Electric Vehicle Route Planning on GPU using Particle Swarm Optimization and Bellman-Ford

Vincent Roberge PhD, P.Eng. and Mohammed Tarbouchi PhD, P.Eng.
Royal Military College of Canada

RMC CMR — ECE GEGI

Abstract

This poster presents a route-planning software for **electric vehicles** (EV) equipped with regenerative breaking. The proposed algorithm calculates the optimal route based on minimum distance, minimum energy or a combination of both. When evaluating the energy required to travel a road segment, the algorithm accounts for the air resistance, the roll resistance, the gravity force, the motor internal losses and the energy gained through regenerative breaking. A **Particle Swarm Optimizer (PSO)** is used to calculate optimized motor control settings in order to minimize the internal losses. The **Bellman-Ford (BF) routing algorithm** is used to calculate the optimized routes. A **Pareto front** of optimized routes is generated to allow the user to select the preferred route based on distance and energy consumed.

Overall Approach

Step 1
EV model to calculate torque required for various speeds and angles

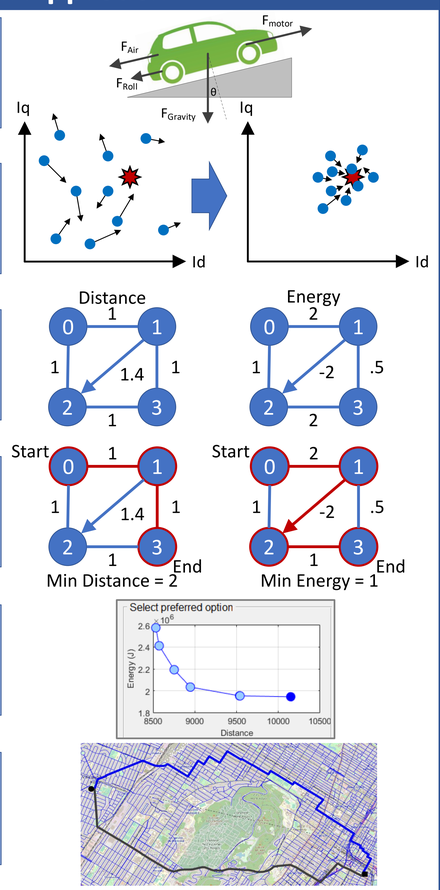
Step 2
PSO to optimize motor parameters and to produce the power matrix

Step 3
Use the power matrix to add weights to the road graph (distance and energy)

Step 4
Run Bellman-Ford to minimize distance and then energy

Step 5
Run Bellman-Ford *N* times to calculate the Pareto front

Step 6
The user selects the preferred route

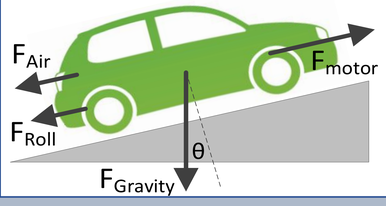


Step 1 - Vehicle Model [1]

- Based on the vehicle mass M, rolling friction coef f_r, drag coef C_D, frontal area A_f and wheel radius r, calculate the torque required T_{req} for:
  - speeds between 1 and 120 km/h (increment of 1 km/h)
  - slope grade between -20° and +20° (increment of 1°)

$$T_{req} = r\left[F_{air} + F_{roll} + F_g \sin(\theta)\right] \quad \textit{where}$$

$$F_{air} = 1/2\,\rho_a C_D A_f v^2$$
$$F_{roll} = M g f_r$$
$$F_g \sin(\theta) = M\, g \sin(\theta)$$



Speed (km/h)

Slope	20	40	60	80	100	120
-20°	T_{00}	T_{01}	T_{02}	T_{03}	T_{04}	T_{05}
0°	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
20°	T_{20}	T_{21}	T_{22}	T_{23}	T_{24}	T_{25}

Step 2 - Induction Motor Optimization using PSO

- For each element in the table generated at the previous step (i.e. given values of T_{req} and ω_{req}), the PSO will find the magnetizing current $(I_d,\ I_q)$ that minimizes $P_{motor\ loss}$ while ensuring T_e is as close as possible to T_{req}.

$$\vec{v}_{t+1} = \omega\vec{v}_t + c_1\vec{r}_1.*\left(\vec{b}_t - \vec{x}_t\right) + c_2\vec{r}_2.*\left(\vec{g}_t - \vec{x}_t\right)$$
$$\vec{x}_{t+1} = \vec{x}_t + \vec{v}_{t+1}$$

- Now that all $P_{motor\ loss}$ are known, the table can be updated using:

$$P_{inst} = v\,T_{req}/r + P_{motor\ loss}$$

Torque T_{req}, Speed ω_{req}, Flux I_d, Flux I_q → Model of Induction Motor [2] → Torque T_e, Power P_{loss}

Slope	20	40	60	80	100	120
-20°	P_{00}	P_{01}	P_{02}	P_{03}	P_{04}	P_{05}
0°	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
20°	P_{20}	P_{21}	P_{22}	P_{23}	P_{24}	P_{25}

Steps 3, 4 - Bellman-Ford Routing Algorithm

- Complexity of Bellman-Ford $\Theta(|V||E|)$ vs Dijkstra $\Theta(|E| + |V|log|V|)$
- Bellman-Ford supports negative weights

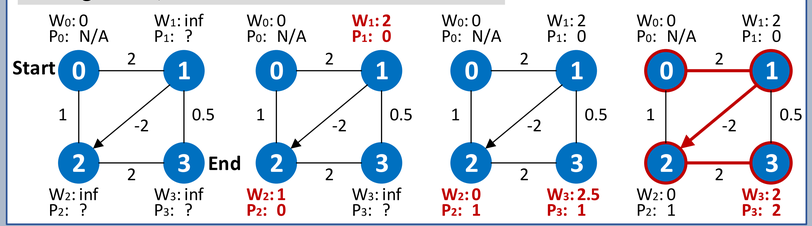
Pseudocode for Bellman-Ford

```
For each node u in graph
  weight(u) = infinity;  previous(u) = -1;
weight(start_node) = 0;  flag = true

While flag == true
  flag = false;
  For each edge (u, v) in graph
    If weight(u) + weight(u, v) < weight(v);
      weight(v) = weight(u) + weight(u, v);
      previous(v) = u;
      flag = true;
```

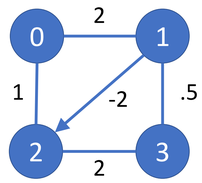
Option 1
For each edge (u, v) **in** graph

Option 2
For each node v **in** graph
  For each node u **going to** v



Parallel Implementation on GPU

- Particle Swarm Optimization in CUDA
  - The PSO is run independently for each pair of $(speed, slope\ angle)$
  - Single kernel where each block runs the entire PSO in shared memory
  - Each thread simulates the movement of a single particle
  - 41 angles, 120 speeds, 128 particles per PSO -> 629,760 threads
- Bellman-Ford in CUDA
  - Use option 1 with 1 thread per edge
    - Can use the COO sparse representation (simpler)
    - Multiple threads may update the same node
    - To avoid race condition, one must combine node's weight and node's previous in a 64-bit C struct
  - Use option 2 with 1 thread per node
    - Must use CSR sparse representation
    - Significant overhead building the CSR
    - No race condition and more efficient



Dense format

$$\begin{bmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & -2 & .5 \\ 1 & 0 & 0 & 2 \\ 0 & .5 & 2 & 0 \end{bmatrix}$$

Sparse COO format

```
Row = [0  1  0  2  1  3  2  3  1]
Col = [1  0  2  0  3  1  3  2  2]
Val = [2  2  1  1 .5 .5  2  2 -2]
```

Sparse CSR format

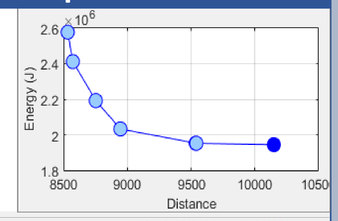
```
Row = [0  2  5  7  9]
Col = [1  2  0  2  3  0  3  1  2]
Val = [2  1  2 -1 .5  1  2 .5  2]
```

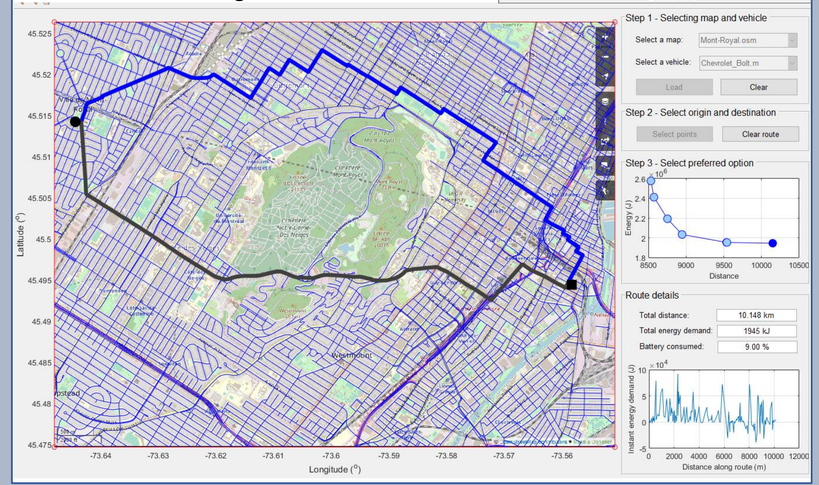
Steps 5, 6 - Multi-Objective Optimization

- Run BF to minimize **distance** (path 1)
- Run BF to minimize **energy** (path 2)
- Calculate the weights of the segments

$$W_{\overline{ij}} = f_a * \frac{L_{\overline{ij}}}{L_{path1}} + f_b * \frac{E_{\overline{ij}}}{E_{path2}}$$

- Vary f_a from 1 to 0 and f_b from 0 to 1
- Run BF each time to get the Pareto front





Results

- C implementation on **Jetson TX2** (4-core CPU, 256-core GPU)
- Uses **OpenMP** for multicore and **CUDA** for GPU
- OpenStreetMap [3], SRTM elevation maps [4], USA road graph [5]

Name	Description	Number of nodes	Number of edges
MR	Mont-Royal in Montréal	12,157	28,632
SF	San Francisco	54,759	122,436
NY	New-York city	264,346	733,846
COL	Colorado state	435,666	1,057,066
NE	Northeast USA	1,524,453	3,897,636

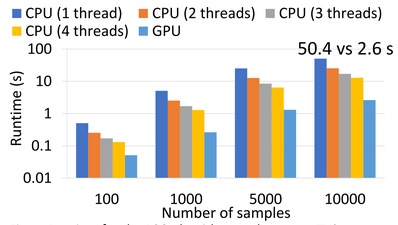


Fig. 1. Runtime for the PSO algorithm on the Jetson TX2

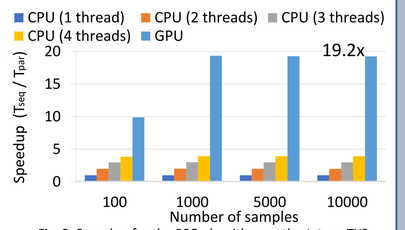


Fig. 2. Speedup for the PSO algorithm on the Jetson TX2

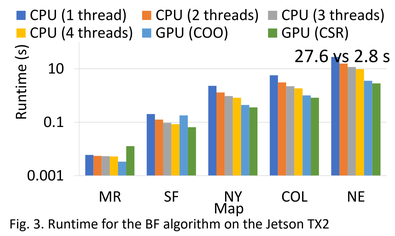


Fig. 3. Runtime for the BF algorithm on the Jetson TX2

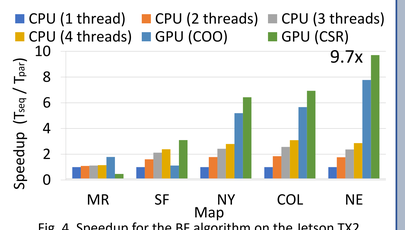


Fig. 4. Speedup for the BF algorithm on the Jetson TX2

References

1. A. Schambers, M. Eavis-O'Quinn, V. Roberge, and M. Tarbouchi, "Route Planning for Electric Vehicle Efficiency Using the Bellman-Ford Algorithm on an Embedded GPU," In Proc. *4th International Conference on Optimization and Applications*, Mohammedia, Morocco, 2018.
2. Z. Rouabah, F. Zidani and B. Abdelhadi, "Efficiency optimization of induction motor drive using fuzzy logic and genetic algorithms," 2008 IEEE International Symposium on Industrial Electronics, Cambridge, 2008, pp. 737-742.
3. "OpenStreetMap." [Online] http://www.openstreetmap.org. [Accessed: 29-Nov-2018]
4. "NASA Shuttle Radar Topography Mission Global 1 arc second V003." [Online] https://earthdata.nasa.gov/. [Accessed: 29-Nov-2018]
5. "9th DIMACS Implementation Challenge: Shortest Paths." [Online] www.diag.uniroma1.it/challenge9/. [Accessed: 29-Nov-2018]