MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

A Framework for Real-time Near-optimal Train Run-curve Computation with Dynamic Travel Time and Speed Limits

Xu, J.; Nikovski, D.N.

TR2015-063 July 2015

Abstract

This paper studies the problem to generate the most energy efficient run-curves subject to given travel time requirements. The target is to provide a train with the ability to quickly adjust its run curve according to different travel time requirements and speed limits along the track before departing a terminal. Using a train model considering train length, varying track gradient and speed limit profile, the optimal run-curve problem is formulated into a bi-criteria optimization problem that minimizes weighted energy consumption and weighted travel time. By selecting appropriate weight values, the optimization problem would generate a run-curve with near-optimal energy consumption. We propose a two stage procedure framework, which includes an off-line stage and a real-time stage. A series of geometric relation between weight in the objective function and travel time are derived. The actual run-curves are generated in the real-time stage using approximate dynamic programming. For the first time, a framework provides trains with the ability to fast response to dynamic travel time requirement using complex physical models.

2015 American Control Conference (ACC)

A Framework for Real-time Near-optimal Train Run-curve Computation with Dynamic Travel Time and Speed Limits

Jingyang Xu[1], Daniel Nikovski[2] and Sae Kimura[3]

Abstract— **This paper studies the problem to generate the most energy efficient run-curves subject to given travel time requirements. The target is to provide a train with the ability to quickly adjust its run curve according to different travel time requirements and speed limits along the track before departing a terminal. Using a train model considering train length, varying track gradient and speed limit profile, the optimal run-curve problem is formulated into a bi-criteria optimization problem that minimizes weighted energy consumption and weighted travel time. By selecting appropriate weight values, the optimization problem would generate a run-curve with near-optimal energy consumption. We propose a two stage procedure framework, which includes an off-line stage and a real-time stage. A series of geometric relation between weight in the objective function and travel time are derived. The actual run-curves are generated in the real-time stage using approximate dynamic programming. For the first time, a framework provides trains with the ability to fast response to dynamic travel time requirement using complex physical models.**

I. Introduction

Energy efficiency becomes important due to the increasing fuel cost and aware of environment effects. In electrified railroads, energy efficiency improvements are been practiced by designing more energy efficient locomotives, using higher voltage for electricity transmission, installing regenerative brakes, and installing on-board and sideways batteries. With upgraded infrastructure and hardware, the potential energy efficiency is improved by reducing the unnecessary energy waste. Run-curve optimization plays a key role in this process as a significant amount of energy waste is related to the unoptimized train operation. Run-curves represent the relationship of positions and target speeds in order to run between stations, and are also referred to as speed profiles. Run-curves are made in advance based on scheduled travel time, and train drivers and Automatic Train Operation (ATO) run trains along them. However, target travel time may be changed from a schedule depending on real-time train operation, and different speed limits may be applied because of weather condition and preceding trains. It is desired that a run-curve to a next station can be calculated for on-board processors in a relatively short amount of time, in which a train stops at a station.

Due to the importance of energy-efficient run curves, there are a number of previous research results. The methods to calculate the optimal run curve fall into two categories: analytical solutions and numerical optimization. Physical models for the traction, resistance and brake forces on trains often play a fundamental role on selecting optimization methods for run-curve computation. When relatively simple physical models are sufficient for specific applications, analytical solutions can be obtained and leads to fast computation of optimal run-curves [3], [14], [5], [16]. Relatively simple track profiles and physical models are sufficient when trips are long and geographic condition can be represented in well-structured expression. They also help on reducing the complexity of the problem by converting the run-curve optimization into coasting point optimization problem. Analytical solution has the advantages of fast computation and giving guidance to driving strategies. The simple track profile might be valid for long distance travel. Thus, even optimal strategy can be derived based on these models [9], [8]. However, when the trip is relatively short so that more complex track profile and physical models should be considered, analytical solution can not be obtained [15].

Numerical optimization becomes a natural approach when more complex physical models are used. Meanwhile, more powerful on-board processors are required for computation. Genetic Algorithm (GA) is a popular method as it is capability to handle multiple inputs and nonlinear physical models. Significant improvement on using GA has been reported in [2], [1]. However, as a meta-heuristic algorithm, GA can not give optimality condition for the solution by itself. As the number of possible coasting points increases, the computational time increase dramatically. A long computation time like 12 hours is required for optimal solutions [15].

Another important numerical optimization method for solving run-curve optimization problem is Dynamic Programming (DP). Similar to GA, DP is capable to handle complex physical models. At the same time, DP has the advantage of generating guaranteed optimal solution with predictable computational time. The weakness for DP is the attempt to reduce discretization error can lead to significant increase of computational effort. In the previous studies, [6] sets the state space using velocity and time, and generate solution in 22 seconds. [13] also uses velocity and time to set up state space and focuses on the design of vehicles instead of fast run-curve optimization. [10], [7], [11] use velocity and position to set up state space, but the significant increasing requirements computational time still limits the application of these methods. The optimal solution requires huge amount of computational time. This would limit its usage with on-board computer and the changing trip time requirement from the dispatchers before the departure of trains. As it is mentioned in [11], dynamic programming is slow for real time implementation.

[1] Corresponding author, Email: jxu7@buffalo.edu
[2] Mitsubishi Electric Research Laboratories
[3] Mitsubishi Electric Corporation, Advanced Technology R&D Center

In this paper, we propose a framework based on dynamic programming and calculate the optimal run-curve within a very short time, which makes it suitable for real-time implementation with an on-board processor. The framework is based on solving a bi-criteria, with weighted energy consumption and weighted travel time, optimization problem using dynamic programming. By introducing an off-line stage, the transition matrix and a monotonic relation between weight and travel time are pre-computed. For a problem with only dynamic travel time requirements, the weight for the optimization problem is obtained by directly interpolating the monotonic relation between weight and travel time. For problem with both dynamic travel time requirements and speed limits, the transition matrix is partially updated and the weight searched with in a bounds based on a series of geometric relation derived. The scenarios considered in this paper assume that travel time requirements and additional speed limits are finalized before the departure of a train. Thus, responding quickly to updated travel time requirements and speed limits enhances a railroad operator's capability to adjust to timetable changes and track condition changes. Meanwhile, a fast optimal run-curve optimization method can also help reduce computational efforts in a large network simulation model.

To the best of our knowledge, this is the first time a fast algorithm is realized on obtaining optimal or near optimal run-curves with respect to dynamic travel time requirement when complex physical models and track profiles are considered. Related patent materials to this paper can be found in [17], [18].

In the remainder of this paper, section II introduces the physical models and assumption for this paper. Section III describes the framework and dynamic programming used. A set of computational experiments and results are shown in section IV. Finally, section V gives a short conclusion.

II. Problem description

A. Problem Description

The problem we consider is a single trip problem as shown in Figure 1. A train travels from terminal A, the origin terminal, to terminal B, the destination terminal. The travel time requirement follows a designated timetable or order from control center. The train is equipped with regenerative brake and on-board battery for regenerative energy storage. The track between the two terminals has different gradient at different location. We assume the track gradient profile does not change during the trip.There are two scenario considered in this paper on speed limits. The first scenario is that speed limits do not change. The second scenario is that new speed limit is added at the departure time. The train has four possible actions: acceleration, constant speed, coasting, and braking. The notation used to describe the problem is as following:

- T is the designated travel time requirement,
- L is the total length of the track from origin terminal to destination terminal,

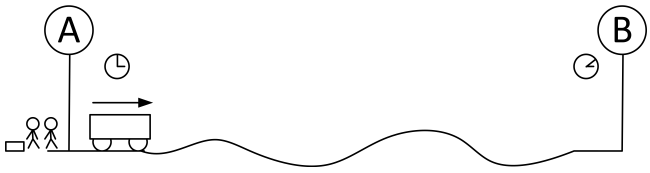

Fig. 1. Single Trip Traveling

- t is the time during the trip with $t = 0$ for departure time and $t = T$ for arrival time,
- $E(t)$ is the energy consumption for the train from departure to time t,
- $x(t)$ is the position information for train at time t,
- $u(t)$ is the action the train takes at time t,
- $v(t)$ is the velocity of the train at time t,
- $f(x(t), u(t), v(t))$ is a function for energy consumption rate at time t based on $x(t)$, $u(t)$, and $v(t)$,
- $g(x(t), u(t), v(t))$ is a function for the train's acceleration calculation at time t based on train's position, action and velocity information $x(t)$, $u(t)$, and $v(t)$,utilized
- $V^{max}(x(t))$ is the maximum allowable speed along the track given the position of train $x(t)$.

The objective function of the problem can be stated as:

Minimize:

$$E(T) = \int_0^T f(x(t), u(t), v(t))dt \tag{1}$$

Subject To:

$$\frac{dx(t)}{dt} = v(t) \tag{2}$$

$$\frac{dv(t)}{dt} = g(x(t), u(t), v(t)) \tag{3}$$

$$v(t) \leq V^{max}(x(t)) \tag{4}$$

$$x(T) = L \tag{5}$$

$$u(t) \in \{1, 2, 3, 4\}, t \in [0, T], \tag{6}$$

$$v(t) \geq 0, t \in [0, T] \tag{7}$$

$$L \geq x(t) \geq 0, t \in [0, T]. \tag{8}$$

In the formulation above, (1) is the objective function whose value is the total energy consumption of the entire trip. Constraints (2) and (3) are the dynamic relations between train's position, velocity, and acceleration. Constraint (4) is the speed limit constraint for the run curve. The trip distance constraint is stated in constraint (5). (6) describes the possible action set for the system, where $u(t) = 1$ means acceleration, $u(t) = 2$ means constant speed, $u(t) = 3$ means coasting, and $u(t) = 4$ means brake. (7) and (8) are bound constraints for the train's velocity and position state variables.

The run curve optimization problem's track information considered in this paper is shown in Figure 2. The speed limit is related to the position of train and has a piece-wise constant shape. The gradient along the track is also related to the position of train and is piece-wise constant.

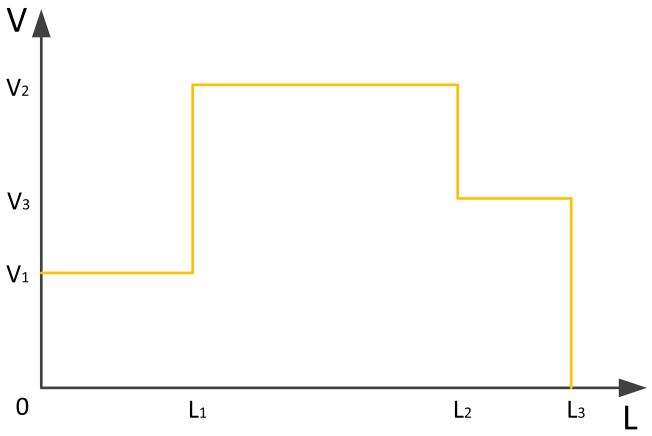

Fig. 2. An Example Speed Limit Profile

B. Physical Models

As it is reported in the literature, dynamic programming is capable to perform under various types of physical models. Our physical model used in this paper for testing the framework is introduced here.

The train with length L' and weight M has the acceleration $\alpha^{tot} = g(x(t), u(t), v(t))$ that is caused by a namely force F^{tot}. The acceleration α^{mot} and F^{mot} comes from four different sources during the trip:

- The traction force F^{mot} comes from train motor, which leads to an acceleration component $\alpha^{mot} = \frac{F^{mot}}{M}$.
- The resistance force F^{brk} comes from breaks, which leads to an acceleration component $\alpha^{brk} = \frac{F^{brk}}{M}$.
- The air resistance force F^{air} comes from air, which leads to an acceleration component $\alpha^{air} = \frac{F^{air}}{M}$.
- The traction or resistance force F^{grd} comes from gradient of the track, which leads to an acceleration component $\alpha^{grd} = \frac{F^{grd}}{M}$. The gradient force F^{grd} is calculated by summing all forces along the entire train, such that $F^{grd} = \int_{-L'}^{0} g^{grd}(x(t)+s)ds$. In the equation to calculate F^{grd}, where $g^{grd}(x(t)+s)$ represents the unit gradient force at time t at position s on the train, and F^{grd} is assumed equivalently on the head of the train.

The total forces on the train is the summation of the four sources, represented as $F^{tot} = F^{mot} + F^{brk} + F^{air} + F^{grd}$. Correspondingly, the total acceleration on the train is also the summation of the four sources represented as $\alpha^{tot} = \alpha^{mot} + \alpha^{brk} + \alpha^{air} + \alpha^{grd}$.

Since a trip between two terminals involves energy consumption and energy regenerating, the energy consumption for an operation plan is calculated by considering both energy consumption and regeneration, as shown in equation (9). In (9), e_1 and e_2 are two parameters stating the motor and brake energy efficiency.

$$E(t) = \int_0^t [\frac{M\alpha^{mot}(t)v(t)}{e_1} - M\alpha^{brk}(t)v(t)e_2]dt \quad (9)$$

C. Operational Constraints and Assumptions

The operational regulations and constraints are mostly related to the geographic and track conditions, for example track curvature, tunnel, track design specification. All these restrictions have their impact to the system in form of train speed limits along the track. Thus, the regulations and constraints, in the most simple way, can be described as a velocity limit function of the track position. When a train is subject to multiple speed limits due to the train length, we use the minimum speed limit as the speed limit for the train.

Some additional assumptions on the physical models in this paper are:

- Locomotive's traction forces is constant to different speed of trains.
- All the regenerative energy is collected with a constant efficiency level.
- The initial velocity of the train, before the train's departure from origin terminal, is 0.
- The final velocity of the train, after the train's arrival at destination terminal, is 0.

III. Methodology

A. The Framework for The Core Problem

We solve this run-curve optimization problem by setting up a bi-criteria optimization problem:

Minimize:

$$\mu E(T) + (1-\mu)T \quad (10)$$

Subject To: (2), (3), (4), (5), (6), (7), and (8).

This bi-criteria optimization problem is very similar to the original run-optimization problem. The differences are the objective function, and the condition that T becomes a variable in the bi-criteria optimization problem instead of given fixed value in the original problem. This bi-criteria optimization problem has been studied by a couple of previous researchers. In [4], the bi-criteria optimization problem is studied and solved using GA for selecting travel time that can save more energy. In [1], optimal strategies are studied also using GA for optimal driving strategies. As Pareto frontiers are profiled for this bi-criteria optimization, one important feature has not been well utilized in previous results: When a μ' value is set, a solution to the problem gives a pair of travel time T' and corresponding energy consumption $E(T')$, where $E(T')$ is the minimum energy consumption if the travel time is designated to be T'. Thus, the run-curve optimization problem with given travel time requirement can be solved via solving a bi-criteria optimization problem with appropriate μ values. By doing this, the original problem is set into a framework that solves a two level optimization problems: the lower level is the bi-criteria optimization problem, the upper level is an optimal μ value problem. A simple proof that supports the validity procedure is as following:

Lemma 1: The optimal solution, which has energy consumption $E(T)$, and travel time T, and run-curve profile, for a specific μ value in the bi-criteria optimization problem with objective function (10) and constraints (2), (3), (4), (5), (6),

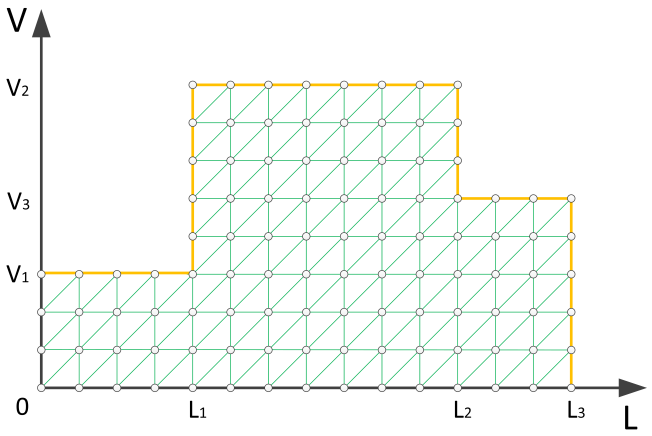

Fig. 3. Delaunay Triangulation with Acceleration Actions

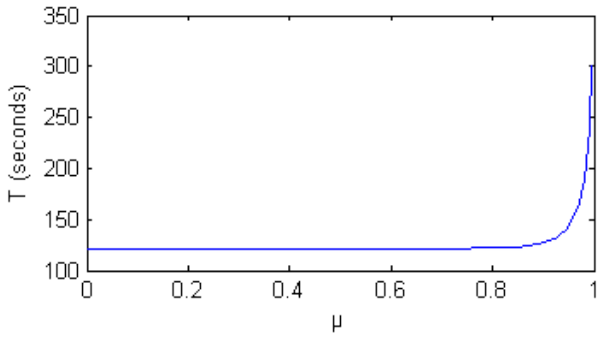

Fig. 4. $T-\mu$ Chart

(7), and (8) is also the optimal solution for the original run-curve optimization problem with objective function of (1) and constraints (2), (3), (4), (5), (6), (7), and (8) when the designated trip travel time requirement is T.

Proof: Assume a μ value named μ_1, and the optimal solution T_1, $E_1(T_1)$, and the run-curve profile in terms of $(x_1(t), u_1(t), v_1(t))$. If this solution is not optimal for the original run-curve optimization problem, then there exists one set of solution in terms of $(x_2(t), u_2(t), v_2(t))$ that leads to energy consumption $E_2(T_1) < E_1(T_1)$ with travel time T_1. Since the two optimization problem shares the same constraints, $(x_2(t), u_2(t), v_2(t))$ can lead to an objective function value $\mu_1 E_2(T_1) + (1-\mu_1)T_1$ that is better than $\mu_1 E_1(T_1) + (1-\mu_1)T_1$ using run-curve profile $(x_1(t), u_1(t), v_1(t))$, which violates the assumption at the beginning of the proof. Thus, the lemma is proved.

B. Solving Single Bi-criteria Optimization Problem

We follow the approximate dynamic programming framework in [12] to solve the single bi-criteria optimization problem. This dynamic program approach for the bi-criteria optimization problem discretizes the velocity and distance state space, which leads to much smaller memory consumption when compared with the dynamic programming discretizing over time by [6].

We apply Delaunay Triangulation on the state space of velocity and location. In Figure 3, the continuous state space is state space is discretized into triangles. The transition of states under action of acceleration is also illustrated in the figure. For a state s, if it is not grid point in the discretized state space, its value is barycentrically approximated using the nearest grid points' values. The backward dynamic programming can solve this problem using the following recursive function:

$$V(s_n) = \min_u [R(s_n, u) + \sum_{s'_{n+1} \in S} P(s'_{n+1}|s_n, u) V(s'_{n+1})] \tag{11}$$

In (11) , $V(s_n)$ is the optimal value of been in state s_n at location n. $R(s_n, u)$ is the reward of action u at state s_n. $P(s'_{n+1}|s_n, u)$ is the probability that system will at s'_{n+1} after applying action u at state s_n.

C. A Two-stage Procedure for Run-curve Optimization with Dynamic Travel Time

As it is shown in section III-B, we can solve the bi-criteria optimization problem time and time again while searching for appropriate μ value iteratively. The two most time consuming computation operations, which are transition matrix computation and μ value searching process, in the approximate dynamic programming procedure are not required to be processed for dynamic travel time requirements. We set up a framework of a two-stage procedure that is composite of an off-line stage and a real-time stage. The off-line stage can be executed any time when track profile and train information are available. The real-time stage is executed after the train receives a trip travel time requirement before departure. By transferring computation loads that do not require trip travel time requirement to off-line stage, the real-time stage can react quickly to different trip travel time requirements.

1) The Off-line Stage: In the off-line stage, the transition matrix is computed, and a set of bi-criteria optimization problems are solved with a number of μ values. As it is shown in Figure 4, there is a monotonic relation between T values and μ values, as μ value changes between 0 and 1. This can be proved as following:

Lemma 2: In the optimal solution of the bi-criteria optimization problem with objective function (10) and constraints (2), (3), (4), (5), (6), (7), and (8), T values have a monotonic relation with μ values. That is, if $\mu_1 \leq \mu_2$, then $T_1 \leq T_2$, $\forall \mu \in (0, 1)$.

Proof: If counter situation exists, we assume a pair of μ values, namely $\mu_1$and μ_2 such that the lemma is violated. Without lose of generality we assume: $\mu_1 \leq \mu_2$ and the corresponding T_1 and T_2 in the solutions of bi-criteria optimization problems have the relation of: $T_1 > T_2$. Let $E(T_1)$, T_1, and $(x_1(t), u_1(t), v_1(t))$ be the optimal solution for the bi-criteria optimization problem when $\mu = \mu_1$, and $E(T_2)$, T_2 and $(x_2(t), u_2(t), v_2(t))$ be the optimal solution for the bi-criteria optimization problem when $\mu = \mu_2$. Thus, according to the optimality of the solution, we have $\mu_1 E(T_1) + (1-\mu_1)T_1 \leq \mu_1 E(T_2) + (1-\mu_1)T_2$, which is equivalent to $T_1 - T_2 \leq \frac{\mu_1[E(T_2) - E(T_1)]}{1-\mu_1}$. According to

the assumption of the proof, we have $T_1 > T_2$. Considering lemma 1, we have $E(T_1) \leq E(T_2)$. Meanwhile, from the definition of μ such that $0 \leq \mu \leq 1$. Thus, we have $T_1 - T_2 \leq 0$, which conflict with the assumption. Thus, the lemma is proved.

Based on lemma 2, we are able to estimate μ values for designated travel time requirement T by interpolating from existing solutions of μ values and corresponding T requirements. Thus, in the off-line stage, two operations are performed:

- Compute and save the transition matrix for solving the bi-criteria optimization problem using approximate dynamic programming.
- Sample a number of μ values and solve the corresponding bi-criteria optimization problems for T values that could profile the $T - \mu$ chart with sufficient accuracy.

2) The Real-time Stage: The real-time stage takes place when a train is about to depart a terminal and the travel time requirement is finalized. Up to this point, the travel time requirement can not be changed and a run-curve is required for train operation. Based on the saved data from off-line stage, the transition matrix and a set of (T, μ) values are available. The on-board processor will perform two tasks:

- Interpolate from existing (T, μ) sample values for an appropriate μ' value for the designated travel time requirement T'.
- Solve the bi-criteria optimization problem with μ' using approximate dynamic programming.

Both of the two tasks in the real-time stage require much less computational efforts comparing with solving a normal dynamic programming approach from scratch. At the same time, the memory utilization is dramatically reduced, when compared with the dynamic programming using velocity and time as state space dimensions. Further, the framework gives electrified railroad the ability to fast react to different travel time requirements due to timetable fluctuations.

Since the newly obtained (T', μ') values should represent the trip travel time requirement in real world better than the sampled pair of (T, μ) values, the newly obtained (T', μ') values should be added to the sample sets of (T, μ) to help improves the accuracy for μ value interpolation for future trips.

A complete description on the algorithm, namely ADP-1, for generate optimal run-curve is described as following:

1) Off-line stage:
   a) Set up a factored MDP problem, where the state space is discretized on the dimension of velocity and distance and the transition matrix is computed;
   b) A set of sample μ values in $[0, 1]$ are generated such that there are more samples for μ near value 1;
   c) Corresponding travel time T are computed by solving the factored MDP using dynamic programming;
   d) (T, μ) values are stored in system memory;
2) Real-time stage:
   a) When a new travel time T' is received, a μ' value is interpolated from existing (T, μ) values that are pre-computed;
   b) If $|h(\mu') - T'| \leq \epsilon$, where ϵ is the tolerance on target travel time requirement and $h(\mu')$ represents the travel time obtained by interpolating existing (T, μ) values in the previous step, accept the run-curve generated with weight value μ', save μ' and corresponding travel time in system memory and stop, otherwise continue;
   c) A bi-secting searching process is executed using $(h(\mu'), \mu')$, and $(\tilde{T}, \tilde{\mu})$ as initial values, where $(\tilde{T}, \tilde{\mu})$ is a sample pair from system memory that has least $|\tilde{T} - T'|$ value and $(\tilde{T} - T')(h(\mu') - T') < 0$;
   d) The searching process updates μ' and $\tilde{\mu}$ until travel time requirement is satisfied.

D. A Two-stage Procedure for Optimal Run-curve with Dynamic Travel Time and Speed Limits

With dynamic travel time and speed limits before trains' departure, the optimal run-curve can still be obtained in a two-stage framework with off-line stage the same as in section III-C.1. However, the transition matrix and state values are changed due to the additional speed limits. At the same time, the $T - \mu$ relation obtained from pre-computing is also affected by the changes on speed limits, which makes it invalid for the direct interpolation in the real-time stage.

1) Update Transition Matrix: When speed limit is unchanged, the transition matrix obtained in the off-line stage can be re-used in the real-time stage. As additional speed limits are added to specific positions along the track, the transition matrix and state values used in the dynamic programming approach are also changed. To generate optimal run-curves using dynamic programming, the transition matrix needs to be updated due to the change on state space. As shown in Figure 5, a portion of the state space is directly eliminated by new speed limits as shown in the shaded area. A neighborhood area of the eliminated state space is affected as their capability to transit from or to the eliminated part of state space has changed. The neighborhood area can be identified using the physical model. By seting the action as acceleration, we can identify one boundary of the neighborhood with the new speed limit at location L_5. As it is shown in Figure 6, we can also get other boundaries of the neighborhood by setting the acttion as deceleration using speed limits at location L_1 and L_2. Thus, only a limited portion of the state space needs to be updated on their values and transition probabilities in the transition matrix. After the the updating process, the transition matrix would be able to generate run-curves under new speed limits.

2) Search for μ value: A function $h(\mu)$ is introduced to represent $T - \mu$ relation under original speed limits, such that $h(\mu) = T$ is the corresponding travel time by solving the bi-criteria optimization problem. As $h(\mu)$ does not represent the $T - \mu$ relation under updated speed limits due to additional

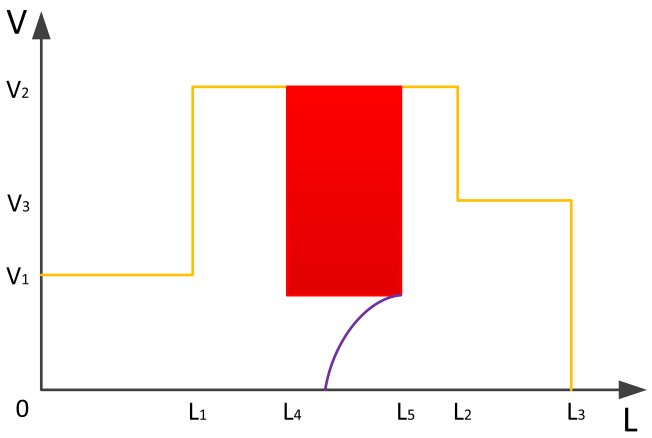

Fig. 5. Additional Speed Limits and Impacts on State Space

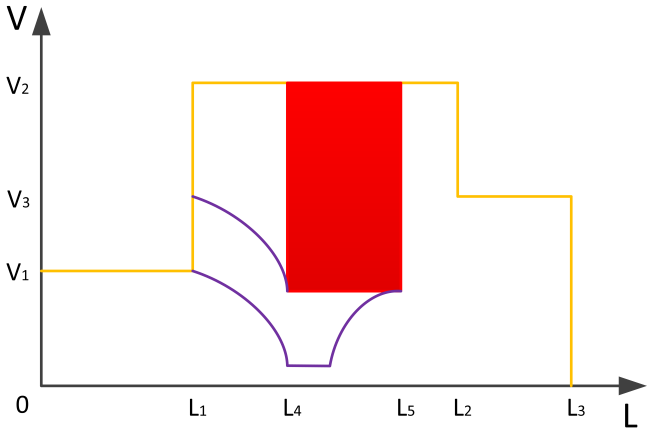

Fig. 6. Transition Matrix Updating

speed limits. Function $\tilde{h}(\mu)$ to represent the $T-\mu$ relation under updated speed limits, then there are followings Lemma hold:

Lemma 3: For a specific μ value, the travel time obtained from solving the optimization problem with additional speed limits is larger than travel time with original speed limits, in other words, $\tilde{h}(\mu) \geq h(\mu), 0 \leq \mu \leq 1$.

Proof: The proof of this Lemma is intuitive. If the speed limits added to the original speed limits is inactive for solving single core problem, then $\tilde{h}(\mu) = h(\mu)$. This situation happens when the new speed limits are redundant to the original speed limits, or the new speed limits is not violated when given travel time requirement is large enough, such that using same μ value under original speed limits would generate the same solution with the additional speed limits. In other words, $\exists \mu_4, 0 \leq \mu_4 \leq 1$, such that $\tilde{h}(\mu) - h(\mu) = 0, \forall \mu \geq \mu_4$. If the speed limits added to the original speed limits is active for solving single core problem, then we have $\tilde{h}(\mu) \geq h(\mu)$ as new constraints are added and feasible region for the updated problem is a subset of the feasible region for the problem with original speed limits.

A new function is defined as $T = h(\mu) + T_1 - T_0$ to represent a curve that is vertically moved by $T_1 - T_0$ from $T-\mu$ relation without additional speed limits. We define $\mu_1 = h^{-1}(T_2 - T_1 + T_0)$. From the definition of $T = h(\mu) + T_1 - T_0$ while considering $T_1 \geq T_0$ and monotonicity of $h(\mu)$, we would know $\mu_1 \leq \mu_3$.

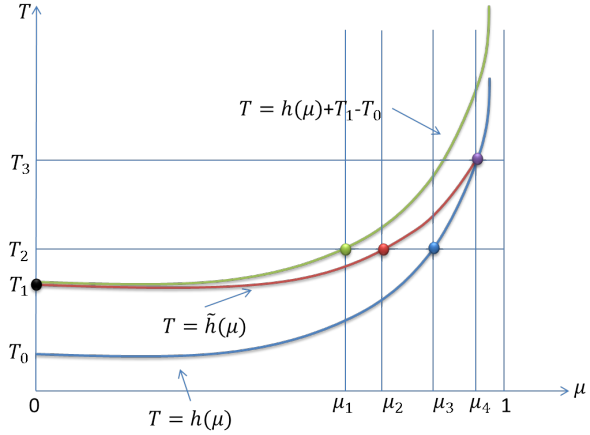

Fig. 7. Searching μ Value with Updated Speed Limits

As shown in Figure 7, $T = h(\mu) + T_1 - T_0$, $T = \tilde{h}(\mu)$, and $T = h(\mu)$ can form a geometric relation to bound μ_2 using 0, μ_1 and μ_3. μ_1 can be used as a first bisecting value for the searching process for μ_2 value.

A complete description on the algorithm for generating optimal run-curve, namely ADP-2, is described as following:

1) Off-line stage is the same as the off-line stage in section III-C.2. The $T-\mu$ relation is stated as a function $T = h(\mu)$;
2) Real-time stage:
   a) When a new travel time T_2 and updated speed limits are received, the transition matrix is updated for partial state space that would affect the run-curve generation;
   b) The fastest run-curve is generated with the updated transition matrix, and the travel time is T_1;
   c) Solve the bi-criteria optimization problem using μ_1, if $\tilde{h}(\mu_1) < T_2$, use $\mu_1 = h^{-1}(T_2 - T_1 + T_0)$ and $\mu_3 = h^{-1}(T_2)$ as initial guess for the searching process for μ_2; if $\tilde{h}(\mu_1) > T_2$, use 0 and $\mu_1 = h^{-1}(T_2 - T_1 + T_0)$ as initial guess for the searching process for μ_2; if $\tilde{h}(\mu_1) = T_2$ for acceptable tolerance, $\mu_2 = \mu_1$, stop.
   d) Update μ_2 value until travel time approaches T_2 within allowed tolerance.

3) Real-time Stage Acceleration: To further improve the reaction speed of the train to changing travel time and speed limit, a simple modification can be made to the algorithm in section III-D.2. In the real-time stage, after a fastest run-curve is obtained in stage 2(b), the run-curve is executed as the searching for μ_2 process continues. Every time a new μ_2 value is obtained and corresponding run-curve is obtained, the train start to use the newly generated run-curve.

	Baseline Algorithm			ADP-1		
Target Travel Time(s)	T(s)	Energy Cost	CPU Time(s)	T(s)	Energy Cost	CPU Time(s)
140	139.0	6.6756	12.4012	140.5	6.2369	1.0882
150	150.3	5.2462	13.6216	150.1	5.2723	1.0996
160	160.3	4.6496	16.5924	160.2	4.6562	1.1274
170	170.4	4.1689	13.8496	169.7	4.2126	1.1623
180	180.4	3.8291	20.0637	180.3	3.8290	1.2223
190	190.2	3.5515	14.0247	189.9	3.5684	1.2395
200	201.0	3.2839	16.5389	199.0	3.3169	1.2833
210	210.4	3.0762	19.4430	210.0	3.0838	1.37274
220	220.2	2.9110	21.0457	220.2	2.9110	1.3659

TABLE I

EXPERIMENTS ON DYNAMIC TRAVEL TIME

Travel Time(s)	Additional Speed Limit ([m, m], km/hr)	Baseline Algorithm			ADP-2			
		T(s)	Energy Cost	CPU Time(s)	T(s)	Energy Cost	CPU Time 1(s)	CPU Time 2(s)
180	[600, 800], 30	180.7	4.9396	21.3960	179.6	5.6199	1.8740	7.3003
180	[100, 200], 30	179.8	4.2094	17.8791	179.8	4.4232	1.4778	9.7978
180	[1600, 1700], 30	180.1	4.1582	17.3009	180.6	4.5238	1.5658	9.0595
210	[600, 800], 30	210.2	3.5745	20.7266	209.5	4.0371	1.7378	8.6795
210	[100, 200], 30	211.0	3.2243	19.4663	204.6	3.5730	1.4461	23.0340
210	[1600, 1700], 30	210.4	3.1026	17.7500	209.5	3.1934	1.6063	7.1235
240	[600, 800], 30	240.4	2.8594	20.4761	239.9	3.1703	1.7948	9.2486
240	[100, 200], 30	239.4	2.7074	18.1596	239.5	2.8082	1.4418	3.9594
240	[1600, 1700], 30	241.7	2.6133	24.9216	227.6	2.8422	1.6303	18.9569

TABLE II

EXPERIMENTS ON DYNAMIC TRAVEL TIME AND ADDITIONAL SPEED LIMITS

IV. EXPERIMENTS AND RESULTS ANALYSIS

A set of computational experiments are executed for testing the acceleration of this framework on a windows 7 PC with Intel i7 2.8GHz processor and 4 GB RAM using Matlab 7.11.0. The optimality of solutions for solving single factored MDP using approximate dynamic programming has been shown in [12]. We will show the computational speed reduction with this framework. We use the track and train profile in [12].

The first set of experiments is on generating optimal run-curve when there are only changes on travel time. As shown in Table I, the first column is the target travel time requirements ranging from 140 seconds to 220 seconds. The baseline algorithm used in this paper is an approximate dynamic programming algorithm from [12]. The results of the baseline algorithm are in columns 2 to 4. Column 2 is the travel time for the run-curve obtained by the algorithm. Energy cost column stores the energy for the run-curve obtained by the algorithm. CPU time is the computational time for generating the run-curve solutions. The results for the two stage algorithm for optimal run-curve with dynamic travel time are in columns 5 to 7. As shown in the table, ADP-1 can reduce the computation time to less than 2 seconds, which are average 92.4% reduction on computation time.

The second set of experiments is on testing ADP-2 under changing travel time and changing speed limits. The additional speed limit of 30km/h is added near beginning of the track from 100 to 200 meters, or in the middle of the track from 600 to 800 meters, or near the end the track from 1600 to 1700 meters. The travel time requirements are 180, 210 and 240 seconds. In Table II, the first two columns are travel time requirements and additional speed limits information. Column 4 to 6 are baseline algorithm's results on travel time, energy cost and CPU time needed for the run-curves obtained. According to the structure of ADP-2, there are two computation time that is of interest. The first one, denoted as "CPU Time 1" in column 8 of Table II, is the time needed before a train reacts to updated travel time and additional speed limits information. The second one, denoted as "CPU Time 2" in column 8 of Table II, is the computation time for ADP-2 finally converges after a number of searching iterations. In real implementation, trains only need to wait for the time listed in column "CPU time 1" before an initial run curves is generated for the updated speed limit profile. While trains are accelerating after departing a station, ADP-2 will keep updating run-curve until convergence to a final run-curve. The final convergence time "CPU Time 2" may vary due to different discretization schema and numerical convergence criteria. As we can see from column "CPU time 1" the table, ADP-2 reduces the reaction time for trains to less than 2 seconds. At the same time, we also observe an average extra cost of energy of 8.6%.

V. CONCLUSION AND FUTURE WORK

A framework for computing optimal run-curves with dynamic travel time is proposed in this paper. By introducing a two stage procedure, significant amount of computation load are completed off-line while real-time stage is adaptive to changing travel time requirement and speed limits. The computational results show significant CPU time reduction is

achieved when using the framework proposed in this paper. Hence, it achieves near optimal run-curve generation in a short enough time for on-board processors such as ATO. As a future research direction, changing train weight and length should also be considered in optimal run-curve generation procedure.

REFERENCES

[1] Y. V. Bocharnikov, A. M. Tobias, C. Roberts, S. Hillmansen, and C. J. Goodman. Optimal driving strategy for traction energy saving on dc suburban railways. *Electric Power Applications, IET*, 1(5):675–682, 2007.

[2] C. S. Chang, W. Wang, A. C. Liew, F. S. Wen, and D. Srinivasan. Genetic algorithm based bicriterion optimisation for traction substations in dc railway system. In *IEEE International Conference on Evolutionary Computation*, 1995.

[3] J. Cheng and P. G. Howlett. Application of critical velocities to the minimization in the control of trains. *Automatica*, 28(1):165–169, 1992.

[4] Y. Ding, H. Liu, Y. Bai, and F. Zhou. A two-level optimization model and algorithm for energy-efficient urban train operation. *Journal of Transportation Systems Engineering and Information Technology*, 11(1):96–101, 2011.

[5] P. G. Howlett and J. Cheng. Optimal driving strategies for a train on a track with continuously varying gradient. *Journal of the Australian Mathematical Society. Series B. Applied Mathematics*, 38(3):388–410, 1997.

[6] H. Ko, T. Koseki, and M. Miyatake. Numerical study on dynamic programming applied to optimization of running profile of a train. *IEEJ Transactions on Industry Applications*, 125(12):1084–1092, 2006.

[7] M. Kuriyama, S. Yamamoto, and M. Miyatake. Theoretical study on eco-driving technique for an electric vehicle with dynamic programming. In *International Conference on Electrical Machines and Systems (ICEMS)*, 2010.

[8] L. Li, W. Dong, Y. Ji, and Z. Zhang. An optimal driving strategy for high-speed electric train. In *2011 30th Chinese Control Conference*, pages 5899–5904, 2011.

[9] R. Liu and I. M. Golovitcher. Energy-efficient operation of rail vehicles. *Transportation Research Part A: Policy and Practice*, 37(10):917–932, 2003.

[10] M. Miyatake, H. Haga, and S. Suzuki. Optimal speed control of a train with on-board energy storage for minimum energy consumption in catenary free operation. In *13th European Conference on Power Electronics and Applications*, 2009.

[11] M. Miyatake and H. Ko. Optimization of train speed profile for minimum energy consumption. *IEEJ Transactions on Electrical and Electronic Engineering*, 5(3):263–269, 2010.

[12] D. Nikovski, W. Zhang, and B. Lidicky. Markov decision processes for train run curve optimization. In *International Conference on Electrical Systems For Aircraft, Railway, and Ship Propulsion*, 2012.

[13] T. Ogawa, H. Yoshihara, S. Wakao, K. Kondo, and M. Kondo. Design estimation of the hybrid power source railway vehicle based on the multiobjective optimization by the dynamic programming. *IEEJ Transaction on electrical and electronic engineering*, 3(1):48–55, 2008.

[14] P. J. Pudney and P. G. Howlett. Optimal driving strategy for a train journey with speed limits. *Journal of the Australian Mathematical Society. Series B. Applied Mathematics*, 36(1):38–49, 1994.

[15] Y. Wang, B. Ning, F. Cao, B. De Schutter, and T. J. J. van den Boom. A survey on optimal trajectory planning for train operations. In *IEEE International Conference on Service Operations, Logistics, and Informatics(SOLI)*, 2011.

[16] K. K. Wong and T. K. Ho. Coast control for mass rapid transit railways with searching methods. In *IEE Proceedings on Electric Power Applications*, 2004.

[17] J. Xu and D. Nikovski. Method for determining run-curves for vehicles based on travel time. In *US Patent 8,838,304*, 2014.

[18] J. Xu and D. Nikovski. Method for determining run-curves for vehicles in real-time subject to dynamic travel time and speed limit constraint. In *US Patent 8,660,723*, 2014.