON). Zhejiang, China; Oct. 24-28, 2010.

[6] T. Korkmaz & M. Krunz, "Multi-Constrained Optimal Path Selection," INFOCOM 2001, Proc. 20^{th} Annual joint Conference of the IEEE Computer & Communications Societies. Anchorage, AK, USA; April 22-26, 2001.

[7] W. Xiao, Y. Luo, B. H. Soong, A. Xu, C. L. Law, K. V. Lin, "An Efficient Heuristic Algorithm for Multi-Constrained Path Problem," Proc. 2002 IEEE 56^{th} Vehicular Technology Conference 2002. VTC-2002-Fall. Vancouver, Canada. Vol. 3; Sept. 24-28 2002, p. 1317.

[8] Shigang Chen & Klara Nahrstedt, "An Overview of Quality of Service Routing for Next-Generation High-Speed Networks," IEEE Network, vol. 12, issue 6; Nov./ Dec. 1998, p. 64-79.

[9] M.R. Garey and D. S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness.*, W. H. Freeman and Co.; 1997

[10] P. Venkataraman, Applied Optimization with MATLAB Programming, John-Wiley & Sons, Inc., 2002.

[11] Konstantinos E. Parsopoulas and Michael N. Vrahatis, Particle Swarm Optimization and Intelligence, Advances and Applications, 2010 Information Science Reference (IGI Global).

[12] J. Kennedy, R.C. Eberhart, "Particle Swarm Optimization," Proceedings of the IEEE International Conference on Neural Networks, 1995.

[13] S.M. Sait & H. Youssef, Iterative Computer Algorithms with Applications in Engineering, IEEE Computer Society Press; 1999.

[14] Salman Yussof & Ong Hang See, "Finding Multi-Constrained Path using Genetic Algorithm," Proc. 2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications, ICT-MICC 2007, 14-17 May 2007, Penang, Malaysia

[15] Jian Li, H. Cui, Ru Gao, Jia Du. & Jianya Chen, "The Application of an Improved Particle Swarm Optimization for Multi-constrained QoS Routing. 2^{nd} International Workshop on Database Technology & Applications (DBTA)," Wuhan, China, Nov. 27-28, 2010.

[16] He Dakuo, Wang Fuli, "Feedback Penalty Function Based on Genetic Algorithm," Proc. 4^{th} World Congress on Intelligent Control and Automation. Shanghai, China, June 10-14, 2002.

[17] Zhenjiang Li, J.J. G.-L.-Aceves, "Solving the Multi-Constrained Path Selection Problem by Using Depth First Search," Proc. 2nd Int'l Conf. Quality of Service in Heterogeneous Wired/Wireless Networks, Lake Vista, FL, 24 August, 2005.

[18] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, & Clifford Stein, Introduction to Algorithms. 3^{rd} Edition, MIT Press 2009.