Markov Decision Processes for Train Run Curve Optimization

Nikovski, D.; Lidicky, B.; Zhang, W.; Kataoka, K.; Yoshimoto, K.

TR2012-083 October 2012

Abstract

We propose three computationally efficient methods for finding optimal run curves of electrical trains, all based on the idea of approximating the continuous dynamics of a moving train by a Markov Decision Process (MDP) model. Deterministic continuous train dynamics are converted to stochastic transitions on a discrete model by observing the similarity between the properties of convex combinations and those of probability mass functions. The resulting MDP uses barycentric coordinates to effectively represent the cost-to-go of the approximated optimal control problem. One of the three solution methods uses equal distance steps, as opposed to the usual equal- time steps, to avoidself transitions of the MDP, which allows very fast computation of the cost-to-go in one pass only.

IEEE International Conference on Electrical Systems for Aircraft, Railway and Ship Propulsion (ESARS)

Markov Decision Processes for Train Run Curve Optimization

Daniel Nikovski[1], Bernard Lidicky[1], Weihong Zhang[1], Kenji Kataoka[1], and Koki Yoshimoto[2]

[1]Mitsubishi Electric Research Labs, 201 Broadway, Cambridge, MA 02139, USA

[2]Mitsubishi Electric Corporation, Hyogo 661-8661, Japan

***Abstract*—We propose three computationally efficient methods for finding optimal run curves of electrical trains, all based on the idea of approximating the continuous dynamics of a moving train by a Markov Decision Process (MDP) model. Deterministic continuous train dynamics are converted to stochastic transitions on a discrete model by observing the similarity between the properties of convex combinations and those of probability mass functions. The resulting MDP uses barycentric coordinates to effectively represent the cost-to-go of the approximated optimal control problem. One of the three solution methods uses equal-distance steps, as opposed to the usual equal-time steps, to avoid self transitions of the MDP, which allows very fast computation of the cost-to-go in one pass only.**

I. Introduction

We consider the problem of computing optimal velocity profiles, also known as run curves, of electric trains with regenerative brakes, subject to operating constraints such as speed and acceleration limits. If the distance along the railways track is denoted by z, then the desired velocity $v(z)$ at position z describes the run curve. The dynamics of the moving train can be represented by a simplified set of ordinary differential equations concerning the relative position $z(t)$ of the train along the tracks at time t, and its velocity $v(t)$:

$$\begin{aligned}\dot{v} &= a(z,v,u)\\ \dot{z} &= v\end{aligned}$$

where the function $a(z,v,u)$ describes what acceleration would be experienced by the train if action u is applied to it at position z while moving at speed v. This function incorporates the inertia of the train, as represented by its mass and velocity, the slope (gradient) of the tracks at location z, as well as the air resistance at velocity v. If we represent the state of the train as the vector $x = [z,v]^T$, then we can represent the dynamics by the vector-valued equation $\dot{x} = f(x,u)$.

The instantaneous power consumed by the train is represented by the function $p(z,v,u)$, which we assume depends on position, velocity, and applied control, but is otherwise time independent. When regenerative brakes are used, the function $p(z,v,u)$ can also be negative, representing energy that is generated by the train and returned to the catenary system above the tracks, when it decelerates. A given control trajectory $u(t)$, $0 \le t \le T$ would then result in total energy expenditure of $E(T) = \int_0^T p[z(t),v(t),u(t)]dt$, where T is the end time. Depending on whether the end time T is fixed or not, there are two possible formulations to the run-curve optimization problem:

Formulation F1: The end time T is not fixed, and the goal is to minimize a weighted sum $J = \mu E + (1-\mu)T$ of energy and time, for a suitably chosen weight μ, $0 \le \mu \le 1$. This weight can be chosen from economic considerations, e.g. the relative cost of a unit of energy (kWh) vs. that of a unit of time (second) for all passengers in the train.

Formulation F2: The end time T is fixed and specified in advance, for example from an existing timetable, and the goal is to minimize the energy only: $J = E$.

In both cases, the goal is to find a function $u(t)$, $0 \le t \le T$, that minimizes the cost functional $J[u(t)]$, subject to the dynamics of the train $\dot{x} = f(x,u)$, and the constraints/conditions $z(0) = 0$, $z(T) = Z$, $v(0) = v(T) = 0$, and $0 \le v(t) < v_{max}(t)$, where Z is the distance between the origin and destination stations, and $v_{max}(t)$ is the speed limit for location $z(t)$.

Both formulations F1 and F2 represent optimal control problems, and it is well known that the optimal function $u(t)$ can be found by solving the Hamilton-Jacobi-Bellman (HJB) equation I.1. If we define the instantaneous cost incurred at time t if control u is applied at state x as $c(x,u)$, and the optimal cumulative cost-to-go until the end destination as $V(x,t)$, the HJB equation allows us to relate the time derivative of V to the instantaneous cost c and the gradient of V in state space [1]:

$$\frac{dV(x,t)}{dt} + \min_u \{c(x,u) + \nabla V(x,t) \cdot f(x,u)\} = 0 \tag{I.1}$$

For formulation F1, the immediate cost function $c(x,u)$ is defined as $c(x,u) \doteq \mu p(x,u) + 1 - \mu$, and for formulation F2, $c(x,u) \doteq p(x,u)$.

The HJB equation is a partial differential equation (PDE) that is seldom possible to solve analytically. Specifically for train run curve optimization, analytical solutions do not appear to be available, and numerical methods must be applied instead [2]. In general, implementing and verifying direct solutions to the HJB equation is quite difficult, and results in slow computation. In the following section, we describe alternative solutions methods based on Markov decision processes. Two of these methods are for formulation F1, and one is for formulation F2.

II. Markov Decision Processes for Run-Curve Computation

Our general solution approach is to represent the continuous-state-space problem in the form of a Markov decision process (MDP), and solve the MDP by means of dynamic programming, value iteration, or policy iteration [3], [4]. A discrete-space MDP is described by the tuple (S, A, P, R). It has a discrete set S of N states $s^{(i)} \in S$, $1 \le i \le N$ such that the MDP occupies one of these states $s_k \in S$ at any time t_k, and a set U of L actions $u^{(l)} \in U$, $1 \le l \le L$ that can be applied at any given time. We assume that the starting state s_0 is known and fixed. A transition probability function P expresses the probability $P_{ijl} \doteq Pr(s_{k+1} = s^{(j)} | s_k = s^{(i)}, u_k = u^{(l)})$ of being in state $s_{k+1} = s^{(j)}$ at time t_{k+1} if the MDP was in state $s_k = s^{(i)}$ at time t_k and control (action) $u_k = u^{(l)}$ was applied at that time. Similarly, a reward function R expresses the reward (or cost) $R_{il} \doteq r(s_k = s^{(i)}, u_k = u^{(l)})$ of applying action $u_k = u^{(l)}$ to state $s_k = s^{(i)}$ at time t_k. The MDP evolves in discrete decision epochs that might occur at regular time intervals (e.g., $t_k = k\Delta t$), or might not have a fixed time duration attached to them. The goal is to optimize a performance measure $J = \sum_{k=0}^{K} r(s_k, u_k)$.

At any given moment, we restrict the actions that the train controller can be executing to one of the following four: accelerating (u_1), decelerating (u_2), running at a constant speed (u_3), and coasting (moving due to the train's own momentum, u_4). Such a restriction appears to be typical for automatic train operation (ATO) systems, and would result in a very compact representation of the optimal action sequences.

The continuous-state and continuous-time dynamics of the train are also discretized to create discrete state space of the MDP, but there are three different methods to do that, depending on the formulation of the problem (F1 or F2) and the MDP solution method. These three methods are described below.

III. Equal-Time MDP for F1

In this method, time is discretized at constant time steps of length Δt, such that decisions and state transitions happen at times $t_k = k\Delta t$, where k is an integer. The equations of motion of the train are integrated forward in time for one time step to obtain a set of difference equations for the successor state at the end of that time step: $x_{k+1} = F(x_k, u_k)$. Similarly, the immediate cost $C(x_k, u_k)$ incurred during one epoch is the integral of the instantaneous cost $c(x,u)$ over that epoch.

The similarities between train dynamics and MDPs are that both evolve in discrete time under the effect of a small number of discrete actions, and both seek to optimize a performance criterion defined over states and actions. The two major differences are in the type of state used (continuous $x \in R^2$ vs. discrete $s \in S$) and in the way state evolution is described (function $F(x,u)$ vs. probability transition function P_{ijl}). The objective of the conversion method, then, is to construct a state set S embedded in R^2 and a transition function P_{ijl} for every triple $(s^{(i)}, s^{(j)}, u^{(l)})$. After the MDP is constructed, an optimal policy $u = \pi(s^{(i)})$ that maps states to optimal controls can be found for every $s^{(i)} \in S$, by using well known algorithms such as policy iteration and value iteration [5].

The proposed method is based on similarities in the mathematical properties of probability functions and convex combinations. A probability function (also called sometimes a probability mass function to distinguish it from a probability density function) specifies the probability that a random variable is equal to some specified value. For the case of MDPs, the transition function is such a (conditional) probability mass function, conditioned on the starting state $s_k = s^{(i)}$ and the applied control $u_k = u^{(l)}$. The random variable for which the probability function is specified is the successor state s_{k+1}. If the size of the state set S is N, let $s^{(1)}$, $s^{(2)}$, ..., $s^{(N)}$ be an enumeration of all states. The elements of the transition function can then be defined as $p_j \doteq P_{ijl} = Pr(s_{k+1} = s^{(j)} | s_k = s^{(i)}, u_k = u^{(l)})$. From the axiomatic properties of probability mass functions, then, it is always true that $\sum_{j=1}^{N} p_j = 1$, and $p_j \ge 0$, $j = 1, N$.

On the other hand, a convex combination of N vectors y_j, $j = 1, N$ is defined as $\sum_{j=1}^{N} c_j y_j$, such that $\sum_{j=1}^{N} c_j = 1$, and

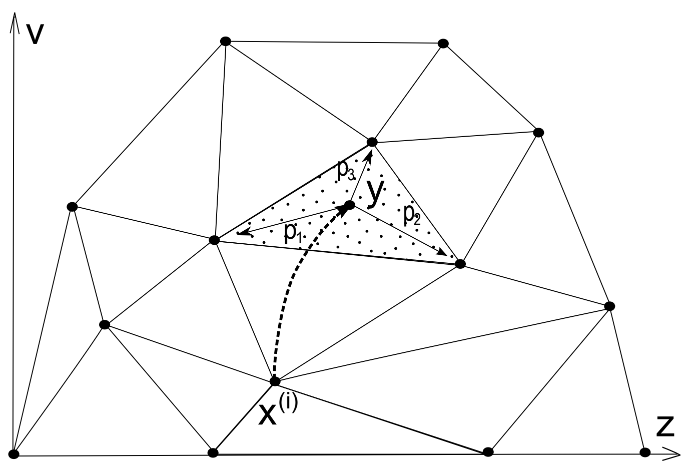

Figure III.1. Triangulation of the state space of a moving train. The transition resulting from one action (acceleration) is shown.

$c_j \geq 0$, $j = 1,N$. By comparing the two definitions, it can be observed that probability mass functions and the set of coefficients defining a convex combination obey exactly the same mathematical constraints, and a valid probability function can be used as coefficients of a valid convex combination, and vice versa. We use this fact to construct all transition functions of the MDP as sets of coefficients for suitably defined convex combinations.

A. *Conversion Algorithm*

The algorithm starts with selecting N states $s^{(1)}$, $s^{(2)}$, ..., $s^{(N)}$ such that each corresponds to a state $x \in R^2$. We denote the continuous state that corresponds to MDP state $s^{(i)}$ by $x^{(i)}$. Every $x^{(i)}$ is a point (vector) in 2-dimensional Euclidean space. Call the set of points $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$. The destination station (point $[Z,0]^T$) should also be in the set X.

The next step is to find the Delaunay triangulation $DT(X)$ of the set of points X (Figure III.1). The Delaunay triangulation consists of triangles each of which has 3 vertices, such that each of these vertices is a member of X. Then, for each point $x^{(i)}$ that corresponds to state $s^{(i)}$, we execute the system function f of the train dynamics to find the successor point y of $x^{(i)}$ under control $u^{(l)}$: $y = f(x^{(i)}, u^{(l)})$.

In general, the successor point y does not coincide with any of the pre-selected points $x^{(i)}$, $i = 1,N$, that is, the transition from a vertex $x^{(i)}$ is not necessarily to another vertex $x^{(j)}$, but somewhere between the vertices. Our proposal, and the key idea of this paper, is *to treat that transition instead as a probabilistic transition to one of the three vertices of the triangle in $DT(X)$ that contains the point y*. In order to find this triangle, we traverse all M simplices in $DT(X)$ and find the barycentric coordinates c_1, c_2, and c_3 of y, i.e. the three coefficients that satisfy $y = c_1 x^{(m,1)} + c_2 x^{(m,2)} + c_3 x^{(m,3)}$, where $x^{(m,1)}$, $x^{(m,1)}$, and $x^{(m,1)}$ are the three vertices of triangle m, $m = 1,2,\ldots,M$. Triangle m does indeed contain point y if and only if the three coefficients constitute a convex combination, i.e., they are all positive. (Their sum is always one.)

As noted above, the transition function of an MDP has the exact same properties: the sum of all probabilities is one, and they are all positive. With this observation, we can easily construct a complete transition probability distribution over all possible N successor states $s^{(j)}$: if $s^{(j)}$corresponds to one of the vertices of the enclosing triangle m, that is, $x^{(i)} = x^{(m,d)}$ for some $d \in 1,2,3$, then the corresponding transition probability of the MDP is $P_{ijl} = Pr(s_{k+1} = s^{(j)} | s_k = s^{(i)}, u_k = u^{(l)}) = c_d$; otherwise, $P_{ijl} = 0$. Finally, the state that corresponds to the end station $[Z,0]^T$ always transitions to itself with probability one, that is, it is an absorbing, terminal state.

Defining the reward function is straightforward: the reward (cost) of taking action $u_k = u^{(l)}$ in state $s_k = s^{(i)}$ is equal to the cost incurred in decision epoch k (between times t_k and t_{k+1}) if action $u_k = u^{(l)}$ is applied in the corresponding state $x^{(i)}$: $r(s^{(i)}, u^{(l)}) = C(x^{(i)}, u^{(l)})$. The cost of any action for the terminal state is zero. Since the reward in this case has the meaning of cost, the objective is to minimize its cumulative value $J = \sum_{k=0}^{K} r(s_k, u_k)$ until the terminal state is reached. With this conversion, the deterministic continuous system dynamics of the train are converted to a stochastic discrete MDP. Full details of the conversion algorithm and its computational complexity are available in [4].

B. *Solving the Equal-Time MDP*

It should be noted, however, that the constructed MDP might contain self-transitions. It can be solved by means of the value iteration algorithm which consists of executing the following assignment in multiple sweeps over the entire state space of the MDP until the value function $V(s)$ converges:

$$V(s) := \min_u [R(s,u) + \sum_{s'} Pr(s_{k+1} = s' | s_k = s, u_k = u) V(s')] \tag{III.1}$$

A single such assignment is known as a Bellman back-up and is computationally inexpensive, because there are at most three possible successor states s' for each state s. Once the value function converges, the value function will satisfy equation III.1

as equality. After that, the optimal policy for the MDP can be determined as $\pi^*(s) = argmax_u Q(s,u)$, where we make use of the auxiliary function $Q(s,u) \doteq R(s,u) + \sum_{s'} Pr(s_{k+1} = s' | s_k = s, u_k = u) V(s')$.

The goal, however, is to find a control law $u = \rho(x)$ that is a mapping from the *continuous* state x of the moving train, as opposed to the discrete state of the embedded MDP s. In order to find such a law, we use the barycentric coordinates to estimate the merit $\hat{Q}(x,u)$ of the individual action u taken in state x as $\hat{Q}(x,u) = \sum_{j=1}^{3} c_j Q(s^{(j)}, u)$, and use the control law $\rho(x) = argmax_u \hat{Q}(x,u)$. Given that the barycentric coordinates c can be interpreted as individual probabilities that the MDP is in one of its discrete states, the function $\hat{Q}(x,u)$ is indeed the exact expected merit of taking action u at the continuous state x.

IV. Equal-Distance MDP for F1

The biggest computational drawback of the equal-time (ET) MDP is its relatively long solution time, due to the need to use the value iteration algorithm. It would be advantageous to construct an MDP without self-transitions, and we can achieve that if we use a regular rectangular grid such that the phase space (z,v) of the moving train is divided into multiple vertical lines $z = z_j$, and each of these lines is covered by a number of MDP states (Figure IV.1). Six lines are shown in that figure, such that all states on a given line have the same coordinate z. The transition rules of the MDP are also changed: each transition starts at a state with coordinates $[z_j, v]$, for some line index j and velocity v, and the chosen action u is executed until the distance coordinate reaches the next line: $z = z_{j+1}$. The duration of that transition can vary according to the starting state and the chosen action, but is uniquely determined by them.

In that case, if we apply the decomposition of the ending state y into barycentric coordinates described above, it will result in at most two non-zero values for the three coordinates (p_2 and p_3 in Figure IV.1), because the ending state y will always lie on one of the sides of a triangle in the Delaunay triangulation of the state space. Equivalently, when the barycentric coordinates are interpreted as transition probabilities of an MDP, transitions will always start at a state on line z_j and end in one or two states on line z_{j+1}, meaning that no self transitions would exist in this MDP.

If we then group all states lying on line z_j into stage j, the resulting MDP could be decomposed in sequential stages, such that the train moves from stage to stage in each decision step.

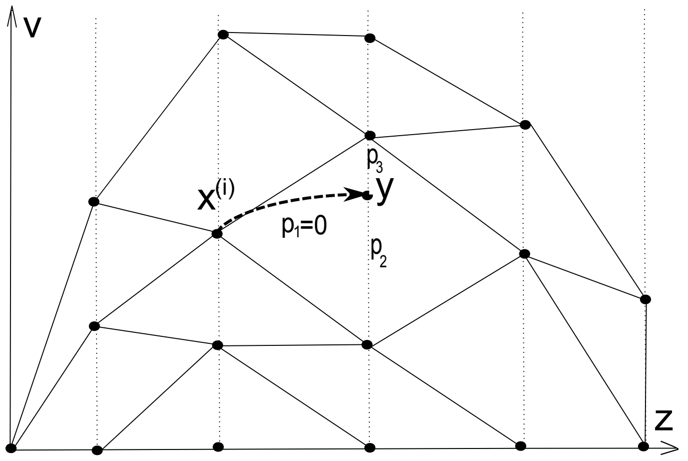

Figure IV.1. In equal-distance MDPs (ED-MDPs), states are located on a number of vertical lines, such that multiple states have the same distance coordinate z_j. Such states form a stage, and transitions happen only between consecutive stages, allowing fast solutions by means of backward dynamic programming.

Then, by performing Bellman back-ups (Equation III.1) stage by stage, starting with the last stage and proceeding backward in time (that is, performing dynamic programming), the value function could be determined in only one sweep of the state space, rather than in multiple sweeps required by the value iteration algorithm.

We will call the resulting MDP an equal-distance MDP (ED-MDP), because all transitions between states in stages j and $j+1$ cover the same distance along the railway track (equal to $z_{j+1} - z_j$). It can be seen that the ED-MDP model is a special case of the general MDP approach, where all barycentric coordinates are degenerate in a way that at most two of them are greater than zero. From computational point of view, finding these positive coordinates is also much simpler than in the general case, because only the states in stage $j+1$ must be considered, and if their velocities are sorted in ascending order, binary search on them can be used to find which two states have positive transition probabilities.

V. Equal-Time MDP for F2

In order to solve the second formulation of the problem, where the end time T is fixed, we augment the state space of the MDP with a time coordinate t_k, such that $t_k = k\Delta t$, and $T = t_K = K\Delta t$. In this case, the end time T must be an integer multiple of the time step Δt. Each state of the MDP, then, is described by the triple $[z,v,t]$, and transitions occur between consecutive time steps. The state space of the MDP can be organized into time slices, where each slice is a replica of the

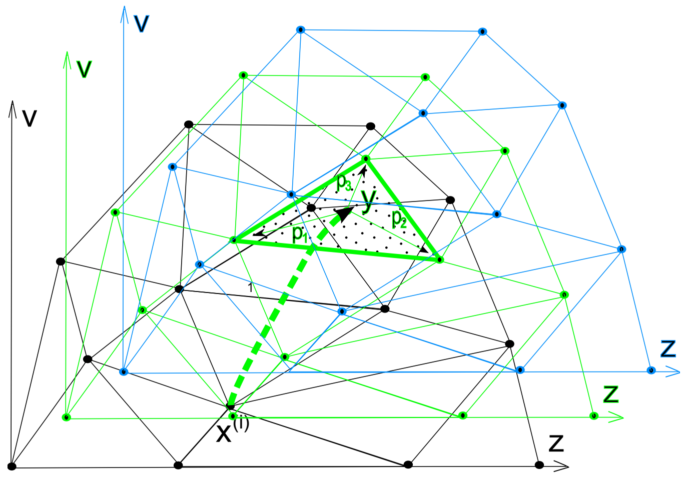

Figure V.1. The MDP for the fixed-end-time version of the problem has a separate replica of the triangulation for multiple time steps, organized into time slices. Three slices are shown here, in black, green, and blue, respectively. Each transition is from one time slice to the next (here, a transition is shown from $x^{(i)}$ in the black slice to y in the green slice).

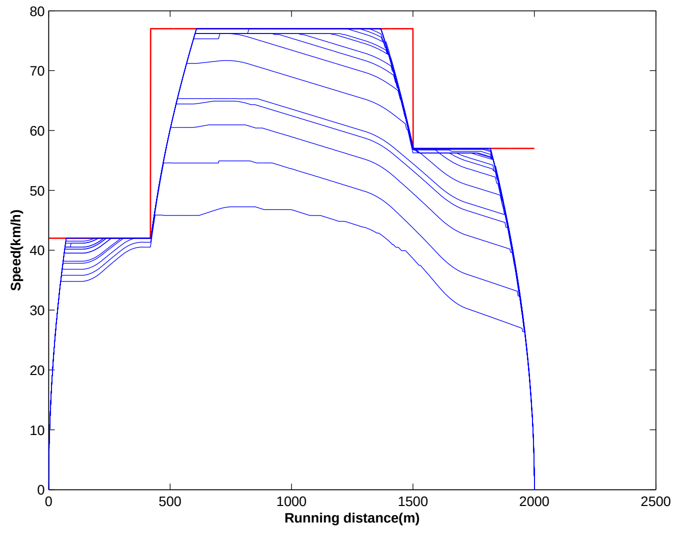

Figure VI.1. Run curves for solver VTT-ED: variable end time, F1, ED-MDP. The red line shows the speed limit at the current location of the railroad.

triangulation for the ET-MDP case (Figure V.1). Each transition takes place between a starting state in one slice and an end state in the next slice.

When finding the optimal policy for this MDP, the goal is to compute the value function $V(s,t)$ for every augmented state (s,t), where the state now includes time. In general, the value function for the same state s, but different times t_1 and t_2, is not the same. In this case, the Bellman back-up for the augmented states looks as follows:

$$V(s,t_k) := \min_u [R(s,u) + \sum_{s'} Pr(s_{k+1} = s' | s_k = s, u_k = u) V(s', t_{k+1})] \tag{V.1}$$

Since the successor states s' always lie in slice $k+1$ (for time t_{k+1}), there are no self transitions in the MDP. Its value function can be computed by means of dynamic programming, using the above equation, starting with the last slice (K), and proceeding backward in time until the starting slice and state. Although the method is not iterative, and only a single sweep over the augmented state space is necessary, this MDP has K times more states than the one for the ET-MDP or ED-MDP for F1, and its computation time is that many times longer.

VI. Experimental Results

We tested the accuracy and computational speed of the three solution methods on a test case with a track of length 2000 meters, with varying slopes and speed limits, and fastest possible running time of 139 seconds (Figure VI.1, showing also multiple run curves for progressively increasing running times. Figure VI.2 describes the experimentally computed optimal trade-off (Pareto boundary) between running time and consumed energy, for the three computational methods described. The states of the MDP were placed on a regular rectangular grid in state space, the same for all methods. Both the variable terminal time case (F1, denoted by VTT), and the fixed terminal time case (F2, denoted by FTT), are shown. VTT-ED is the curve for the ED-MDP case, and VTT-ET-1 and VTT-ET-2 are curves for the ET-MDP case, from two different implementations. All four curves describe the same underlying trade-off, but computed by means of different methods. Moreover, all four are approximations to the true curve, and their close grouping suggests that they have similar accuracy. Overall, the ED-MDP method shows the smoothest Pareto boundary, and finds best optima for longer running times.

Next, we explored the effect of the size of the MDP on optimality. The size of the MDP is determined by the number of states of the MDP, and when these states are located on a regular rectangular grid, their number depends on the discretization step of the grid. Figure VI.3 shows the time/energy trade-off curves for different discretization steps. The computed curve for 101 steps in distance and 40 in velocity is not substantially different from the one for 201 steps in distance in 80 in velocity, and is four times faster to compute. This suggests that the discretization steps for the former curve (of size 20m along the distance direction and 2km/h in the velocity direction) are sufficient to compute close to optimal run curves.

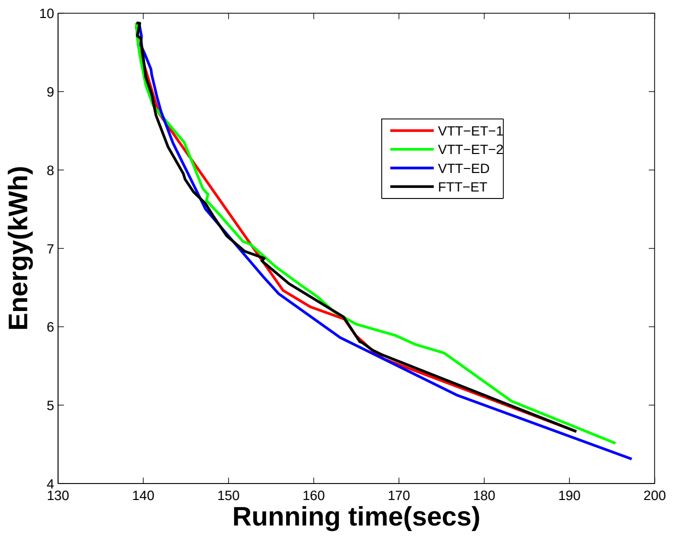

Figure VI.2. Optimal trade-off between running time and consumed energy, using both the variable terminal time case (F1, denoted by VTT), and the fixed terminal time case (F2, denoted by FTT). VTT-ED is the curve for the ED-MDP case, and VTT-ET-1 and VTT-ET-2 are curves for the ET-MDP case, from two different implementations. All four curves describe the same underlying trade-off, but computed by means of different methods.

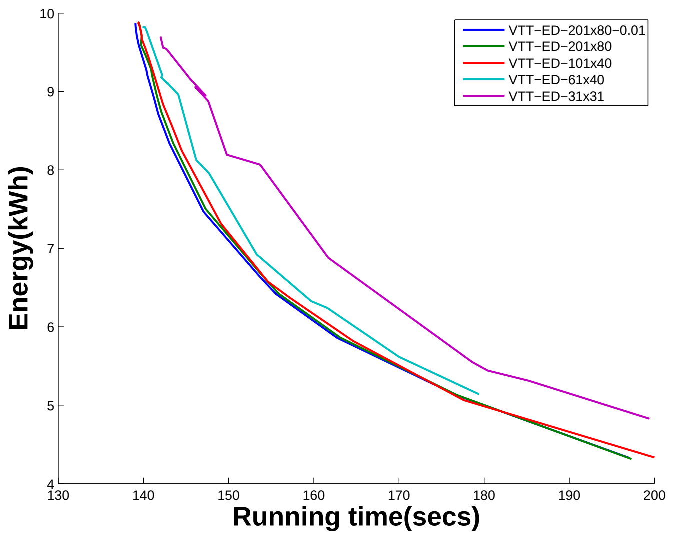

Figure VI.3. Time/energy trade-off for different discretization levels. The computed curve for 101 steps in distance and 40 in velocity is not substantially different from the one for 201 steps in distance in 80 in velocity, and is four times faster to compute.

That figure also contains a line labeled “VTT-ED-201x80-0.01” for the case when the simulation/control step of the train system was equal to $0.01s$, whereas all other graphs used step of $0.1s$. Visual comparison with the graph labeled “VTT-ED-201x80-0.1” suggests that using the smaller time step does have some minimal effect on the optimality of the curve, but using time steps of $0.1s$ should be acceptable for practical operation.

In terms of computational time, the three methods exhibited very different speeds. All experiments were performed on a computer with Intel Core 2 Duo E6600 CPU (2.40GHz), and all solvers were implemented in MATLAB 7.9.0 (R2009b). The VTT-ED method is by far the fastest of all three, because its MDP has no self-transitions or loops, and has many fewer states than the MDP for the FTT case (F2). For example, it took only 20.6 seconds to compute the optimal policy for an MDP with 201 steps in distance, and 40 states (velocities) per line.

For comparison, the VTT-ET-2 solver took 527 seconds on an MDP with 201 steps in distance, 81 steps in velocity, and 5 seconds transition time. Its computation time depends strongly on the time step: the same discretization, but with 10-second transition times, resulted in computation time of 993 seconds. Based on this, we can conclude that the ED-MDP method is at least an order of magnitude faster than the ET-MDP method, and should be preferred in practice. This can be attributed to the lack of cycles and self-transitions in the ED-MDP.

VII. Conclusions and Future Work

Three methods for converting train dynamics and run-curve optimization problems into MDPs were proposed and tested on the same test problem. Of these, it is recommended to use the ED-MDP method for its high speed and smooth resulting run curves. Experimental results suggest that discretization steps of 20m in distance and 2km/h in velocity are sufficient for computation of accurate optimal run curves.

Future work will focus on methods for speeding up the computation for the fixed terminal time case, and representing the control law compactly. Intuitively, each slice contains many states that the train cannot be in — either cannot get to them at that time, or if it is there, cannot get to the destination station at time t_K. It might be possible to prune these states out of the state space of the MDP, speeding up computation even more.

References

[1] R. F. Stengel, *Optimal Control and Estimation*. Mineola, NY: Dover, 1986.

[2] H. Ko, T. Koseki, and M. Miyatake, “Application of dynamic programming to optimization of running profile of a train,” in *Computers in Railways IX*, A. et al., Ed., 2004, pp. 103–112.

[3] J. Kushner and P. Dupuis, *Numerical Methods for Stochastic Control Problems in Continuous Time*. Heidelberg: Springer Verlag, 2001.

[4] D. Nikovski and A. Esenther, “Construction of embedded Markov decision processes for optimal control of non-linear systems with continuous state spaces,” in *IEEE Conference on Decision and Control and European Control Conference*, M. Polycarpou, Ed., 2011, pp. 7944–7949.

[5] M. L. Puterman, *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. New York, NY: John Wiley & Sons, Inc., 1994.