Proceedings of the Sixth International Conference on
Railway Technology: Research, Development and Maintenance
Edited by: J. Pombo
Civil-Comp Conferences, Volume 7, Paper 10.4
Civil-Comp Press, Edinburgh, United Kingdom, 2024
ISSN: 2753-3239, doi: 10.4203/ccc.7.10.4

Simultaneous Optimisation of Energy-Efficient Speed Profiles of Two Trains by Parallel Dynamic Programming

K. Sakai, W. Ohnishi and T. Koseki

**Department of Electrical Engineering and Information Systems,
The University of Tokyo, Japan**

Abstract

Energy-saving methods for electric railways by modifying driving methods have been proposed for carbon neutrality. In a direct current electric railway network, regenerative energy from a braking train needs to be consumed by another powering train to avoid wasting kinetic energy. The aim of this paper is to propose a dynamic programming method with parallel computing that solves large-scale optimisation problems for the speed profiles of two trains considering regenerative energy. The numerical evaluation results show that simultaneous optimisation can reduce total energy consumption by using regenerative energy. They suggest that it is effective to optimise the train receiving regenerative energy. The advantage of dynamic programming is that its solution follows causality even when a train operation is disturbed.

Keywords: direct current electric railway, energy-efficient driving, regenerative braking energy, speed profile, dynamic programming, parallel computing.

1 Introduction

Reducing driving energy on electric railways is necessary to realise a decarbonised society. Carbon dioxide emissions from electric train driving in Japan are about 7

million tonnes per year and account for more than 70% of the total emissions from railway operators [1].

There are two approaches to energy-saving: the hardware approach and the software approach. The former approach uses installing power storage devices, for example. The latter approach uses devising driving operations. It is known that driving energy can be reduced by increasing the coasting time without using motors or brakes [2,3]. Many methods have been proposed for optimising the train speed profile that determines the driving operation between stations. Dynamic programming is one of the typical methods to solve optimisation problems for energy-saving. It supports complex models [4] such as speed limit including signals [5,6], and nonlinear running resistance and motor efficiency [7]. Its performance for single-train optimisation is better than the genetic algorithm and ant colony optimisation [8].

However, dynamic programming has a problem in that a large amount of calculation costs is needed to improve the solution quality. It also takes huge calculation costs to utilise regenerative energy between multiple trains for further energy saving. Simultaneous optimisation of two trains by dynamic programming is technically challenging, and previous research on two-train cooperative optimisation used other optimisation methods [9, 10]. Energy-saving optimisation problems can be solved rapidly by applying parallel computing to the dynamic programming method [11, 12], and it is thought that it will be possible to solve large-scale problems in a practical time.

This paper focuses on dynamic programming and aims to solve a large-scale problem to optimise the speed profiles of two trains simultaneously to minimise the total energy consumption. Simultaneous optimisation problems of two trains have a four-dimensional state space considering regenerative energy, and we propose a parallel computing method to be solved in a practical time. We numerically verify the effectiveness of simultaneous optimisation based on the case study of two trains considering regenerative energy transfer. The numerical evaluation is performed for different departure delay times because the timing to transfer regenerative energy may be lost due to the departure delay of either train.

2 Mathematical optimisation of the energy-efficient train speed profile

2.1 Principles of speed profile optimisation by dynamic programming [4]

The optimisation problem of the energy-efficient train speed profile is usually formulated as a problem to find the operation curve $x(t)$ that minimises energy consumption E for each stop as (1)–(4) below. There are some constraints: (2) the position and speed at the beginning and end of the speed profile, (3) the maximum speed for each

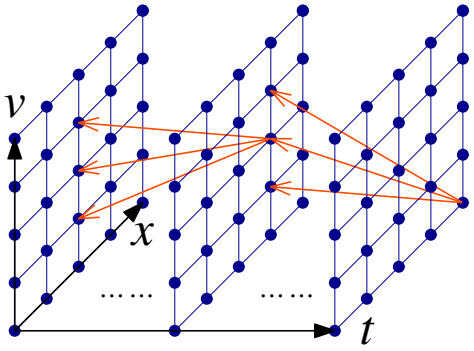


Figure 1: Principle of initial searches in Dynamic Programming for single-train optimisation.

section, and (4) the maximum value of the acceleration and deceleration.

$$\text{minimise} \qquad E \tag{1}$$

$$\text{subject to} \qquad x(0) = 0, v(0) = 0, x(T) = L, v(T) = 0 \tag{2}$$

$$0 \leq v(t) \leq v_{max} \tag{3}$$

$$-\beta - R(v) \leq \frac{\mathrm{d}v}{\mathrm{d}t} \leq \alpha - R(v), \tag{4}$$

where T is the travelling time between the two stops, L is the distance between the two stops, α and β are the maximum value of the acceleration and deceleration of the vehicle, respectively, and R is the running resistance.

The train speed profile $x(t)$ is calculated by the motion equation (5), and the consumption energy E is obtained by integrating the instantaneous power P (6).

$$\frac{\mathrm{d}v}{\mathrm{d}t} = a(t) - R(v), \ \frac{\mathrm{d}x}{\mathrm{d}t} = v(t) \tag{5}$$

$$E = \int_0^T P(a, v)\mathrm{d}t, \tag{6}$$

where a is the train acceleration equivalent to the control input.

In the dynamic programming method, the minimum accumulative energy is evaluated based on the principle of optimality of a multi-staged optimisation problem. The optimal speed profile is determined regardless of the driving operations before the beginning of the speed profile to optimise.

Dynamic programming approximates optimisation problems by discretising time and space. It discretises the time axis t and divides the state space consisting of train position x and speed v into a grid, and N phase planes are obtained, as shown in Figure 1.

Each lattice point on the grid has accumulative energy consumption as an evaluation value. It is the minimum required energy for the train from the time and state at the lattice point to the final state at the end time T. An accumulative energy consumption at a lattice point $(x(k\,\Delta t), v(k\,\Delta t))$ at the time $k\,\Delta t$ is calculated as the sum of

the evaluation value of a lattice point $(x((k+1)\Delta t), v((k+1)\Delta t))$ at the next time $(k+1)\Delta t$ and the energy increment during the time Δt. The evaluation value is defined as the minimum accumulative energy consumption by changing the control input u between the two lattice points $(x(k\,\Delta t), v(k\,\Delta t))$ and $(x((k+1)\Delta t), v((k+1)\Delta t))$. The lattice points are evaluated retroactively in the order of the final time $T, T-\Delta t, \ldots$. After completing the evaluations, the optimal control inputs $u(k\,\Delta t)$ and the speed profile $x(t)$ can be obtained from the starting time $0, \Delta t, \ldots$.

The boundary condition of the final state is added to the objective function value (7) and treated as a penalty.

$$\phi(x(T), v(T)) = c_x(x(T) - L)^2 + c_v\, v(T)^2, \tag{7}$$

where c_x and c_v are the penalty factors. This penalty value is the energy evaluation value of stage N.

2.2 Application of parallel computing

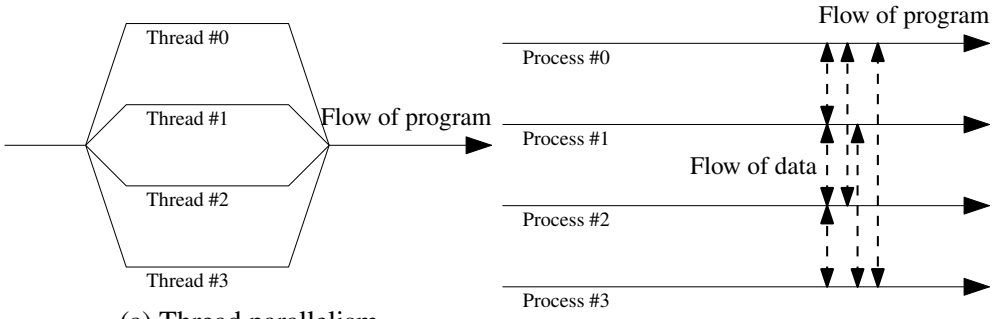
Optimising the speed profile by the dynamic programming method is suitable for parallel computing. That is because there is data dependence in the temporal direction, but there is no data dependence in the spatial direction. Therefore, the calculation at each lattice point does not require data on other lattice points at the same time t to calculate the lattice points simultaneously.

First, the solutions of the difference equation discretised from the differential equation (5) in the literature [4] are memorised in parallel. The reference lattice point locations and interpolation coefficients are stored in memory for each lattice point and control input pair. Next, the energy values of each lattice point are evaluated in parallel. Steps 1 and 2 are repeated in the backward direction of the time axis. Finally, the optimal path is searched in the forward direction of the time axis. Optimal paths for multiple delay cases can be searched in parallel in Step 3.

2.3 Sliding the confined state space to calculate

In this paper, the state space to evaluate the energy in the backward direction of the time axis is confined depending on the time using the results of single-train optimisation. Some regions of the state space where trains cannot exist for each time step are excluded from the calculation. It is effective in shortening the calculation time of dynamic programming [4]. It will be more effective for simultaneous optimisation of two trains particularly because it requires a four-dimensional state space. However, there are few effective and general-purpose methods for confinement because assuming a region where trains could exist requires experience.

In this paper, we propose a method that calculates only the periphery of the state in the optimal solution of a single train for each time. That is because the optimal solution of simultaneous optimisation of two trains is not likely to be far from the



(a) Thread parallelism (b) Process parallelism

Figure 2: Concept of two parallel computing methods.

combination of two optimal solutions of single-train optimisation. In the numerical calculation of this paper, the state space is confined depending on x_w [m] before and after the position in the optimal solution, but not depending on speed. That is because there is a trade-off relationship between computer memory usage and calculation time in the memorisation of the solution of the differential equation in calculation step 1. If the same process in parallel computing handles lattice points in the same speed region, solutions of the differential equation are likely to be reusable.

3 Methods of parallel computing

There are several parallel computing methods used in the railway field [13], and in this paper, thread parallelism by OpenMP [14] and process parallelism by Message Passing Interface (MPI) [15] are used together.

3.1 Thread parallelism (OpenMP)

It is a method of simultaneously launching multiple threads, which are execution units, as shown in Figure 2(a). Parallel calculation is possible by running multiple cores with different threads. Parts of the program can be parallelised, and memory is shared between threads. OpenMP is an application programming interface (API) to realise thread parallelism.

3.2 Process parallelism (MPI)

It is a method of simultaneously launching multiple processes, which are program units, as shown in Figure 2(b). Memory is not shared between processes, and data is distributed in each process. It can handle data so large that it does not fit in one computer, but the communication time to synchronise data with inter-process communication can be a bottleneck. MPI is the standard for inter-process communication.

	Train 1	Train 2
Case 1	Proposed simultaneous optimisation for two trains	
Case 2	Single-train optimisation	Separate optimisation for single train
Case 3	Single-train optimisation	Single-train optimisation based on the trajectory of Train 1

Table 1: Cases of numerical calculations.

Maximum acceleration α	$3.3\,\mathrm{km/h/s}$ $(0.9167\,\mathrm{m/s^2})$
Maximum deceleration β	$3.5\,\mathrm{km/h/s}$ $(0.9722\,\mathrm{m/s^2})$
Mass	$295\,\mathrm{t}$
Travelling time for each train T	$100\,\mathrm{s}$
Departure interval of two trains	$50\,\mathrm{s}$
Distance between stations L	$1000\,\mathrm{m}$
Maximum speed v_{max}	$65\,\mathrm{km/h}$

Table 2: Train specifications and line conditions.

4 Numerical evaluation of simultaneous optimisation

Numerical calculations are performed to compare the energy-saving effect of the simultaneous optimisation method for two trains with the conventional method. Performed numerical calculations are shown in Table 1. Case 1 optimises two trains simultaneously considering regenerative energy transfer, Case 2 combines the two optimisation results of each train without considering regenerative energy transfer, and Case 3 optimises the second train considering regenerative energy from the first train optimised as a single train.

4.1 Calculation conditions

Train specifications and route conditions are configured for a DC-electrified metro line as shown in Table 2. The running resistance $R\,[\mathrm{km/h/s}]$ depends on the speed $v\,[\mathrm{km/h}]$ and is defined in the following equation (8).

$$R = 0.00001v^2 + 0.0002v + 0.056 \tag{8}$$

The available control inputs consist five levels of 1, 0.5, 0, -0.5, and -1. 1 is the maximum acceleration, 0 is coasting, and -1 is the maximum deceleration. Their powering efficiency is 1.0, and regenerating efficiency is 0.5. Running energy is calculated by integrating the difference between the mechanical power required for acceleration and the regenerated power while the powering load of the other train is present without considering the electrical circuit. Gradients and curves are ignored.

The time division step is fixed value $1\,\mathrm{s}$. The position division step varies depending on the region of the state space and the number of trains to optimise as shown in Figure

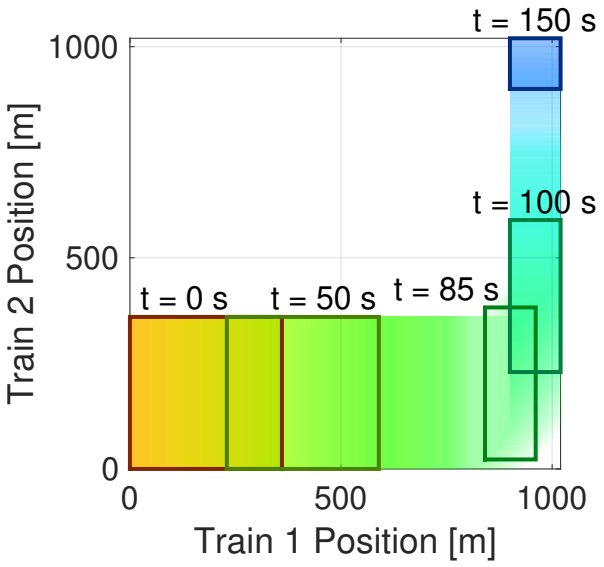


Figure 3: The sliding state space to calculate.

CPU	Fujitsu A64FX (48 Cores, 2.2 GHz)
RAM	32 GB per node
Compiler	Fujitsu C++ Compiler (Clang mode)

Table 3: Supercomputer performance.

3. For two trains, assuming the departure point is $0\,\mathrm{m}$, it is divided every $1.5\,\mathrm{m}$ from $0\,\mathrm{m}$ to $750\,\mathrm{m}$ between stations, and every $0.5\,\mathrm{m}$ from $750\,\mathrm{m}$ to $1020\,\mathrm{m}$ near the arrival station. The number of points to calculate on the each position axis is 241. For a single train, the position is divided every $0.125\,\mathrm{m}$. The speed division step is $1\,\mathrm{km/h}$ for two trains, and $0.125\,\mathrm{km/h}$ for a single train. It slides based on the single-train optimisation and x_w is configured to support up to $20\,\mathrm{s}$ delay.

The optimisation problem was calculated by The University of Tokyo Information Technology Centre's supercomputer Wisteria/BDEC-01 Odyssey with the performance shown in Table 3. The number of used nodes increases according to the problem size. Four processes per node and twelve threads per process are launched.

4.2 Comparison with single-train separate optimisation

Figure 4 shows the speed profiles and energy consumption optimised by the simultaneous optimisation for two trains, and Figure 5 also shows the speed profiles and energy consumption optimised by the separate optimisation for a single train. Figures on the same column have the same delay time of Train 1, and figures on the same row have the same delay time of Train 2. The top left figure is the case where there is no delay.

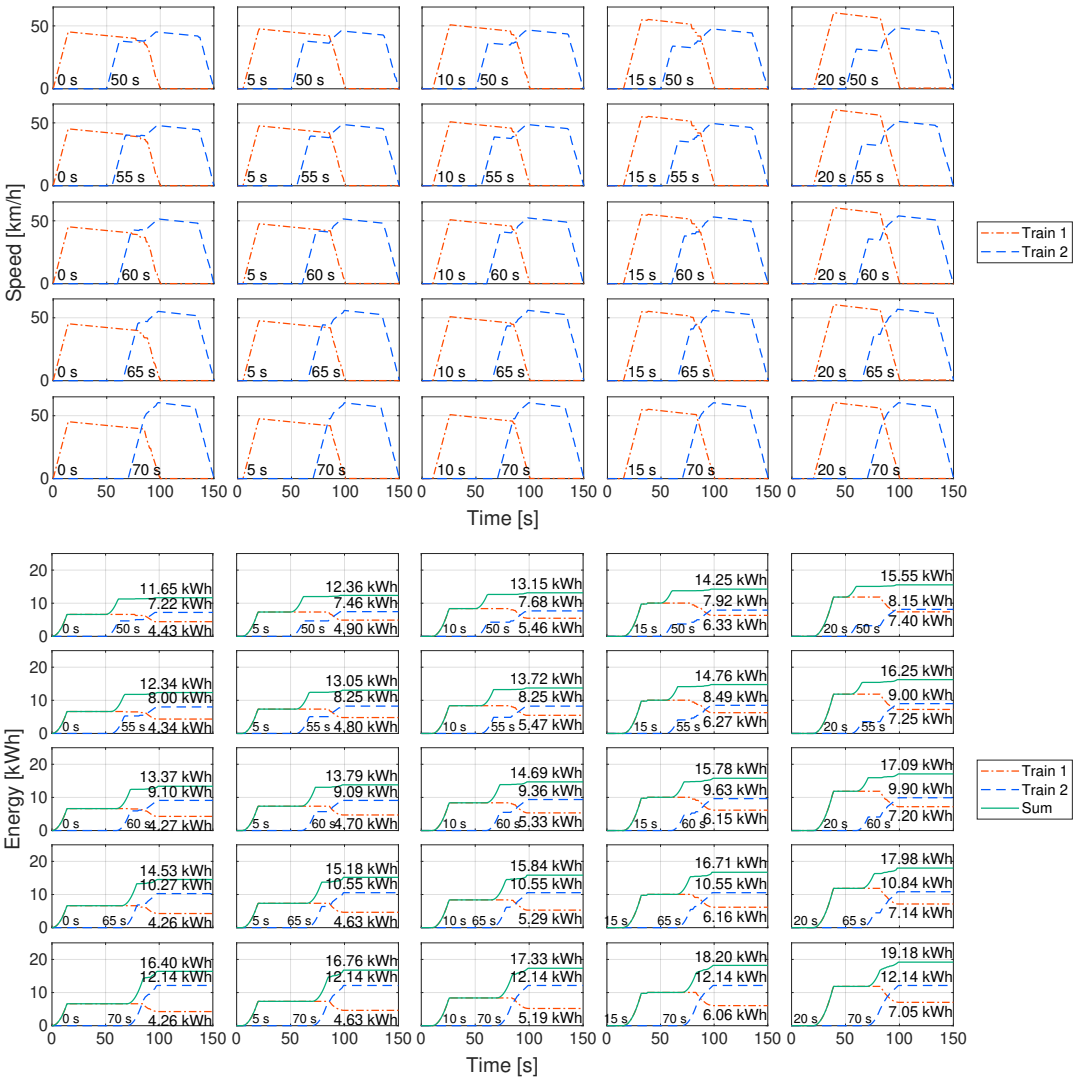


Figure 4: Optimisation results of Case 1. Case 1 optimises two trains simultaneously considering regenerative energy transfer.

Simultaneous optimisation of Case 1 shown in Figure 4 reduces the total energy consumption compared to separate optimisation of Case 2 shown in Figure 5. The energy consumption of Train 2 in Case 1 is larger than that in Case 2 for all delay times. However, in Case 1, the increase of the acceleration energy of Train 2 is provided by the regenerative energy of Train 1. It is important to consider regenerative energy in net energy optimisation.

Even when a departure delay occurs on either or both, the regenerated energy is transferred at the appropriate timing in the simultaneous optimisation of Case 1. On the other hand, regenerated energy transfer happens by chance in separate optimisation of Case 2. The advantage of simultaneous optimisation is that the timing when Train 2 accelerates and the timing when Train 1 decelerates can be matched.

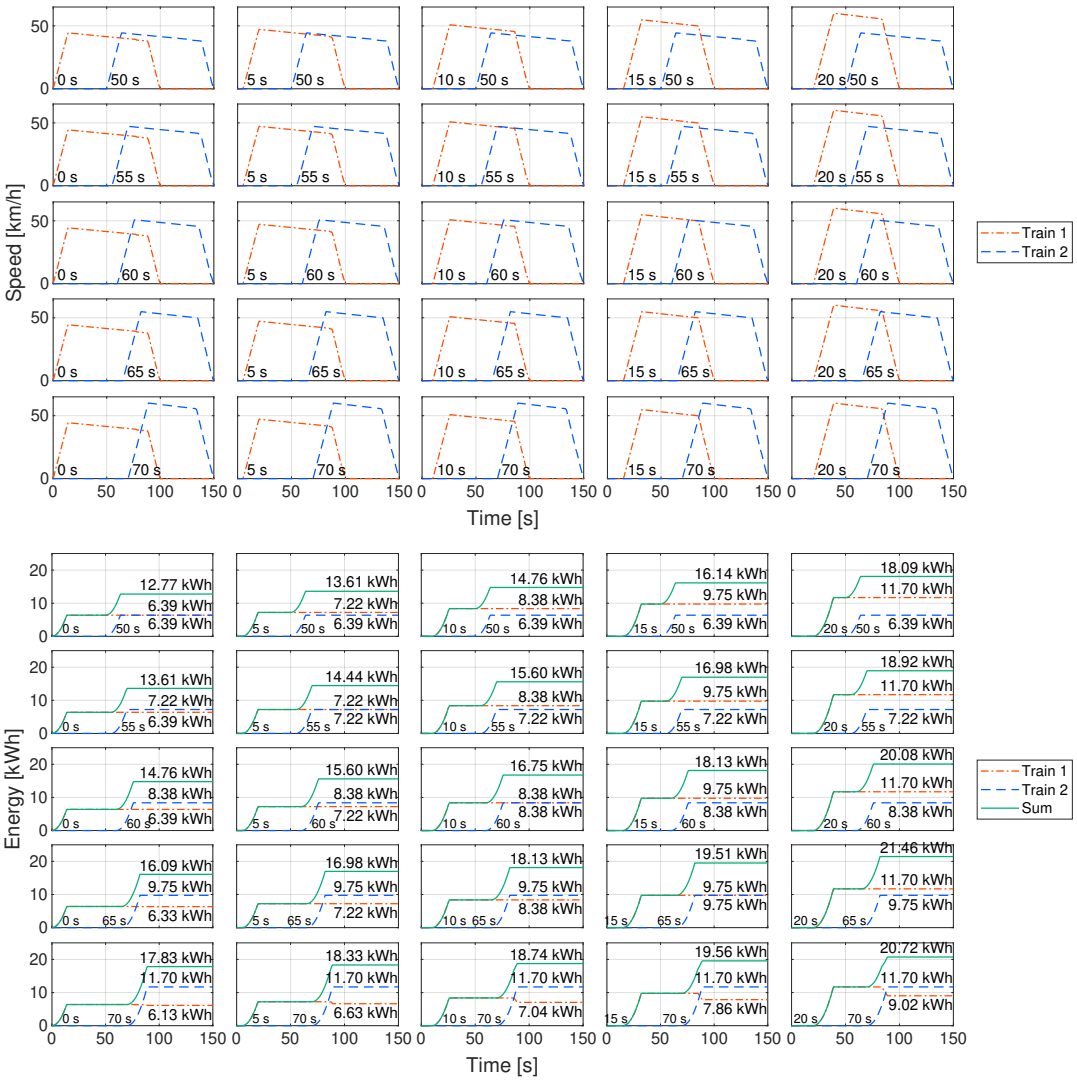


Figure 5: Optimisation results of Case 2. Case 2 combines the two optimisation results of each train without considering regenerative energy transfer.

The speed profiles optimised by dynamic programming follow causality in delayed cases. The energy evaluation value from $50\,\mathrm{s}$ to $100\,\mathrm{s}$ assumes that both trains 1 and 2 are running at that time. If a departure delay of the following train can be foreseen, the optimisation results will be more energy-efficient. However, the results in Figure 4 are optimal without the foresight of delays that have not yet occurred.

4.3 Comparison with single-train optimisation from the preceding to the following

Next, we compare the proposed method of Case 1 with Case 3, single-train optimisation from the preceding to the following. In Case 3, the speed profile of Train 1 is

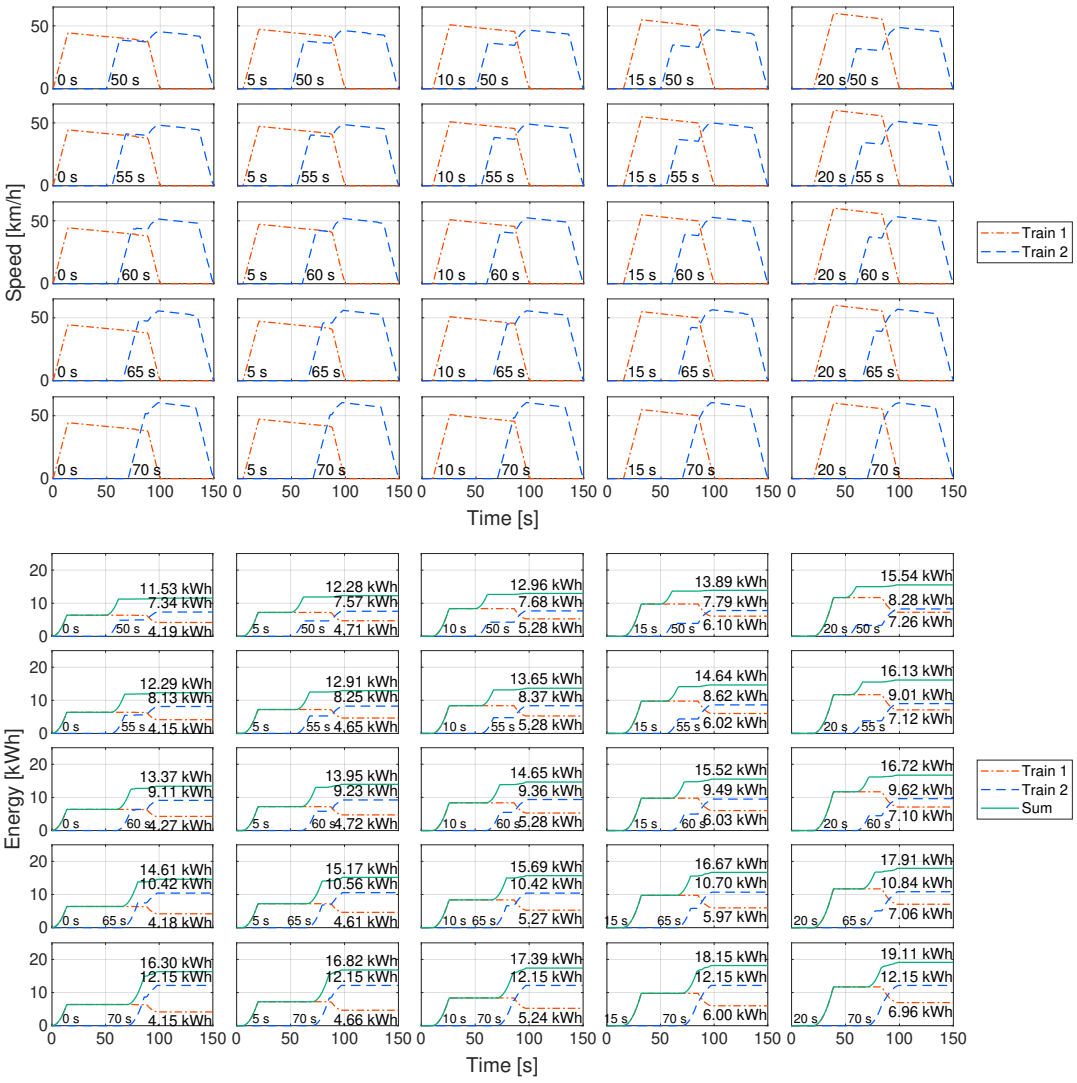


Figure 6: Optimisation results of Case 3. Case 3 optimises the second train considering regenerative energy from the first train optimised as a single train.

optimised independently without considering regenerative energy, and then, the speed profile of Train 2 is optimised considering regenerative energy transfer with Train 1. Figure 6 shows the optimised speed profiles and energy consumption of Case 3.

The total energy in Case 3 is slightly smaller than that in Case 1 in many delay cases even though only the train receiving regenerative energy considers regenerative energy transfer. That is because the division of the state space in Case 3 is finer. The calculation time in Case 1 was $255\,\mathrm{s}$ by using 24 nodes, and that in Case 3 was $243\,\mathrm{s}$ by using 6 nodes. Simultaneous optimisation has a four-dimensional state space and needs more calculation costs to divide the state space finely.

These results suggest that it is effective to optimise only the train receiving regenerative energy. However, the simultaneous optimisation method of two trains is

meaningful in that the importance of adjusting the speed profile of the receiver train was clarified by global optimisation.

5 Conclusions

The dynamic programming method used for energy-efficient optimisation of train speed profiles requires a lot of computation costs to find a high-quality solution. Parallel computing can accelerate the computation speed. This paper proposes a parallel computing method to solve larger optimisation problems. The larger problem needs more computer memory usage, but the memory usage can be reduced by confining the state space based on the speed profile optimised for the single train.

In a case study of two DC metro trains considering regenerative energy, it was verified that simultaneous optimisation for multiple trains can reduce the total energy consumption by utilising regenerative energy. However, when the departure of either train is delayed, there is a possibility that the regenerative energy cannot be transferred as planned. Simultaneous optimisation by dynamic programming can find the optimal speed profiles for multiple delay cases in parallel. The speed profiles optimised by dynamic programming follow causality in delayed cases and it has a practical advantage in that it does not need departure prediction. The result also suggests that it is effective to optimise only the train receiving regenerative energy.

Acknowledgement

This research was conducted using the FUJITSU Supercomputer PRIMEHPC FX1000 and FUJITSU Server PRIMERGY GX2570 (Wisteria/BDEC-01) at the Information Technology Center, The University of Tokyo.

References

[1] Ministry of Land, Infrastructure, Transport and Tourism, "Carbon neutrality acceleration in the railways field study group (1st document)." `https://www.mlit.go.jp/tetudo/content/001474317.pdf`. (in Japanese)

[2] Y. Kimura, S. Koga, "Energy saving operation of a frequently stopped commuters' train," Transactions of the Society of Instrument and Control Engineers, 20, 357–360, 1984, DOI: 10.9746/sicetr1965.20.357 (in Japanese)

[3] P. Howlett, I. Milroy, P. Pudney, "Energy-efficient train control," Control Engineering Practice, 2, 193–200, 1994, DOI: 10.1016/0967-0661(94)90198-8

[4] H. Ko, T. Koseki, M. Miyatake, "Application of dynamic programming to the optimization of the running profile of a train," WIT Transactions on The Built Environment, 74, 103–112, 2004, DOI: 10.2495/CR040111

[5] N. Oba, M. Miyatake, "Design of energy-efficient train speed profiles considering fixed-block signaling system," Electrical Engineering in Japan, 205, 26–35, 2018, DOI: 10.1002/eej.23114

[6] S. Ichikawa, M. Miyatake, "Energy Efficient Train Trajectory in the Railway System with Moving Block Signaling Scheme," IEEJ Journal of Industry Applications, 8, 586–591, 2019, DOI: 10.1541/ieejjia.8.586

[7] S. Watanabe, T. Koseki, Y. Noda, M. Miyatake, "Optimized energy-saving speed profile in linear-motor railway system," Electrical Engineering in Japan, 202, 22–32, 2018, DOI: 10.1002/eej.23041

[8] S. Lu, S. Hillmansen, T. K. Ho, C. Roberts, "Single-Train Trajectory Optimization," IEEE Transactions on Intelligent Transportation Systems, 14, 743–750, 2013, DOI: 10.1109/TITS.2012.2234118

[9] Y. Huang, H. Yu, J. Yin, H. Hu, S. Bai, X. Meng, M. Wang, "An integrated approach for the energy-efficient driving strategy optimization of multiple trains by considering regenerative braking," Computers and Industrial Engineering, 126, 399–409, 2018, DOI: 10.1016/j.cie.2018.09.041

[10] M. Chen, X. Feng, Q. Wang, P. Sun, "Cooperative Eco-Driving of Multi-Train Under dc Traction Network," IEEE Transactions on Transportation Electrification, 7, 1805–1821, 2021, DOI: 10.1109/TTE.2021.3059433

[11] G. Matsuura, M. Miyatake, "Optimal train speed profiles by dynamic programming with parallel computing and the fine-tuning of mesh," WIT Transactions on the Built Environment, 135, 767–777, 2014, DOI: 10.2495/CR140641

[12] K. Sakai, W. Ohnishi, T. Koseki, "Verification of fast calculation of energy-saving speed profile by parallel dynamic programming," in 29th Jointed Railway Technology Symposium (J-RAIL2022), 322–325, 2022. (in Japanese)

[13] Q. Wu, M. Spiryagin, C. Cole, T. McSweeney, "Parallel computing in railway research," International Journal of Rail Transportation, 8, 111–134, 2020, DOI: 10.1080/23248378.2018.1553115

[14] `https://www.openmp.org/`.

[15] `https://www.mpi-forum.org/`.