Vol.2011-MPS-85 No.15
2011/9/16

Multi Constrained Route Optimization for Electric Vehicles (EVs) using Simulated Evolution (SimE)

Umair F. Siddiqi,[†1] Yoichi Shiraishi[†1] and Sadiq M. Sait [†2]

Route Optimization (RO) is an important feature of Electric Vehicles (EVs) navigation systems. This work performs the RO for EVs using the Multi Constrained Optimal Path (MCOP) problem. The proposed MCOP problem aims to minimize the length of the path and meets constraints on travelling time, time delay due to traffic signals, recharging time and recharging cost. The optimization is performed through a design of Simulated Evolution (SimE) which has innovative problem specific goodness and allocation operations. The simulations using Java shows that the proposed algorithm has obtained performance equal to or better than GA. It requires memory which is 1.65 times lesser than GA. Therefore, we can conclude that it is suitable for implementation on the embedded system of an EV.

1. Introduction

The adoption of Electric Vehicles (EVs) is growing rapidly[1]. Route optimization is an important feature of EVs' navigation systems. The EVs' navigation systems contain road networks of different geographical areas. The driver wishes to find optimum paths between any source (s) and destination (d) nodes in the road networks. The driver can also specify to find routes which are optimum w.r.t. minimum length, minimum traffic, minimum recharging time and recharging cost. The calculation of 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 of EVs is performed at recharging stations which 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[2],[3].

The Multi Constrained Optimal Path (MCOP) approach can be used to perform the route optimization in which the driver can set the requirements. The MCOP approach aims to minimize the length of the path and satisfy the constraints on: total travelling time, total time delay due to signals, total recharging time and total recharging cost. The Penalty Function (PF) method[4] is used to transform the MCOP problem into unconstrained optimization problem. The unconstrained optimization can yield feasible solutions which can satisfy the MCOP problem. This work proposes a Simulated Evolution (SimE)[5] based algorithm with innovative and problem specific goodness and allocation operations to perform the optimization. The SimE Algorithm starts from an initial solution and then, following an evolution based approach, it seeks better solution from one generation to the next. SimE can be compared with Genetic Algorithm (GA)[5] which works with a set of solutions but requires more memory than SimE. The proposed algorithm aims to obtain feasible solutions in execution time which is comparable to GA while consuming lesser memory than GA. The advantages of the proposed SimE-based algorithm are that: (i) embedded systems with limited amount of memory can be used for its implementation, or (ii) Due to its simple operations and memory efficiency, Field Programmable Gate Arrays

†1 Department of Production Science & Technology, Gunma University, Gunma, Japan
†2 King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia

(FPGAs) or Application Specific Integrated Circuits (ASICs) can also be used to implement the proposed algorithm. The proposed algorithm is suitable for implementation in the embedded system of an actual EV's navigation system. This paper is organized as follows: The second section will describe the problem in detail. Third section contains the detail of the proposed algorithm. Fourth section contains the simulation results and comparison of the proposed algorithm with another algorithm. The last section contains the conclusion.

2. Problem Description

2.1 Description of the Road Network for Electric Vehicles

Let us consider a road network $G(V, E, R, EV)$, where V is the set of vertices or nodes, E is the set of edges, R is the set of recharging stations and EV is the set of electric vehicles travelling on the road network. Nodes represent the intersections in the road network and edges represent the roads which join the intersections. The set of nodes is represented as: $V = \{n_0, n_1, n_2, ..., n_{N_n-1}\}$, when the total number of nodes $= N_n$ and n_0 to n_{N_n-1} represent the intersections in the road network. The set of edges is represented as: $E = \{e_0, e_1, .., e_{N_e-1}\}$, where N_e is the total number of edges. Any edge $e_i = (n_x, n_y)$, for $i = 0$ to $N_e - 1$ and n_x and $n_y \in V$ and n_x is the node which is starting the edge e_i and n_y is the node which is ending the edge e_i. $e_i \neq e_j, \forall i \neq j$. The set R which contains all the recharging stations in the road network is represented as $R = \{r_0, r_1, ...r_{N_R-1}\}$, where N_R is the total number of recharging stations in the road network. The recharging stations are situated along the edges in the road network. The set of electric vehicles, i.e., $EV = \{EV_0, EV_1, ..., EV_{N_{EV}-1}\}$, where N_{EV} is the total number of EVs in the road network. The properties associated with different components of the road network are shown in Table 1.

2.2 Formation of the Multi Constrained Optimal Path (MCOP) Problem

Any EV_k needs to perform the route optimization to find optimized paths from a source (s) to a destination (d) node. In the route optimization problem, the path P which exists from the source node s to the destination node d should be optimized. The P consists of edges from E and $P \subsetneq E$. If e_x and e_y are the first and last edges in P, then $e_x = (s, n_x)$ and $e_y = (n_y, d)$, where

Table 1 Properties of the components of the road network

Symbol	Description
Properties of any edge $e_i \in E$	
$l(e_i)$	length of the edge
$T(e_i)$	Traffic on the edge
$R_{pts}(e_i)$	Number of recharging stations on the edge
$N_s(e_i)$	Number of traffic signals on the edge
$W_s(e_i)$	Average waiting delay at any signal on the edge
Properties of any recharging station $r_i \in R$	
$a_e(r_j)$	The edge along which r_j is situated
$s_o(r_j)$	Sequence order of the recharging station from the start of the edge
$loc(r_j)$	Location of r_j in terms of its distance from the start of the edge
$T_{chrg}(r_j)$	Average time required to fully recharge an EV
$T_{wd}(r_j)$	Average waiting delay at any signal on the edge
$R_{cost}(r_j)$	Average waiting delay for the EVs at r_j
Properties of any electric vehicle $EV_k \in EV$	
$S_{max}(EV_k)$	Maximum speed of the EV
$S(EV_k)$	Average speed of the EV
$BCap(EV_k)$	Maximum capacity of the battery of EV_k
$D_{limit}(EV_k)$	The distance which the EV_k can travel without requiring to recharge
Properties associated with any $e_i \in E$, $r_i \in R$, and $EV_k \in EV$	
$D(r_j, EV_k)$	The maximum distance from the start of the edge which the EV_k can travel without requiring to recharge after passing by or recharging at the station r_j
$B_{start}(e_i, EV_k)$	Battery level of the EV_k at the start of e_i
$t_R(e_i, EV_k)$	Total recharging time required by the EV_k to travel on e_i
$C_R(e_i, EV_k)$	Total recharging cost for EV_k to travel on e_i

Table 2 Mapping of the parameters in the MCOP problem to the EV-related problem parameters

MCOP problem parameter	Road Network Parameter	Description
$l(e_i)$	$l(e_i)$	length of the edge
$f_1(e_i)$	$\frac{l(e_i)}{S(EV_k)}$	Average travel time for the EV_k to travel on e_i
$f_2(e_i)$	$W_s(e_i) \times N_s(e_i)$	Average time Delay due to signals on e_i
$f_3(e_i)$	$t_R(e_i, EV_k)$	Recharging time required by the EV_k to travel on e_i
$f_4(e_i)$	$C_R(e_i, EV_k)$	Recharging cost to travel on e_i for EV_k

$n_x, n_y \in V$ and for any two consecutive edges e_i, $e_j \in P$, the ending node of e_i should be equal to the starting node of e_j. The path P should be cycle-free. The proposed MCOP problem consists of one primary cost parameter which is represented as $l(e_i)$ and four additive weight parameters which are: $f_1(e_i)$, $f_2(e_i)$, $f_3(e_i)$, and $f_4(e_i)$. The mapping of the MCOP parameters to the parameters of the EV road network is shown in Table 2. The values of the primary cost and the four additive weight parameters for the complete path (P) can be obtained by taking the sum of the values of the individual edges, i.e., $L(P) = \sum_{e_i \in P} l(e_i)$, $F_k(P) = \sum_{e_i \in P} f_k(e_i)$, for k= 1 to 4. The MCOP problem can be defined as: $Minimize(\sum_{e_i \in P}(l(e_i)))$, such that $\sum_{e_i \in P}(f_k(e_i)) \leq c_k$, for k= 1 to 4 The values of constraints are represented as c_k and the constraints are applied to the path's parameter values. Penalty Function Method[6] is a technique which can transform any MCOP problem into unconstrained optimization problem. The Penalty Function method can be applied to the MCOP problem defined in the above equation which transforms it into an unconstrained optimization problem. The objective function of the unconstrained optimization problem can be represented as $Obj(P)$ and is shown in the following equation: $Obj(P) = Minimize(\sum_{e_i \in P} l(e_i) + \sum_{k=1}^{4} r_k[(max(0, \sum_{e_i \in P} f_k(e_i) - c_k))^n]), n \in Z^+$. The value of n can be 1, 2, or so on. r_k represents the penalty factors. Generally, r_k are assigned values which are greater than the total value of the primary cost parameter of any path from s to d, i.e., $r_k > \sum_{e_i \in P} l(e_i)$, for k= 1 to 4. When the solution satisfies all constraints then the value of the term $[(max(0, \sum_{e_i \in P} f_k(e_i) - c_k))^n]$ is 0, therefore, the $Obj(P)$ value remains lesser than r_k. Based on this observation, it can be inferred that if P is a feasible solution, then $Obj(P) < min(r_1, r_2, r_3, r_4)$. The values of constraints, i.e., c_k should be decided before the optimization can be performed. There are several methods to decide the values of the constraints. But one simple approach is to set the constraints in terms of percentage of the path parameters of the initial solution. An initial solution $P_{initial}$ should be determined before optimization can be applied to it. The constraints should be set on the four additive weight parameters which are represented as c_k. The user should enter the values of percentages which varies from [0,1] for each constraint. Let the user inputs are represented as u_k, then the values of c_k can be determined as: $u_k \in [0,1] \in R, F_k(P_{initial}) = \sum_{e_i \in P_{initial}} f_k(e_i)$, $c_k = F_k - (u_k \times F_k)$, for k= 1 to 4.

```
Input: nodes: $u$, & $v$, G=(V,E), $N_e$= Number of elements in $E$
Output: Q: A path from $u$ to $v$ nodes.
1: $W = random(N_e)$
2: Q= Apply Dijkstra's Algorithm $(u, v)$
3: return Q
```

Fig. 1 Method to find paths: $form\_path(u, v)$.

3. Proposed Algorithm

First a method of building a random path from nodes u to v is shown which will be used by the optimization algorithm to build a path between any two nodes u and v. The algorithm is shown in Fig. 1 and is represented by the function $form\_path(u, v)$. In line 1, W stores N_e randomly generated integers between $[1, 1000]$. The elements in W are used to assign weights to each edge in E, since E also have N_e elements. The Dijkstra's Algorithm finds shortest path w.r.t. the weights assign to edges in line 1. Therefore, every time the function $form\_path(u, v)$ is called a different path is returned.

The SimE Algorithm consists of the following operations: Evaluation, Selection, and Allocation. In Evaluation, the goodness values of all edges in the current solution (P) are evaluated. In the Selection step, the edges which have lower goodness values are selected into the S. In the Allocation step, the edges in the set S will be used to perform random changes in the current solution (P). In the proposed algorithm, a mutation step is added after the allocation which is performed after every N_m iterations. The problem specific goodness, allocation and mutation operations are defined in the following.

3.1 Goodness Function

```
Input: Current Solution $P$, an edge $e_i = (n_a, n_b) \in P$, $d$: destination node
Output: goodness of the edge $e_i = g_{e_i}$
1: $O_P = Obj(P)$ (with $n = 1$)
2: $L = \sum_{e_x \in P} l(e_x) - l(e_i)$
3: for $k$=1 to 4 do
4:     $F_k = \sum_{e_x \in P} f_k(e_x) - f_k(e_i)$
5: end for
6: $O'_P = L + \sum_{k=1}^{4} r_k[max(0, F_k - c_k)]$
7: $g_{e_i} = (1 - \frac{O'_P}{O_P})$
8: return $g_{e_i}$
```

Fig. 2 Goodness Function

The principle used in determining the goodness of edges is that: Lower goodness value is assigned to edges in P which if removed from it causes more reduction in the objective function than the other edges. In that way, the allocation of low goodness edges can bring improvement in the solution. Fig. 2 shows the method proposed for finding the goodness of any edge $e_i \in P$. The inputs are: the current solution (P), edge $e_i = (n_a, n_b) \in P$ whose goodness value need to be determined, and destination node d. The output is the goodness of the edge e_i, which is represented by g_{e_i} and its value can vary between [0,1]. The *Evaluation* step measures the goodness of all edges in P by using the method shown in Fig. 2.

3.2 Allocation Operation

```
Input: Current Solution $P = \{e_a, e_b, .., e_c, e_d\}$, Selection set $S$ & $S \subsetneq P$, $d$: destination
    node
Output: $P$ Solution after the allocation operation
1: $P_{min} = P$
2: for each edge $e_j \in S$ do
3:     $n_x$: starting node of $e_j$
4:     $P' = P(e_a, .., e_i)||form\_path(n_x, d)$, (the ending node of $e_i = n_x$)
5:     if $Obj(P') < Obj(P_{min})$ then
6:         $P_{min} = P'$
7:     end if
8: end for
9: return $P_{min}$
```

Fig. 3 Allocation Operation

The Selection step consists of selecting up-to 70% of the lowest goodness edges from P into S. The allocation operation is applied after the selection operation. The proposed allocation operation is based on the idea that: for each edge (e_j) in S, the current solution (P) is modified from the starting node of e_j to the destination node to form a new path. If the objective function value of any of the new paths is lesser than the current solution (P) then P is updated to that path. Fig. 3 shows that the outer most *for* loop selects a different edge (e_j) from S in each iteration. If the starting node of e_j is n_x, then in line 4, the function $form\_path$ is used to find a path from n_x to d. The path P' is formed by concatenating the portion of P from the first edge in P to e_i with the path found from the $form\_path$ function. The ending node of e_i should be n_x. In line 5, the values of the objective function are compared and the variable P_{min} stores the path which has minimum value of the objective function.

3.3 Mutation

The mutation operation is shown in Fig. 4. It first generates a random number R which have value between $[0, m-1]$, where m is the total number of elements or edges in P. Let us consider that e_R is the edge which exists at the R^{th} position in P and n_R

is the starting node of e_R. The for loop (lines 3-8) in its each iteration calls the function $form\_path$ to find a new path from n_R to d. Then it is concatenated with the portion of P from s to n_R. In lines 5-6, the path having minimum value of the objective function is stored in the variable P_{min}. At the end of theN_m^{th} iteration, the path which has minimum value of the objective function is returned and the current solution is updated in the optimization process.

```
Input: Current Solution $P = \{e_a, e_b, .., e_c, e_d\}$, $N_m \in Z^+$ (e.g. 10), $d$: destination node
Output: $P$ after the mutation operation
1: $m$= number of elements in $P$, $P_{min} = P$
2: $R$= random integer between $[0, m-1]$, $n_R$= starting node of the edge $e_R \in P$
3: for $i = 0; i < N_m; i++$ do
4:     $P' = P\{e_a, .., e_x\} || form\_path(n_R, d)$, (the ending node of $e_x$ should be $n_R$)
5:     if $Obj(P') < Obj(P_{min})$ then
6:         $P_{min} = P'$
7:     end if
8: end for
9: return $P_{min}$
```

Fig. 4 Mutation Operation

4. Simulations

The Java programming language is used to implement the proposed algorithm. The values of parameters used are: N_{mut}= 20, i.e., mutation operation is performed after 20 iterations. The value of N_m in the mutation operation is set to 20, i.e., the outermost for loop in the mutation operation executes for 20 times. The value of B_s, which is the battery level at the source node, is set to 100. The performance of the proposed algorithm is compared with a design of Genetic Algorithm (GA) which was proposed by Salman Yussof et al.[7]. In the GA, the cross-over operation is performed by using the roulette wheel approach. The population size (i.e. N) in the GA is set to 20. The road network of the Dhahran city which has 108 nodes and 432 edges is taken 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 integers between [0,3]. Average delay at any signal (W_s) was set to 3 minutes. A test instance consists of finding an optimum path from a source (s) to a destination (d) node. The source and destination nodes are randomly selected from the graph and $s \neq d$. In each test case, the s and d nodes are usually different from other test cases. The simulations are performed at six different optimization timings which are: 200 milliseconds (msec.), 400 msec., 800 msec., 1 seconds (sec)., and 2 sec. The optimization time refers to the stopping criteria of the optimization loops in the proposed algorithm and the GA. The user input for the constraints, i.e., u_k= 0.40 for k= 1 to 4 in all test instances. The penalty factors r_k, for k=1 to 4 are set to 10,000. Any solution P whose objective function value is lesser than r_k is a feasible solution. In Table 3, the results at each optimization time are inferred from 100 test cases. The results in Table 3 compared the performance of the algorithms in four terms: (A) minimizing the objective function value equal to or more than the other algorithm, (B) finding at-least one feasible solution per test instance, (C) the number of unique feasible solutions returned by the algorithms and (D) the maximum and average number of iterations required by the algorithms to obtain the minimum value of the objective function. The columns "Max." and "Average" are represented as percentage of the total iterations. The "Max." column indicates the maximum iteration value among the 100 test cases and "Average" column contains the average iteration value of the 100 test cases. The results show that performance of the proposed algorithm in terms A, B and C is 28% better than GA, 7% better than GA, and 12.5% lesser than GA, respectively. The proposed algorithm requires lesser iterations to obtain its minimum value for the first time. Table 4 shows average memory required by the algorithms. It shows that the proposed algorithm requires 5.3 MB of memory which is 1.65 times lesser than GA having population size of 10 and 2.3 times lesser than GA having population size of 20. Although the population size in GA is kept more than 2 but in order to compare the memory requirements, the data in Table 4 show that the proposed algorithm requires 1.43 times lesser memory than the GA with population size of 2. As shown in the table that the memory requirement of GA increases with the population size.

5. Conclusion

This work shows that the problem of route optimization in Electric Vehicles (EVs) can be efficiently solved as a Multi Constrained Optimal Path (MCOP) problem. The MCOP problem is then transformed into unconstrained optimization problem by applying the Penalty Function method. A Simulated Evolution (SimE) based algorithm is proposed to perform the unconstrained optimization. The proposed algorithm has problem specific goodness, allocation and mutation operations which enable it to achieve the performance almost equal to the GA but requires lesser memory than the GA. The performance of the proposed algorithm is compared with GA in terms of minimizing the objective function value and finding feasible solutions. The experiments show that its performance is equal to or better than the GA. It also has a memory requirement which is 1.6 times lesser memory than the GA.

Table 3 Comparison of the proposed algorithm with GA using Dhahran city map

Optimization Time	Algorithm	Min Obj (%) (A)	Feasible Solution (%) (B)	# of Paths (C)	Iterations (D)	
					Max	Average
200 msec.	proposed	83	60	157	93.90	7.04
	GA	60	58	152	100	21.49
400 msec.	proposed	87	61	193	96.88	9.00
	GA	62	59	205	100	18.99
800 msec.	proposed	95	69	309	93.81	10.97
	GA	73	64	337	100	15.03
1 sec.	proposed	92	68	252	99.74	9.32
	GA	73	59	267	100	10.28
2 sec.	proposed	93	77	505	88.86	9.05
	GA	83	73	656	90.24	10.88
$\frac{\sum proposed}{\sum GA}$		1.28	1.07	0.87	0.97	0.59

Table 4 Comparison of the average memory requirements

proposed	GA(pop= 20)	GA(pop=10)	GA(pop=2)
5.3 MB	12.33 MB	8.77 MB	7.6 MB
Gain	2.32	1.65	1.43

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) 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.
3) 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.
4) P. Venkataraman, Applied Optimization with MATLAB Programming, John-Wiley & Sons, Inc., 2002.
5) S.M. Sait & H. Youssef, Iterative Computer Algorithms with Applications in Engineering, IEEE Computer Society Press, 1999.
6) Konstantinos E. Parsopoulas and Michael N. Vrahatis, "Particle Swarm Optimization and Intelligence, Advances and Applications", 2010 Information Science Reference (IGI Global).
7) 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