Chapter 5
Energy-Efficient Route Navigation (Eco-Routing)

Eco-routing methods are the strategies and tools aimed at minimizing a vehicle's energy consumption by route selection. Given some origin and destination, which are typically chosen by the driver or user, eco-routing plans an energy-minimal route.

The set of possible routes from origin to destination constitutes a graph, where nodes and links represent junctions and roads. In the next Sect. 5.1 we will define a weighting function, which associates each link of the graph with a weight. In conventional routing graphs, the weight associated with each arc is either the length of the arc or its travel time. In the eco-routing framework, each link of the graph is assigned a weight that represents the travel energy expenditure. Differently from length or travel time weights, energy weights can be negative when EVs or HEVs with regenerative braking are concerned. Then in Sect. 5.2 we present how the eco-routing algorithms can be used to predict the maximal driving range of a vehicle. Finally, Sect. 5.3 will discuss some practical implementation issues.

5.1 Eco-Routing as a Shortest-Path Problem

The energy-minimal navigation problem introduced above is treated by formulating an *eco-routing shortest-path problem* (ER-SPP) in Sect. 5.1.1, while Sect. 5.1.2 presents various techniques to solve such a problem.

5.1.1 Problem Formulation

The eco-routing as a shortest path problem can be stated in the following way. For a given directed graph $\mathcal{G} = (V, A)$, where V is the set of nodes n_i, and A is the set of

A. Sciarretta and A. Vahidi, *Energy-Efficient Driving of Road Vehicles*,
Lecture Notes in Intelligent Transportation and Infrastructure,
https://doi.org/10.1007/978-3-030-24127-8_5

links (or edges) e_k connecting these nodes, find the path $\mathbf{p} = (e_1, e_2, \ldots) \subset \mathcal{P}$, where $\mathcal{P}$ is the set of all simple[1] paths in $\mathcal{G}$, such as to minimize the objective function

$$J(\mathbf{p}) \triangleq \sum_{e_k \in \mathbf{p}} w_k(t, b_k(t)) \ , \tag{5.1}$$

where w_k is the weight attributed to the link k, possibly as a function of time and of additional *decision variables* b_k. Minimization of (5.1) is subject to (i) initial and terminal conditions (*single-source* shortest path)

$$n_1 = n_O, \quad n_{(|\mathbf{p}|)} = n_D \ , \tag{5.2}$$

where n_O and n_D are the origin and the destination sought, respectively, and $|\cdot|$ denotes cardinality, (ii) first-order dynamic constraints on the *state vector* x_k for each node n_k belonging to $\mathbf{p}$,

$$x_{k+1} = x_k + f_k(t, w_k(t, b_k(t)), b_k(t)) \quad x_1 = x(n_O) \ , \tag{5.3}$$

(iii) algebraic constraints on the state,

$$g_i(x_k) \leq 0, \quad i = 1, \ldots, \ell \ , \tag{5.4}$$

terminal inequality constraints over (iv) the state,

$$h(x(n_D)) \leq 0 \ , \tag{5.5}$$

and (v) possibly, over a certain *resource*,

$$R(\mathbf{p}) \triangleq \sum_{e_k \in \mathbf{p}} r_k(t, b_k(t)) \leq R_f \ , \tag{5.6}$$

where r_k is the resource consumption over link k.

How the quantities V, A, w_k, b_k, x_k, f, g, h, and r_k, are particularized for our ER-SPP will be discussed in the following sections. Although costs, state variations, and resource consumptions might change with time, we will consider only the *time-independent* SPP. Similarly, although the situation in the road network is continually evolving and predicting its state in future is arguably difficult, we shall not consider stochastic SPP.

5.1.1.1 Graph

The most intuitive choice to set $\mathcal{G}$ for a given road network is such that nodes represent road intersections and links represent roads. However, the use of this *primal*

[1] A simple path is a path without cycles: its edges are distinct.

graph presents some major difficulties. In particular, consider the case of an intersection with two or more incoming roads and one outgoing road i. The weight of the latter clearly depends on which upcoming link the vehicle comes from because each movement to enter i (e.g., left-turn, right-turn, through) might be accompanied by a different speed transition. In fact, the synthetic profile $v_i(\tau)$ introduced in Sect. 4.4 is not unique since it depends on the upstream average speed V_{i-1} that is not unique in this case.

This difficulty is resolved by modeling the road network as an *adjoint graph*, which is defined as follows. The adjoint graph $\mathcal{G} = (A, A^*)$ of a directed graph $\mathcal{G}' = (V, A)$ has a node for each link in $\mathcal{G}'$, and a link connecting two nodes if the corresponding two links in $\mathcal{G}'$ share a common node.

In other words, each link $i \in A$ of $\mathcal{G}'$ becomes a node of $\mathcal{G}$, and each link $k \in A^*$ of $\mathcal{G}$ encompasses two generic links $i-1, i \in A$ of $\mathcal{G}'$. Thus, the adjoint graph allows to correctly assign unique weights to all the possible maneuvers in the original graph $\mathcal{G}$. Figure 5.1 illustrates this concept. Useful properties of adjoint graphs allow to compute the number of links of $\mathcal{G}$, and therefore its size, based on the connectivity

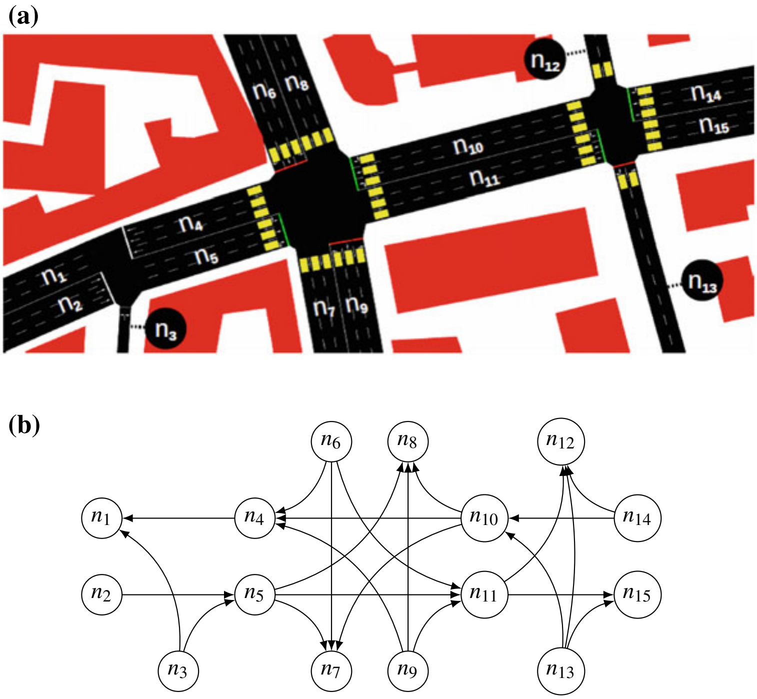


Fig. 5.1 Relation between a road network (**a**) and the corresponding road network graph $\mathcal{G}$ (**b**) [2]. The nodes are shown as circles, the edges as arrows. There are fifteen nodes n_1 to n_{15} representing the roads in the road network. The edges (arrows) spanning from nodes indicate to which nodes (roads) can the vehicle turn to at the downstream intersection

properties of $\mathcal{G}'$ [1]. Note that the origin and destination nodes of (5.2) represent in $\mathcal{G}$ the initial and target links (i_O, i_D) of the primal graph.

In this context, battery recharge stations (of either type, swapping, full/partial refill, continuous wireless charging, etc.) can be attributed to some nodes of the primal graph, thus to a subset $A_c^* \subset A^*$ in the adjoint graph.

5.1.1.2 Objective Function

In standard navigation systems, the objective function $J(\mathbf{p})$ is chosen such as to represent the trip length or the travel time. In eco-routing, the focus is on energy saving and therefore the weight assigned to each link of the graph represents the associated (tank) energy consumption E_T.

Using the synthetic energy consumption defined on the links of the primal graph $\mathcal{G}'$, we can now attribute energy weights to each link $e_k \in A^*$. Reminding that the link k of the adjoint graph encompasses the road segments $i-1$ and i of the primal road network, the most natural choice is to attribute to e_k the energy consumption of the downstream road segment (i) that already accounts for the transition from road segment $i-1$, see Sect. 4.4.3. In other words, we set

$$w_k = E_{T,i} \ , \tag{5.7}$$

where $E_{T,i}$ can be evaluated with (4.38) for ICEVs, (4.41) for EVs, and with (4.46) for HEVs. Note that in the latter case, the fuel energy consumption depends on the electric energy consumption, which thus plays the role of a decision variable b_k in (5.1).

The shortest path (SP) and the fastest path (FP) routing can be defined analogously such that the SP routing minimizes the distance and the FP routing minimizes the travel time. In these cases, the link weights w_k are represented by the length of the outgoing road segment or its travel time, respectively,

$$w_k^{(SP)} = \ell_i \ , \quad w_k^{(FP)} = \tau_i \ . \tag{5.8}$$

This travel time can be evaluated with (4.31).

5.1.1.3 Decision Variables

Decision variables b add degrees of freedom to the eco-routing problem besides the choice of the link sequence in the path.

In HEVs, the additional decision variables are the electricity consumption on links k,

$$b_k^{(HEV)} = E_{b,i} \ , \tag{5.9}$$

whereas the cost is represented by the minimal fuel consumption. Note that the b_k's can be negative in this case.

When recharge facilities for electrified vehicles (a scenario denoted here as xEVR) are taken into account, that is, the path is not supposed to be covered with one single battery recharge, the additional decision variable is the quantity of electricity refilled at links where a recharge facility is available,

$$b_k^{(xEVR)} = \Delta E_{c,i}, \quad e_k \in A_c^* \ . \tag{5.10}$$

In both aforementioned cases, the kth graph weight w_k can take on infinitely-many values depending on the decision variable over the graph link. The problem of finding the optimal b_k for each link of the graph while searching the optimal route from an origin to a destination has been addressed in the literature [3]. However, the proposed solution strategies are often impractical due to large computational cost. A practical approach therefore consists of augmenting the adjoint graph $\mathcal{G} = (A, A^*)$ by creating as many *copies* of each link of the graph as the number of pre-defined possible values of b_k. Using the discretized values $b_k^{(j)} \in [b_{k,min}, b_{k,max}]$, $j = 1, \ldots, N_b$, let us define a b-augmented graph $\mathcal{G}_B = (A, A_B^*)$ with link copies $k^{(j)} \in A_B^*$, and a new cost function

$$w_k^{(j)} \triangleq w_k : b_k = b_k^{(j)}, \quad j = 1, \ldots, N_b \ , \tag{5.11}$$

that represents the cost to perform the maneuver $k^{(j)} \in A_B^*$ for a given decision variable $b_k^{(j)}$. The number of copies, or decision variable levels, N_b is a design parameter, in a trade-off between discretization accuracy and computational burden. Furthermore, the variation range of $b_k^{(j)}$ depends on the physical properties of the maneuver $k \in A^*$.

Another possibility to treat the problem of eco-routing with additional decision variables is to make copies of the links w.r.t. discretized values of the state (see below) at the end of the link [3, 4]. However, augmenting the graph in terms of decision variables offers a much higher precision as compared to these alternative graph expansions, while the best choice in terms of algorithmic complexity is a matter of debate.

5.1.1.4 State Dynamics and Constraints

In both scenarios where states are relevant, that is, HEVs and xEVs with battery recharge, the role of state variable is played by the electrical energy stored in the battery, $x = \{\varepsilon_b\}$. Given (5.7) and (5.9)–(5.10), the state dynamics (5.3) reads

$$f_k^{(HEV)} = -E_{b,i} = -b_k \tag{5.12}$$

for HEVs and

$$f_k^{(xEVR)} = -E_{b,i} = -w_k + b_k \tag{5.13}$$

for xEVs with battery recharge, with $x(n_O) = \varepsilon_{b,0}$ in both cases. These dynamics are further subject to the physical limits of the battery, that is, the local constraints (5.4) that are to be applied at each link and for partial paths,

$$0 \leq \varepsilon_b \leq \varepsilon_{b,max} \ , \tag{5.14}$$

with $\varepsilon_{b,max} \triangleq Q_b V_{b0}$ being the battery maximum capacity, while it has been assumed for simplicity that the battery can be completely depleted.[2]

Depending on the type of HEV, the battery energy at the end of the trip should match a prescribed value. For charge-sustaining HEVs, the final SoC should match the initial value, thus $\varepsilon_b(n_D) - \varepsilon_{b,0} = 0$ in (5.5). For plug-in HEVs, if a recharge is possible at the destination, the battery should be depleted. Thus the desired value for $\varepsilon_b(n_D)$ is usually close to zero, that is, a battery fully discharged.

5.1.1.5 Resource Constraints

Although energy efficiency is the main objective in ER-SPP, neglecting travel time might reduce the appeal of the selected route and the drivers' compliance to the routing suggestion. Hence, it is important to optimize energy consumption while providing the opportunity to the drivers to define their preferred trade-off with travel time. This trade-off can be achieved by selecting the resource consumption r_k in (5.6) as to represent the travel time of the link k,

$$r_k = \tau_i \ , \tag{5.15}$$

where τ_i is the travel time of the downstream link of the primal graph. The term R_f in (5.6) thus plays the role of the maximum time allowed for the route.

When battery recharge is taken into account, the travel time must include the waiting time at stations and the recharging time. Therefore, the resource consumptions become a function of the electricity recharged, thus of the decision variable (5.10),

$$r_k^{(xEVR)} = \tau_i + \Delta\tau_{c,i}(\Delta E_{c,i}, \varepsilon_{b,k}) \ . \tag{5.16}$$

For the charging function $\Delta\tau_{c,i}(\Delta E_{c,i}, \varepsilon_{b,k})$, the models introduced in Sect. 4.1.4 can be used.

*
* *

The presence of terminal state and resource constraints makes the eco-routing problem a resource-constrained SPP (RCSPP), which is known to be NP-hard. In order to overcome this complexity, which makes the problem impractical for end-

[2] In practice, that is not true, and a practical SoC window must be considered instead.

user applications and driving assistance, a more tractable problem formulation can be introduced under the form of a *Multi-Objective Optimization* Problem (MOOP), where resources and state constraints are transformed into additional objective functions to be minimized alongside with $J(\mathbf{p})$.

Solving a MOOP requires an external decision making process to give relative preference to the various objectives. In a-posteriori methods, preference is given, and a decision is made, once a representative set of optimal solutions is found. In contrast, a-priori methods require that preference information is available before searching for optimal solutions. A standard a-priori method is scalarization, that is, combining the various objectives into a single objective function. Several approaches to scalarization exist, depending on the form of the single objective function. In particular, the standard approach of the weighted-sum scalarization [5] casts the MOOP as a single-objective problem by appending the terminal constraints as additional terms of the objective function,

$$(1-\lambda_1-\lambda_2)\sum_{e_k\in\mathbf{p}} w_k + \lambda_1\sum_{e_k\in\mathbf{p}} r_k + \lambda_2\sum_{e_k\in\mathbf{p}} f_k\ , \tag{5.17}$$

where the first term of (5.17) is the original objective function, the second is the terminal value of the resource that is constrained by (5.6), and the third is the terminal value of the state constrained by (5.5), where for simplicity a hard constraint has been assumed. The optimization weights $\lambda \in [0, 1]$ define the trade-off between the objectives, and should be sought such that the terminal constraints for the optimal path $\mathbf{p}$ are met, see below Sect. 5.1.2.3.

The presence of local state constraints (5.4) could be treated by making several copies of each link for several state levels in a discretized set and thus directly enforce the constraints. However, this problem makes the graph complexity grow exponentially. Therefore, a common approach to deal with these constraints is to relax them in the problem formulation, and verify them a posteriori on the optimal path [6]. Such a path is feasible if $\forall e_k \in \mathbf{p}$, x_k fulfills (5.4). If it is not, it is necessary to select the next-best path, which in turn requires a routing algorithm that is capable to sort paths according to the objective function chosen (see Sect. 5.3).

5.1.2 Routing Algorithms

Graphs modeling road networks are *directed* and *cyclic* by nature, and, due to limited connectivity and the presence of one-way roads, the adjacency matrix representing road networks graphs tends to be highly sparse. Furthermore, the consideration of electric vehicles and energy recuperation phenomena implies that the graph links may be weighted with *negative costs*. In theory, negative weights on a cyclic graph may lead to the criticality of negative cycles: there might be cycles in the road network graph whose sum of costs is negative. Routing in such graphs is not possible, as the vehicle can gain any amount of energy simply by running along these cycles sufficiently many times. However, this issue does not make physical sense in our framework, provided that the modeling approach is correct.

In the next sections two common shortest-path algorithms are presented, namely Dijkstra's and Bellman-Ford (BF). Then, algorithms are presented that can be used to effectively find the Pareto front for the bi-objective optimization (5.17).

5.1.2.1 Dijkstra's Algorithm

Dijkstra's algorithm, conceived by Edsger Dijkstra in the 1950s [7], is perhaps the most commonly used algorithm in routing context. A basic implementation is described in Algorithm 1. The procedure starts marking all nodes of the graph unvisited and assigning a cost, denoted with J, equal to zero to the origin node, infinity to the others. Then an iteration over the unvisited nodes is launched, starting from the origin node. For the current node, all of its unvisited neighbors are considered and their link costs through the current node are evaluated. This value is compared to the current assigned cost and the smaller one is assigned. When all of the unvisited neighbors of the current node are considered, the current node is marked as visited (it will never be checked again). If the destination node has been marked visited, then the procedure has finished. Otherwise, the unvisited node with the smallest cost is selected as the new current node and a new iteration begins. Once this procedure is completed, a reverse iteration allows then reconstructing the optimal path.

The computational complexity of Dijkstra's algorithm is $O(||A||^2)$, where $||\cdot||$ denotes the set cardinality, which makes its use particularly attractive. However, Dijkstra's algorithm requires that the costs are non-negative to provide an optimal solution. Therefore, it cannot be generally used for the eco-routing problem with EVs or HEVs.

A widely used extension of Dijkstra's algorithm is the A* algorithm. This search method differs from Dijkstra's essentially because it uses a heuristic function to evaluate the current node's neighbors in addition to the "real" cost w. This heuristic is an estimate of the remaining cost from the neighbor to the target node.

5.1.2.2 Bellman-Ford Algorithm

The well-known Bellman-Ford (BF) algorithm [8] can be used to route on graphs with negative costs. Similarly to Dijkstra's algorithm, BF can be considered fast since their runtime is polynomially bounded. However, while Dijkstra's method visits only those nodes that can potentially be on the shortest path, Bellman-Ford operates on every node in the graph. Hence the computational effort associated with Dijkstra's method is dominated by trip properties while the road network size dominates the computational effort associated with the Bellman-Ford algorithm. In particular, its computational complexity is $O(||A|| \cdot ||A^*||)$.

For large road networks, such a computational complexity leads to significantly high computation time, which is not suitable for user-oriented applications and real-time use. However, early termination conditions [9] can be introduced in order to stop the search when an iteration of the algorithm main loop ends without making any link relaxation. When this happens, it means that the algorithm has already found all the shortest paths from the origin and the following iterations would not modify them. This does not improve the worst-case performance of the algorithm, but it performs extremely well on real road networks [10]. In practice, the Algorithm 2 drastically reduces the computation time.

Algorithm 1 Dijkstra algorithm

```
Require: G, w_k, n_O, n_D
Ensure: p, J
                                              ▷ Initialization
  p ← ∅,  J ← ∞,  pred(n_O) ← 0,  J(n_O) ← 0
  Q ← A
  while Q ≠ ∅ do
                                              ▷ Find current node
     J_opt ← ∞
     for u ∈ Q do
        if J(u) < J_opt then
           J_opt ← J(u)
           u_opt ← u
        end if
     end for
     u ← u_opt
                                              ▷ Remove u from Q
     Q ← Q \ u_opt
                                              ▷ Termination condition
     if u_opt = n_D then
        return
     end if
                                              ▷ Evaluate neighbors
     for v ∈ neighbors(u) do
        if J(u) + w(u, v) < J(v) then
           J(v) ← J(u) + w(u, v)
           pred(v) ← u
        end if
     end for
  end while
                                              ▷ Reverse iteration
  u ← n_D,  p ← ∅
  repeat
     p ← u ∪ p
     u ← pred(u)
  until u = n_O
```

Algorithm 2 Bellman-Ford algorithm

Require: $\mathcal{G}, w_k, n_O, n_D$
Ensure: $\mathbf{p}$, J

▷ Initialization

$\mathbf{p} \leftarrow \emptyset, \quad J \leftarrow \infty, \quad \text{pred}(n_O) \leftarrow 0, \quad J(n_O) \leftarrow 0$

▷ Cycle

```
for i ∈ {1, ..., ||A||} do
    optimal ← True
    for arc ∈ {1, ..., ||A*||} do
        u ← tail(arc)
        v ← head(arc)
        if J(v) > J(u) + w(arc) then
            J(v) ← J(u) + w(arc)
            pred(v) ← u
            optimal ← False
        end if
    end for
    if optimal then
        return
    end if
end for
```

▷ Reverse iteration

```
u ← n_D,   p ← ∅
repeat
    p ← u ∪ p
    u ← pred(u)
until u = n_O
```

5.1.2.3 Bi-objective Optimization

We consider a particular case of the MOOP defined in Sect. 5.1.1 where only a resource constraint is present (bi-objective optimization), weighted with $\lambda_1 = 1 - \lambda$, $\lambda_2 = 0$.[3] The goal is to find the paths that do not allow to improve one component of the objective function without deteriorating the other one (*non-dominated* solutions). The corresponding set in the objective space $\{J(\mathbf{p}), R(\mathbf{p})\}$ is generally called *Pareto front*, see Fig. 5.2. Given the nature of the eco-routing problem, the Pareto front is intrinsically made of a discrete set of solutions. Spanning the entire front thus provides the exhaustive list of possible solutions compromising between the selected cost (energy) and resource consumption (often, travel time). On the one hand, that allows a decision maker to arbitrarily select a desired trade-off. On the other hand, the particular non-dominated solution that minimizes the original objective function J without violating the resource constraint (5.6) can be easily identified.

The combinatorial nature of such problems makes the search of all Pareto solutions a very time-consuming task. Thus the practical goal is often to find the most diverse set of solutions, i.e., solutions that are sufficiently varied in the true Pareto set. An

[3]The same approach could be of course applied to situations where the state constraints but not the resource constraints are relevant, just by replacing R with h.

Algorithm 3 Scalarization bi-objective optimization

Require: w_k, r_k, n_O, n_D
Ensure: S
  $S \leftarrow \emptyset$
  ▷ Initialize binary search
  $\lambda \leftarrow 0$
  $\mathbf{p}^1 \leftarrow \text{BFalgorithm}\,(\mathcal{G}, \lambda w_k + (1-\lambda) r_k, n_O, n_D)$
  $S \leftarrow S \cup \left\{J(\mathbf{p}^1), R(\mathbf{p}^1)\right\}$
  $\lambda \leftarrow 1$
  $\mathbf{p}^2 \leftarrow \text{BFalgorithm}\,(\mathcal{G}, \lambda w_k + (1-\lambda) r_k, n_O, n_D)$
  $S \leftarrow S \cup \left\{J(\mathbf{p}^2), R(\mathbf{p}^2)\right\}$
  ▷ Compute S
  $S \leftarrow \text{solveRecursion}\left(J(\mathbf{p}^2), R(\mathbf{p}^2), J(\mathbf{p}^1), R(\mathbf{p}^1), S, 1, 0\right)$
  ▷ Recursive function
  function SOLVERECURSION($z_1^l, z_2^l, z_1^r, z_2^r, S, \lambda_l, \lambda_r$)
    $\lambda \leftarrow (\lambda_l + \lambda_r)/2$
    $\mathbf{p} \leftarrow \text{BFalgorithm}\,(\mathcal{G}, \lambda w_k + (1-\lambda) r_k, n_O, n_D)$
    $z_1 \leftarrow J(\mathbf{p})$
    $z_2 \leftarrow R(\mathbf{p})$
    if $\{z_1, z_2\} \notin S$ **then**
      $S \leftarrow S \cup \{z_1, z_2\}$
      if $|\lambda - \lambda_r| \geq \gamma_\lambda$ AND $|(\lambda z_1 + (1-\lambda) z_2) - \left(\lambda z_1^r + (1-\lambda) z_2^r\right)| \geq \gamma_d$ **then**
        $S \leftarrow \text{solveRecursion}\left(z_1, z_2, z_1^r, z_2^r, S, \lambda, \lambda_r\right)$
      end if
      if $|\lambda - \lambda_l| \geq \gamma_\lambda$ AND $|(\lambda z_1 + (1-\lambda) z_2) - \left(\lambda z_1^l + (1-\lambda) z_2^l\right)| \geq \gamma_d$ **then**
        $S \leftarrow \text{solveRecursion}\left(z_1^l, z_2^l, z_1, z_2, S, \lambda_l, \lambda\right)$
      end if
    else
      if $\{z_1, z_2\} = \left\{z_1^l, z_2^l\right\}$ **then**
        if $|\lambda - \lambda_r| \geq \gamma_\lambda$ **then**
          $S \leftarrow \text{solveRecursion}\left(z_1, z_2, z_1^r, z_2^r, S, \lambda, \lambda_r\right)$
        end if
      else
        if $|\lambda - \lambda_l| \geq \gamma_\lambda$ **then**
          $S \leftarrow \text{solveRecursion}\left(z_1^l, z_2^l, z_1, z_2, S, \lambda_l, \lambda\right)$
        end if
      end if
    end if
    return S
  end function

example of efficient dichotomic algorithm for the search of the non-dominated solutions is Aneja's [11], which was proved to find all the non-dominated solutions within a finite number of iterations. In [10], a new binary search algorithm (see Algorithm 3) has been proposed to significantly reduce the computation time, while computing a representative sub-set of non-dominated solutions.[4] The main difference between

[4]In theory, the algorithm only finds a subset of extreme supported non-dominated solutions, that is, a subset of those non-dominated solutions that lie on vertices of the convex hull of the Pareto front, see Fig. 5.2.

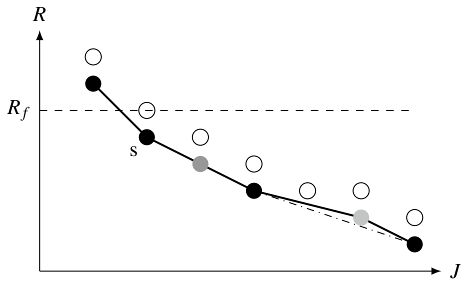


Fig. 5.2 Objective space of a bi-objective optimization problem. Solid line: Pareto front. Dash-dot line: convex hull. Black circles: extreme supported non-dominated solutions. Gray circle: non-extreme supported non-dominated solution. Light gray circle: non-supported non-dominated solution. White circles: dominated solutions. Solution s is the one that minimizes energy consumption without violating the constraint on resource consumption

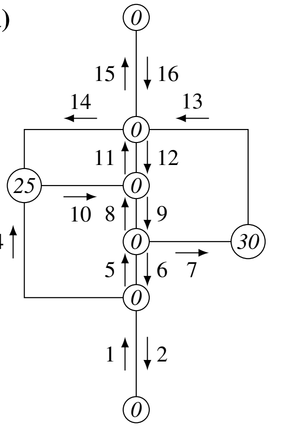
the two algorithms consists of the space where the solutions are searched: Aneja's algorithm recursively explores the objective space selecting the decision weight λ based on the known non-dominated solutions, the modified algorithm explores the decision space performing a recursive binary search of the decision weight. The latter algorithm presents a standard initialization phase in which the two single-objective solutions for $\lambda = 0$ (minimizing the resource consumption, e.g., the fastest route) and $\lambda = 1$ (the energy-optimal route) are computed. The parameters γ_λ and γ_d in Algorithm 3 are set in such a way to find the right trade-off between number of algorithm iterations and number of solutions found.

5.1.3 Numerical Solutions

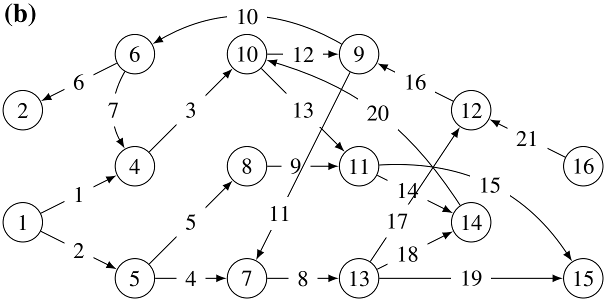
This section presents some results obtained using the modified BF algorithm (Algorithm 2) for a simple route graph. The primal graph is shown in Fig. 5.3a, which has been obtained from the road network of Fig. 4.2 by further considering separate directions of the original database links as different segments. Also indicated in the figure are average speeds for each route segment and altitude at each node. The adjoint graph is shown in Fig. 5.3b, with energy cost and travel time (resource consumption) indicated at each link. In this illustration, wheel energy is considered as the energy cost, which is independent from the particular powertrain used. No recharge facilities are considered. Extra energy and travel time are assigned to links, according to the method in Sect. 4.4, depending on the value of the parameter ζ defined in (4.27), that is randomly assigned to the links to represent traffic signals ($\zeta = 1$ is for free flow, $\zeta = 0$ is for a stop).

Figure 5.4 shows the eco-route and the space of energy-time solutions for a given sequence of stop signals. Note that, for such a small graph, the distinct routes with no loops are easily enumerated and the Pareto front explicitly calculated.

(a)



(b)



(c)

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\bar{v}_i$ (km/h)	50	45	–	40	50	45	35	50	45	50	50	45	45	35	50	45
ℓ_i (dam)	50	50	–	40	45	45	40	30	30	80	50	50	45	30	50	50

(d)

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
ζ_k	1	1	1	1	0	1	0	1	0	1	0	1	1	1	1	0	0	0	0	0	1
τ_k (s)	49	34	29	37	27	38	36	37	29	43	44	23	43	33	36	32	49	43	47	38	40
$E_{W,k}$ (Wh)	66	32	39	71	84	30	106	25	86	23	104	−1	6	62	33	73	33	53	39	79	31

Fig. 5.3 Simple route graph, with directions on route segments numbered from 1 to 16 and altitude values on nodes (**a**). Corresponding adjoint graph with nodes numbered from 1 to 16 and links numbered from 1 to 21 (**b**). Values of average speed and length for each road segment/graph node (**c**). Values of parameter γ, travel time, and wheel energy for each graph link, for a vehicle with $C_0 = 161.7$, $C_2 = 0.409$, $m_v = 1100$ kg (**d**)

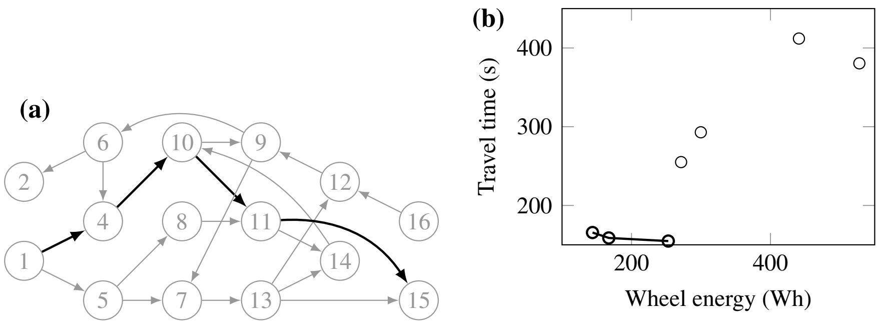


Fig. 5.4 Minimal-energy path (eco-route) in red for the graph of Fig. 5.3 and $n_O = 1$, $n_D = 15$ (**a**). Results for the the seven possible solutions with no loops (1-5-7-13-15, 1-5-8-11-15, 1-4-10-11-15, 1-4-10-9-7-13-15, 1-5-7-13-14-10-11-15, 1-5-8-11-14-10-9-7-13-15, 1-5-7-13-12-9-6-4-10-11-15), in terms of wheel energy and travel time; Pareto set (non-dominated solutions) in blue (**b**)

5.2 Energy-Optimal Driving Range Estimation

Vehicles' owners often wonder how far they can drive with the on-board stored energy. Such a question is particularly relevant for electric vehicles, where the limited amount of energy stored in the battery is often perceived as a strong limitation to the penetration of this powertrain technology ("range anxiety").

Driving range is often estimated in terms of distance making assumptions on the average energy consumption per kilometer. Such estimations are made either before the trip based on worst-case average energy consumption assumptions, or during the trip based on measured consumption [12]. Because the range output to the user is typically conservative to avoid running out of charge, imprecise range estimates may further increase range anxiety.

In order to be insightful and effective, a driving range estimation strategy should be accurate by specifically considering the characteristics of the vehicle and the road transportation network. Also, when predicting driving range, it is of paramount importance to clearly state the route types allowing the driver to reach the locations within the driving range. For instance, routes favoring lower travel time could be more energy-expensive than other types of routes, therefore the route choice directly affects the size of the driving range. This aspect, although very important, is often neglected [13]. In only few works [14, 15], the authors realize the importance of choosing eco-routes to determine the driving range, however the simplified energy consumption and road network model do not allow for a satisfactory precision in urban and suburban environments.

In the next sections, it will be shown how the eco-routing methods of Sect. 5.1 can be used to provide an energy-optimal driving range, by predicting the optimal energy consumption to reach all the possible destinations within the driving range. This allows to relax the typical unrealistic assumptions of a worst-case average energy consumption and to give a clearer insight into the energy characteristics of the road network [16].

5.2.1 Problem Formulation

The energy-optimal driving range to find may be defined as

$$\text{range}(n_O) \triangleq \left\{\Delta \subseteq A : \forall n \in \Delta,\ E_T(n|n_O) \leq E_T^*\right\}, \tag{5.18}$$

where n_O is the origin node, A is the set of nodes in the graph, Δ is the subset of nodes within the energy-optimal driving range, $E_T(n|n_O)$ is the predicted energy consumption from the origin node to node n, and E_T^* is a desired (tank) energy consumption. Typically, for EVs, this quantity is the electric energy stored in the battery at the origin node.

5.2.2 Solution Method

The single-source shortest-path algorithms used for eco-routing, typically Dijkstra's or Bellman-Ford algorithms, have the property of returning within the same execution not only the shortest path from origin to destination, denoted in Sect. 5.1 with **p**, but also the optimal costs to reach all the nodes in the graph, denoted, e.g., in Algorithms 1 and 2, with J. Such a characteristic may be exploited to identify all the destinations reachable with a certain desired energy consumption.

In practice, it suffices to replace $E_T(n|n_O)$ in (5.18) with $J(n)$, the minimal cost function to reach node n. Note that the driving range obtained by means of a single-source SP algorithm is optimal in the sense that each node within the range is reachable by following the shortest path from the origin to that node. In other words, since the routing graph is weighted with energy costs (for a choice of $\lambda = 1$), the nodes in the energy-optimal driving range are reachable via eco-routes.

This approach enables a high level of precision in the driving range calculation. As opposed to the techniques providing only the polygonal curve delimiting the driving range, this strategy allows to analyze the energy consumption characteristics of the region within the driving range and, more importantly, to determine whether such a region is simply connected. If the properties of simple connectedness do not hold, then some nodes within the driving range are unreachable even by means of an eco-route.

The strategy may be easily extended to compute a round-trip driving range, which may be of interest especially in the case of electric vehicles. The round-trip driving range can provide an insight into the reachable destinations that also allow to return to the departure point within a desired energy consumption threshold. In order to do so, the single-destination shortest path problem is simply solved on the same routing

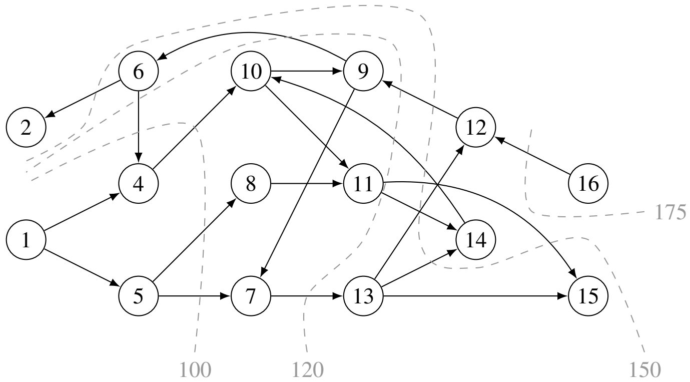


Fig. 5.5 Energy range evaluated for the problem of Fig. 5.3 with $n_O = 1$. Gray dashed lines are the contour lines at the wheel energy consumption indicated (Wh)

graph with reversed links orientation. The Bellman-Ford algorithm is run on such a graph using again the origin (i.e. the destination of the round-trip itinerary) as the starting node.

5.2.3 Numerical Solutions

An example result of energy-optimal driving range estimation is presented in Fig. 5.5. The graph considered is the simple road network of Fig. 5.3. Taking $n_O = 1$ as the origin node, the figure shows some contour lines of equal energy consumption.

More extensive experiments [16] have demonstrated that the methods of this section allow to correctly capture important characteristics of the energy driving range for EVs and HEVs, such that:

- the driving range may be asymmetric about the origin;
- the driving range region may be not simply connected;
- auxiliary power demand may have a significant impact on the driving range;
- the round-trip driving range may be able to restore symmetry about the origin.

A detailed case study is presented in Sect. 9.5.

5.3 Practical Implementation

In principle, the eco-routing algorithm is run at the demand of the user and should start with an update of the energy weights and resource consumptions (w_k and r_k in this chapter) for each link of the road network as a function of the current traffic conditions. In other terms, w_k and r_k should be evaluated from the speed data v_k in order to implement a procedure like the one introduced in Sect. 4.4. This calculation could be even projected for the future time when the host vehicle will cross that particular link. However, this complete update is seldom practical, due to the large computing time required.

A more practical implementation of an eco-routing system aimed at reducing the computing time, and thus the service time for the user, distributes the calculations between an offline and an online layer, as illustrated by the flowchart in Fig. 5.6.

A remote server (cloud-computing) stores the road network of the preferred region, with N sets of different energy weights and resource consumptions for each link. These sets have been evaluated offline, based on historical traffic information. Typically, they correspond to different daytimes and weekdays to represent the most diverse possible set of traffic conditions.

Through an HMI (see Sect. 8.31), the user enters either an address or coordinates for its destination. In the former case, street addresses are converted by most GIS into geographic coordinates, in a process called *geocoding*. As for the origin of the trip, it is provided by a GPS signal.

These pieces of information are sent to the remote server where calculations are performed. At first, origin and destination locations must be converted into useful nodes or links in the road graph in a process called *map-matching*. For example, the point-to-point method matches a given location to the nearest node in A. Conversely,

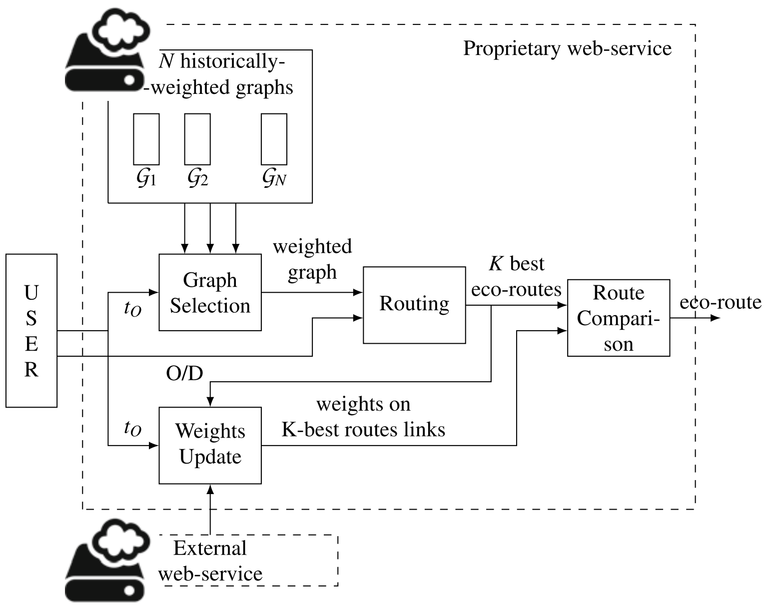


Fig. 5.6 Conceptual sketch of an eco-routing system

the point-to-curve method matches a given location to the nearest point on the nearest road in A^*. When entire trajectories, i.e., sequences of locations, are to be matched, curve-to-curve methods try to match them to the most similar route in the road network.

Based on current time of request (t_O in Fig. 5.6), the most suitable among the N weighted graphs stored is selected. A routing algorithm is then executed on this graph to output the K best routes according to the objective function implemented. For single-objective routing problems, the K best routes can be evaluated using Yen's algorithm, a variant of the Belmann-Ford algorithm. For multi-objective cases, the spanning of the Pareto front described in Sect. 5.1.2.3 already provides several alternative routes.

The actual traffic conditions are provided by an external web-service and transformed in weights and resource consumptions for the selected routes only using, for example, the model of Sect. 4.4.

The K best routes are then evaluated using these updated weights. In other terms, the objective function is evaluated for each of these routes and the constraints on the state are verified to discard possible unfeasible routes. Finally, the best ("eco") route is found as the optimal sequence of links in A^*. To transform this sequence into a sequence of geographical coordinates suitable to be displayed by the HMI, a new map-matching stage is required.

References

1. De Nunzio G, Thibault L, Sciarretta A (2016) A model-based eco-routing strategy for electric vehicles in large urban networks. In: Proceedings of international conference on intelligent transportation systems (ITSC), pp 2301–2306. IEEE
2. Kubička M (2017) Constrained time-dependent adaptive eco-routing navigation system. PhD thesis, Université Paris-Saclay
3. Strehler M, Merting S, Schwan C (2017) Energy-efficient shortest routes for electric and hybrid vehicles. Transp Res Part B: Methodol 103:111–135
4. Fontana MW (2013) Optimal routes for electric vehicles facing uncertainty, congestion, and energy constraints. PhD thesis, Massachusetts Institute of Technology
5. Caramia M, Dell'Olmo P (2008) Multi-objective optimization. Springer, Berlin
6. De Nunzio G, Sciarretta A, Gharbia IB, Ojeda LL (2018) A constrained eco-routing strategy for hybrid electric vehicles based on semi-analytical energy management. In: Proceedings of international conference on intelligent transportation systems (ITSC), pp 355–361. IEEE
7. Dijkstra EW (1959) A note on two problems in connexion with graphs. Numer Math 1(1):269–271
8. Bellman R (1958) On a routing problem. Q Appl Math 16(1):87–90
9. O'Connor D (2012) Notes on the Bellman-Ford-Moore shortest path algorithm and its implementation in MATLAB. Dublin University College, technical report
10. De Nunzio G, Thibault L, Sciarretta A (2017). Bi-objective eco-routing in large urban road networks. In: Proceedings of international conference on intelligent transportation systems (ITSC), pp 1–7. IEEE
11. Aneja YP, Nair KPK (1979) Bicriteria transportation problem. Manag Sci 25(1):73–78
12. De Cauwer C, Van Mierlo J, Coosemans T (2015) Energy consumption prediction for electric vehicles based on real-world data. Energies 8(8):8573–8593

13. Ferreira JC, Monteiro V, Afonso JL (2013) Dynamic range prediction for an electric vehicle. In: Proceedings of world electric vehicle symposium and exhibition (EVS27), pp 1–11. IEEE
14. Neaimeh M, Hill GA, Hübner Y, Blythe PT (2013) Routing systems to extend the driving range of electric vehicles. IET Intell Transp Syst 7(3):327–336
15. Stankoulov P (2015) Vehicle range projection, September 1 2015. US Patent 9,121,719
16. De Nunzio G, Thibault L (2017) Energy-optimal driving range prediction for electric vehicles. In: Proceedings of intelligent vehicles symposium (IV), pp 1608–1613. IEEE