# Simulation-Based Hybrid Model for a Partially-Automatic Dispatching of Railway Operation

## (Simulationsbasiertes Hybrid Modell für eine teilautomatisierte Disposition des Eisenbahnbetriebs)

Von der Fakultät Bau- und Umweltingenieurwissenschaften
der Universität Stuttgart zur Erlangung der Würde eines
Doktors der Ingenieurwissenschaften (Dr.-Ing.) genehmigte Abhandlung

vorgelegt von


**Yong Cui**

aus Shitai, China


| | |
|---|---|
| Hauptberichter: | Prof. Dr.-Ing. Ullrich Martin |
| Mitberichter: | Prof. Dr.-Ing. Jörn Pachl |
| Tag der mündlichen Prüfung: | 07.12.2009 |


Institut für Eisenbahn- und Verkehrswesen der Universität Stuttgart

2010

# Foreword

The use of software-based simulation models for infrastructure dimensioning and scheduling has become common practice within the last few years due to the enormous increase in the performance capabilities of both computer hardware and software.

However, the practical application of these models had been restricted due to the lack of:

- theoretical approaches regarding the advantageous, integrated interaction of asynchronous and synchronous simulation
- consideration to the specific requirements of shunting movements
- real-world-applicable solutions for avoiding deadlocks

One of the most important results of the research in this thesis is the design of a comprehensive, hybrid simulation model based on the current knowledge of the synchronous and asynchronous simulation of railway operational processes that can be used for the dimensioning of infrastructures, scheduling, as well as during the automated dispatching of railway operation.

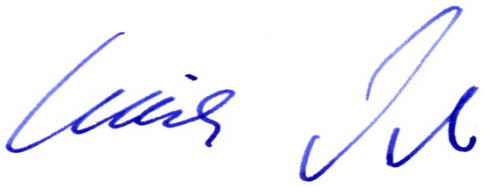
Until now, there had been no solutions to the problem of the gaps existing between the macroscopic and microscopic models.

With this model, a standardised application for the simulation of railway operations in applications ranging from extensive networks to shunting services is now possible. This is an important prerequisite for the implementation of automatic train and shunting dispatching solutions.

The improved capability of the modelling and the further automation of the extensive network simulation at different levels are particularly commendable innovations achieved in this thesis.

Furthermore, the consideration given to specific issues, the detailed discussion of topic-related facts, the objective evaluation of individual influence factors, as well as the outstanding methodical combination of the components of the model itself are equally commendable.

This thesis represents a significant advance in the state of knowledge in the scientific field of transportation modelling and can furthermore be used in other comparable fields where the subject of deadlock avoidance requires special attention.



Dr.-Ing. Ullrich Martin

Universitätsprofessor

# Vorwort

Die Anwendung von softwaregestützten Simulationsmodellen zur Dimensionierung von Infrastrukturen und zur Fahrplanerstellung wurde in den letzten Jahren durch die erheblich verbesserten rechentechnischen Möglichkeiten zum Standard. Allerdings schränken die hier eingesetzten Modelle aufgrund fehlender

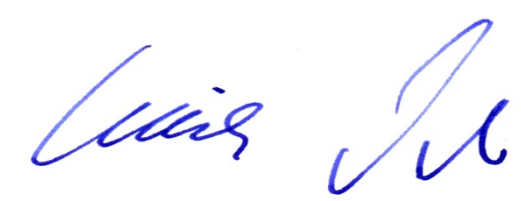
- modelltheoretischer Ansätze zum vorteilhaften Zusammenspiel asynchroner und synchroner Simulationslösungen für deren integratives Zusammenwirken,
- Integration der besonderen Bedingungen des Rangierdienstes sowie
- praxisorientierter Lösungen zum Vermeiden von Deadlocks

den praktischen Einsatz ein oder erschweren diesen signifikant.

Wesentliches Forschungsergebnis der vorliegenden Arbeit ist der Entwurf eines umfassenden hybriden Simulationsmodells auf der Grundlage der aktuellen Erkenntnisse der synchronen und asynchronen Eisenbahnbetriebssimulation, das sowohl betriebsvorbereitend im Rahmen der Infrastrukturdimensionierung und Fahrplanerstellung als auch im Eisenbahnbetrieb bei der teilautomatisierten Disposition zum Einsatz kommen kann. Bislang noch vorhandene Lücken im Zusammenspiel zwischen makroskopischer und mikroskopischer Betrachtung und die praxisorientierte Berücksichtigung von Deadlocks werden durch neuartige Lösungen geschlossen. Damit wird mit dem vorgeschlagenen Modell auch eine einheitliche Anwendung der Eisenbahnbetriebssimulation von der großräumigen Netzbetrachtung bis hin zum Rangierdienst möglich und somit eine wichtige Voraussetzung für eine selbsttätige Zug- und Rangierlenkung geschaffen.

Die Verbesserung der Modellierung und die weitere Automatisierung bei Netzbetrachtungen sind besonders anerkennenswerte Neuerungen. Die Berücksichtigung der einzelnen Fragestellungen, die ausführliche Auseinandersetzung mit verschiedenen, die Thematik tangierenden Sachverhalten und eine objektive Wertung einzelner Einflußfaktoren sowie die hervorragende methodische Zusammenführung der einzelnen Komponenten des Modellkomplexes sind besonders hervorzuheben. Die vorliegende Dissertation stellt eine signifikante Erweiterung des Wissensstandes der verkehrswissenschaftlichen Modellierung im Bereich der Simulation dar und kann darü-

ber hinaus auch in anderen vergleichbaren Gebieten, in denen die Deadlockvermeidung besonderer Aufmerksamkeit bedarf, zum Einsatz kommen.



Dr.-Ing. Ullrich Martin

Universitätsprofessor

# Acknowledgement

It is a great honor for me to thank those who supported and inspired me throughout my dissertation work in the past years.

I would like to gratefully acknowledge the help given by Prof. Dr.-Ing. Ullrich Martin. Motivated from his research, I was encouraged to explore the solution in the field of automatic dispatching and deadlock avoidance. During the research, he gave me valuable and constructive suggestion and advice that enrich the algorithm and my idea. Meanwhile he also demonstrated me how to carry out the research work in a structural and effective way.

Prof. Dr.-Ing. Jörn Pachl has provided me with a comprehensive understanding on the theory and the state of art of deadlock avoidance. I appreciate him agreeing to be my co-supervisor and his insightful comments on my work.

It is a great opportunity to express my thanks to my colleagues, who gave me plenty of support and help in the past projects. Dr.-Ing. Harry Dobeschinsky and Stefan Trischler helped me to get a good understanding on the workflow of dispatching and the traffic simulation in project RUDY. Macro Neuber introduced me the development of railway dispatching and gave me a lot of support in my master thesis and in project PULZURE. Working with Bernd Raubal in project PULRAN, I gained the experience of synchronous simulation and deadlock avoidance. These projects and experiences are the foundation of my dissertation work.

During the writing of the dissertation, Maureen Lynch gave me a lot of help and suggestion on the way for good expression and my English. Teresa Krohn helped me to improve and standardize the formatting of mathematic formula and literature reference. I am also thankful for their carefully reading and commenting on my work.

Finally I would like to thank my wife Jing Fu. It is impossible for me to concentrate and enjoy my work without her unconditional support, encouragement and understanding.

# Table of Contents

# Index of Figures

# Index of Tables

# Abstract

The efficiency and service quality in railway operation can be improved with the support of a partially-automatic dispatching system. Three types of automatic dispatching system model are prevailing: simulative, analytical, and heuristic. However, none of them is able to reconcile the different preferences between system performance and the quality of dispatching solutions individually. A hybrid model is therefore designed in this dissertation.

In the hybrid model, the synchronous simulation is utilized as a basis in order to generate a basic dispatching solution. This dissertation focuses on the components and the workflow of a synchronous simulation and addresses the deadlock problem during a synchronous simulation. Deadlock avoidance can be achieved by the Banker's algorithm and the associated improvements, which are designed to prevent trains from unnecessarily stopping and to improve system performance. The implementation shows that the synchronous simulation can reschedule train movements reliably, and the Banker's algorithm can deal with deadlock problems even for very complex train movements (e.g. shunting movements). In a simulative dispatching mode, the calculated train priority parameters are utilized in requesting infrastructure resources, allocating infrastructure resources, or both.

After a basic dispatching solution has been generated, further optimization can be carried out on a macroscopic level, and then be elaborated on a microscopic level. Several different optimization techniques, including Tabu search and Linear Programming, can be utilized in such a multi-level dispatching and optimization framework. Finally, an optimized dispatching solution will be developed with consideration to the balance of system performance and dispatching solution quality using a simulation-based hybrid model.

In this dissertation, the framework of a hybrid model for a partially-automatic dispatching of railway operation is proposed. A synchronous simulation model is implemented in the work of the dissertation as the basis, from which further implementation can be designed and be developed continuously.

# Zusammenfassung

Mit der Unterstützung einer teilautomatisierten Disposition kann die Effizienz und Qualität im Eisenbahnbetrieb verbessert werden. Obwohl grundsätzlich bereits drei Typen automatisierter Dispositionsmodelle existieren, kann weder der simulative noch der analytische oder der heuristische Ansatz als einzelnes Modell die Systemleistung und die Qualität der Dispositionslösung gleichzeitig berücksichtigen. Um diesen Anspruch praxisorientiert umzusetzen, wird in dieser Dissertation ein Hybrid Modell entwickelt.

Die synchrone Simulation wird bei dem entwickelten Hybrid Modell als Grundlage verwendet, um eine erste Dispositionslösung zu generieren. Der Schwerpunkt der Dissertation liegt auf den Komponenten und dem Ablauf der synchronen Simulation sowie der Lösung des in der synchronen Simulation auftretenden Deadlock-Problems. Durch Verwendung des Banker-Algorithmus und der damit verbundenen Verbesserungsmaßnahmen, die zur Vermeidung von unnötigen Halten und zur Verbesserung der Systemleistung dienen, werden Deadlocks auch bei sehr komplexen Betriebssituationen, wie sie beispielsweise regelmäßig im Rangierdienst auftreten, ausgeschlossen. Im simulativen Teil des Hybrid Modells wird ein Zugprioritätsparameter berechnet, der bei der Ressourcenbeantragung und/oder der Ressourcenreservierung verwendet wird. Die Implementierung zeigt, dass mit synchroner Simulation zwar zuverlässig ein neuer Dispositionsfahrplan erstellt werden kann, eine Optimierung in Abhängigkeit von der Komplexität jedoch sehr schnell an Grenzen stößt.

Nachdem die erste Dispositionslösung generiert wurde, wird eine Optimierung der Lösung auf einer makroskopischen Ebene ermöglicht. Bei Erfordernis kann die Lösung auf einer mikroskopischen Ebene verfeinert werden. Verschiedene Methoden der Optimierung, z. B, Tabu Search und Lineare Optimierung, können im Rahmen einer Multi-Level-Disposition und -Optimierung über mehrere Ebenen eingesetzt werden. Schließlich wird in dem simulationsbasierten Hybrid Modell eine optimierte Dispositionslösung unter Berücksichtigung des Ausgleichs zwischen der Systemleistung und der Qualität der Dispositionslösung entwickelt.

In dieser Dissertation wird der Rahmen eines Hybrid Modells für eine teilautomatisierte Disposition des Eisenbahnbetriebs entworfen. Ein erweiterungsfähiges synchrones Simulationsmodell wird als Grundlage implementiert.

# 1 Introduction

Today's railway dispatching is still carried out more or less manually and still relies heavily on experienced dispatchers. The ever increasing running speeds and traffic volume of trains require a high quality operation control to ensure the availability, reliability, and efficiency of railway services. It is these requirements that are challenging the traditional manual dispatching approaches.

Several endeavors have aimed to transform the manual dispatching approaches into automated dispatching processes. However, it is now common knowledge that employing a fully-automatic dispatching system is unrealistic with the current technologies. The most practical goal is to implement a computer-supported, partially-automatic dispatching system. From this point on, in this dissertation "partially-automatic dispatching" will be referred to as "automatic dispatching".

The purpose of designing a partially-automatic dispatching system is to relieve the dispatchers from routine and tedious, computational workloads especially in case of deviations from regular scheduled service. After being generated from a partially-automatic dispatching module, the basic dispatching solutions may be further optimized with certain objective functions. Therefore, the automatic dispatching workflow can be categorized into two types of processes: 1) generating basic dispatching solutions and 2) optimization. However in some cases, the two processes are tightly integrated together.

In the past years, several commercial software tools for automatic dispatching have been adopted for use in the railway business. These software tools focus on generating dispatching solutions with little or completely without optimization features. Meanwhile, research has been conducted and applications in the fields of automatic dispatching with varied methodologies have been developed at the IEV (Institute of Railway and Transportation Engineering at the Universität Stuttgart, in German: *Institut für Eisenbahn- und Verkehrswesen der Universität Stuttgart*). Most of these applications concentrate on optimization processes, and until now only laboratory versions have been available due to the highly computational complexity of the optimization processes.

After examining and comparing different simulation and optimization approaches, a hybrid approach between the laboratory versions and the versions used in real railway operation has been designed in this dissertation to provide optimized dispatch-

ing solutions with consideration to the balance of system performance and dispatching solution quality. Since a feasible dispatching solution is always desirable in an automatic dispatching system, and simulative models are the most realistic approach to generate a basic dispatching solution for an automatic dispatching system with today's technology, a simulation-based approach serves the basis of the hybrid approach. Further optimization is then carried out to improve the quality of the basic dispatching solutions.

In this dissertation, the synchronous simulation model has been designed and implemented. The special problem of deadlocks in a synchronous simulation model is examined. The solution based on the Banker's algorithm for deadlock avoidance and several improvements of the algorithm are developed. With the input of the basic dispatching solution, a multi-level dispatching and optimization framework is proposed. Within this framework, the dispatching solutions are optimized on a macroscopic level and are further elaborated on a microscopic level.

The purpose of this dissertation is not to cover all of the possible dispatching technologies nor simply combine them together. Instead, the core of the hybrid approaches is to identify the basic requirements and characteristics in the processes of generating dispatching solutions and optimization, and thus to find a suitable approach to fulfill the requirements with the current technologies. In such a way, the view of an automatic dispatching system is divided into different levels, each equipped with a corresponding dispatching approach. All of the adopted approaches cooperate with each other, and the implementation of each level can be improved without influencing the other levels. In brief, the framework of simulation-based hybrid approaches is a concrete solution to balance the system performance and solution quality for railway dispatching services.

The structure of this dissertation is organized as follows: in Chapter 2 the processes related to the dispatching system in railway operation control, with the discussions of different automatic dispatching approaches are introduced. In Chapter 3 the synchronous simulation model, including its components and how the different components interact with each other are explored. The problem of deadlocks in a synchronous simulation model is presented in Chapter 4, and deadlock avoidance algorithms with their associated improvements are also discussed in depth. In Chapter 5, the methods for determining train priorities in simulative dispatching are given. A basic dispatching solution can be optimized with a multi-level dispatching and optimization

framework, which is proposed in Chapter 6. Within the framework, different optimization techniques are utilized and integrated, and these techniques are also presented and discussed in Chapter 6.

This dissertation work is based on German practices and systems. However, the principles discussed can also be applied to other systems in other countries. Several projects developed at the IEV supported the research. The project PULZURE (see Section 2.1.4 and [IEV, 2005]) provided the knowledge base of train regulation and railway operation control system. Within the project PULRAN (see Section 4.5 and [IEV, 2009]), the synchronous simulation model and deadlock avoidance algorithm were examined and evaluated. A series of referenced projects either dealt with the analytical optimized dispatching approaches based on MARTIN's model (see Section 2.2.2, [SCHLAICH, 2002], [HLAWENKA, 2003] and [CUI, 2005]), or explored other optimization techniques (see Section 2.2.3 and [THOMA, 2008]).

# 2 Railway Operation Control and Automatic Dispatching

Railway operation control system helps railway business to manage train movements and to ensure railway service quality. In order to save the staff costs and improve the productivity in the railway business, highly centralized operation control system is desirable. Today, it is a trend to integrate traffic controls within a high-performance railway operation control center. Such a highly concentrated operation control system challenges dispatchers to manage a considerable amount of train movements on a large scale. As a component in railway operation control system, dispatching system is used for solving conflicts and rescheduling train movements in real time operations. With the help of computerized operation control system, an automatic dispatching system can support dispatchers for decision-making efficiently.

The development of automatic dispatching system requires multiple-disciplinary knowledge including transportation operation and management, operations research, as well as software and information technology. Several research and applications endeavor to design and to implement automatic dispatching systems with varied algorithms. The current practices show that an automatic dispatching system cannot completely replace experienced dispatchers yet. Dispatches are still managing dispatching process in a manual way.

This chapter starts to introduce the functions and the characteristics of dispatching systems in railway operation control system, and then several different methods of automatic dispatching system are discussed. The pros and cons of these methods are compared finally.

## 2.1 Dispatching in Railway operation Control System

In this section, the structure of modern railway operation control system is discussed, with the introduction of dispatching processes. Two different kinds of dispatching measures, time-related dispatching and location-related dispatching, are also summarized.

### 2.1.1 Railway Traffic Control

Two types of traffic control authorities are in effect for the current, signalized railway operation control: traffic control with local operators/train directors (in German: *Fahr-*

*dienstleiter*) and centralized traffic control (CTC). The design of a dispatching system should be considered for both control types.

Every railway line is divided into many different operational territories. The definitions of territory limits vary in different countries according to their practices. In [PACHL, 2002], the definitions of the territory limit in North America, Great-Britain, and Germany are given.

In Germany, the railway traffic controlled with local operators is applied in many conventional railway lines. Local operators working in the interlocking tower charge all the train movements inside an operational territory. The local operator takes the full responsibility to supervise train movements and to issue movement authorities. A dispatcher monitors train movements and coordinates with related local operators to avoid possible conflicts caused by deviations or to deal with other emergency events. In such a way, a dispatcher supports several local operators to control the rail operations in a large network efficiently. The traffic control with local operators is also applied in other countries such as in China and other European countries railways.

Originally CTC is applied for the lines with long distance and low population density, such as in North America, where the layout in an operational territory is simple (comparing with the network in Europe). It is efficient to authorize all train movements via dispatchers directly (see [WHITE, 2003]). After the utilization of electronic interlocking systems in Europe since 1980's, operation control could also be fulfilled remotely on computers for a large and complex network, and therefore local operators would theoretically not be necessary. With a centralized traffic control, the costs for personnel, infrastructure, energy, and vehicles can be saved if further optimizations are adopted (see [MÜCKE, 2002]). By applying optimized dispatching solutions, the infrastructure utilization level can be improved, and the investment for line expansion can be reduced. Energy-efficient driving style and driver support systems can be achieved with the information provided by the control center, and then the energy consumption can be reduced accordingly (see [ALBRECHT, 2008]). The investment costs for vehicles would be reduced if vehicle circulation schedules are optimized in the dispatching system, and the maintenance costs can also be saved if unnecessary acceleration and braking phases are avoided with optimal driving strategy. Besides, additional indirect benefits, such as the increased traffic demand and passenger acceptance, will be gained due to the better service quality and improved user satisfaction.

CTC is utilized widely in modern rail operations. For example, seven operation control centers (OCC) are established in Germany with centralized traffic control in order to increase the system effectiveness. In China, CTC is also put into operation in some new built lines, such as Longhai line and Qingzang railway.

Dispatching operations are often categorized into some levels. A typical structure of operation control center is described in [PACHL, 2002] as follows:

In a German operation control center, all the train movements are supervised by the regional dispatcher (in German: *Bereichsdisponenten*). The whole region is divided into some regulating areas, and a traffic controller (in German: *Zugdisponent* or *Zuglenker*) takes the responsibility of solving possible conflicts as well as other dispatching tasks in each regulating area. Cooperating with the traffic controller, traffic operators fulfill the operational tasks, including the safety related operations and shunting operations, in the regulating area. Compared to the traffic control with local operators, the CTC in Germany has a similar structure but different interlocking technology. The structure of a German operation control center is shown in Fig. 2-1.

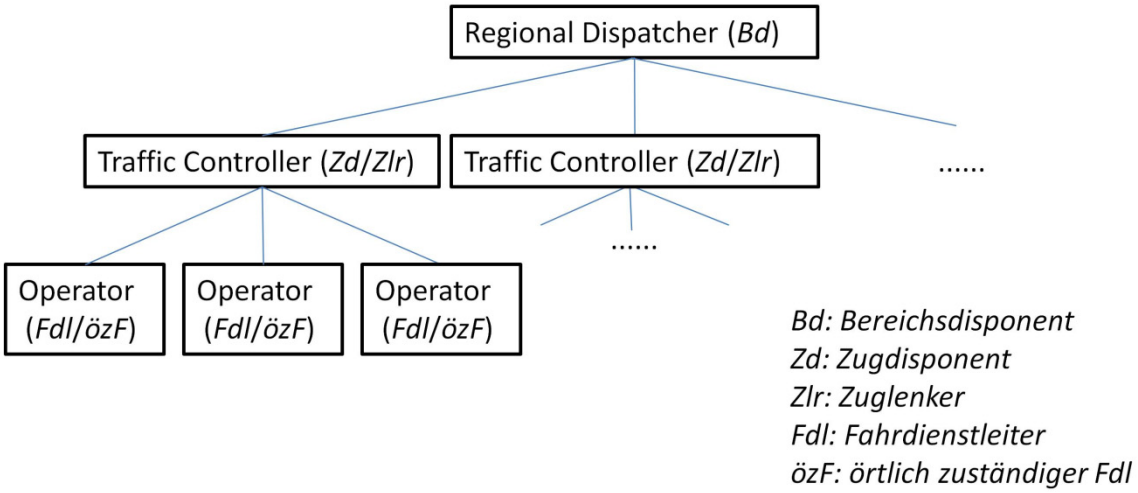



**Fig. 2-1 The Structure of a German OCC**

Above regional operation control centers is the network operation control center (in German: *Netzleitzentrale*), which dispatches long distance passenger trains and selected freight trains, coordinating train movements over many regions and borders.

Although theoretically a dispatcher could directly control all the operations with the assistance of CTC in an operation control center, it is not efficient to divide the network into small pieces and to assign all dispatching and related operations of each piece to one dispatcher. Working with such a flat structure, it would be difficult to

keep a clear overview of the whole supervised lines, and the coordination and communication workloads among all the small control areas would be immense.

Two characteristics of modern railway traffic control are highlighted here: centralized traffic control and a structured hierarchy. Within a centralized traffic control system, it is efficient to implement a highly automated system and to optimize the involved operations based on common data basis, standardized workflows, and unified user interfaces. In a structured hierarchy between dispatchers and operators, different levels of details are concerned, while irrelevant aspects are separated from the core dispatching decision processes. The structured hierarchy also influences the design of an automatic dispatching system. Further discussion will be given in Chapter 6.

### 2.1.2 Dispatching Process in Railway operation Control

The most important target of a train dispatching system is to identify and to solve potential conflicts caused by deviations or other accidents in railway operation. The dispatching process shown in Fig. 2-2 is executed as a cycle to monitor and forecast the operating situation, identify and solve possible conflicts, and reschedule train movements (if necessary) continuously.

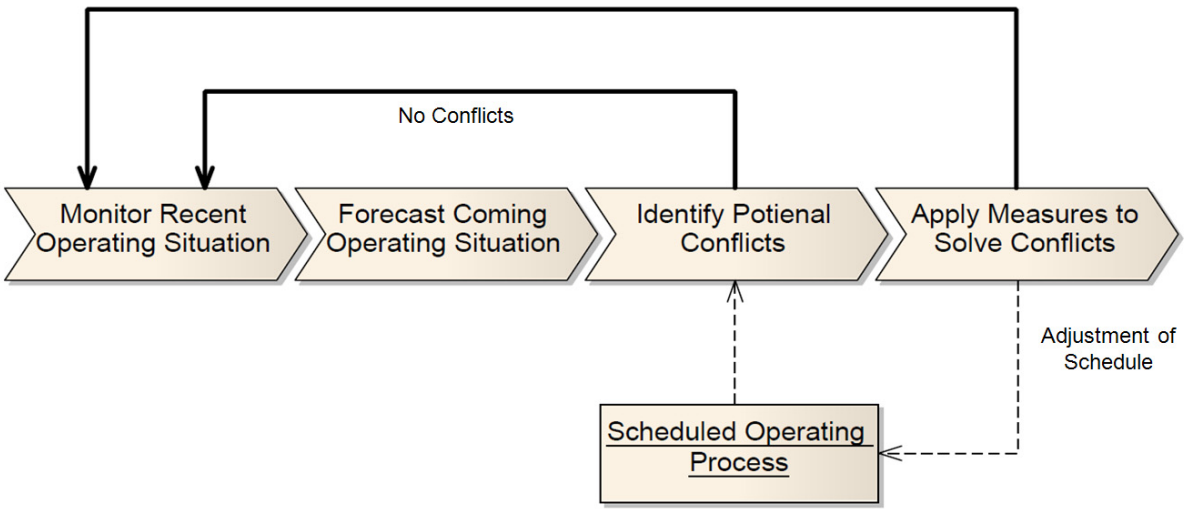



**Fig. 2-2 Dispatching Process in Railway operation Control**

The information about recent operating situation is collected periodically. Local operators or CTC operators provide the information as messages to dispatchers, and the messages are also exchanged among neighborhoods and the network control center. Special information including delay registrations or other accident (out of service) registrations is reported from traffic controllers and operators to regional dispatchers. The delay registrations are defined by certain thresholds. For example, according to

the German dispatching guideline (see [DB NETZ AG, 420.0105]), a delay situation should be registered if a passenger train has a delay of more than 5 minutes, or a freight train has a delay of more than 15 minutes.

Based on the presented information, the prognosis of coming operating situation is concluded, which depend highly on the extrapolated running time. From the forecasted operating situation, potential conflicts can be identified. There are six kinds of conflicts defined in [PT1, 2007]:

- Conflicts at track sections or routes
- Conflicts at scheduled stops
- Connection conflicts
- Timetable conflicts
- Dispatching conflicts
- Deadlock conflicts

Conflicts at track section, routes, and scheduled stops are caused by deviations, system failure, or accidents. These conflicts can be perceived as occupancy conflicts. A connection conflict is the discord of trip connections among passenger trains or freight trains. When a train is deviated from timetable, it is possible that other trains are influenced and the punctuality of timetable is violated where timetable conflicts take place. In a dispatching system with many hierarchies, dispatching conflicts may happen when the dispatching instructions issued from different levels are inconsistent. A deadlock conflict is the situation when all conflicted trains are blocked by others and none of them can continue their trip. A detailed discussion on deadlocks will be given in Chapter 4.

As long as conflicts occur, the dispatcher should take one or more dispatching measures to solve the conflicts (see Section 2.1.3). After a dispatching solution is found, the operation schedule is adjusted and a new round of cycle is executed repeatedly.

### 2.1.3 Measures of Railway Dispatching

The solutions for handling different kinds of conflicts are illustrated in [MARTIN, 1995], and the possible dispatching measures are also summarized in [MARTIN, 1995], [DB NETZ AG, 420.0105], and [STANGE, 2007]. The measures of railway dispatching can be categorized as time-related dispatching and location-related dispatching.

#### 2.1.3.1 Time-related dispatching

Time-related dispatching is to adjust the operating schedule by changing train driving speed as well as the trip time (conveyance time). It is the most frequently used dispatching measure. Even if location-related dispatching is utilized for one train, the trip time may be adjusted also in order to coordinate with other trains.

Not only the running time, but also the dwell time, can be manipulated for adjusting train movements. In case of a minor delay, the designed recovery time of running time/dwell time in the timetable provides the possibilities for the delayed train to catch up with the original schedule. The trip time is not always reduced in time-related dispatching. Additional unscheduled waiting time may be introduced, for example, if a delayed train is designated to wait until another successive train overtakes it.

Time components of trip time will be identified as the input data for a time-related dispatching. The time components include (see [PACHL, 2002]):

- Required minimal operation time (pure calculated running time, passenger related pure dwell time, operational dwell time, and time for preparing train departure)
- Recovery time (regular recovery running time, special recovery running time, and passenger related recovery dwell time)

As a result of a time-related dispatching, the reduced/increased trip time at an infrastructure resource (e.g. at a block section or at a station area, see Section 3.1.1) is decided. Afterwards, the driving support module will recommend an energy efficient driving solution. However, the calculation for optimized driving style is an advanced task of dispatching system. The energy efficient driving solution will not be covered in this dissertation.

#### 2.1.3.2 Location-related dispatching

According to the definition in [MARTIN, 1995], location-related dispatching is to adjust train movements by building a new route consisting of a series of reference points (a reference point can be a block section, a track, a node, or a line section). Usually it is applied accompanying with time-related dispatching. Location-related dispatching may be used in the following situations:

- An infrastructure resource or an infrastructure object (track, point or signal) is out of service (e.g. in case of system failure or accident) or unavailable (with unscheduled maintenance or other occupancies on tracks).

- System efficiency will be improved (e.g. with reduction of overall weighted trip time or delay time) when alternative routes are applied.
- Additional unscheduled train movements (e.g. shunting movements) are introduced.

A faster train can overtake a slower train to avoid to be obstructed in order to reduce total delay time. Overtaking is a typical location-related dispatching applied in railway operation. It is frequently used to improve system efficiency.

Other location-related dispatching including detour, shortening scheduled route, and dispatching of the rolling stocks are introduced in [STANGE, 2007]. Besides, the personnel dispatching and the (temporarily) management of constructing and maintaining sites are also defined as possible dispatching measures in [DB NETZ AG, 420.0105]. These methods are still difficult to be handled by an automatic dispatching system completely, and are not covered in this dissertation.

### 2.1.4 PULZURE: a Software Tool for Guiding Train Movements

At the end of 2005, the project PULZURE (Program to Support a Partially-automatic Train Running Control, in German: *Programm zur Unterstützung einer Teilautomatischen Zuglaufregelung*) was launched for German Railway Network (DB Netz AG). The aim of the project was to improve capacity and punctuality of rail transport by means of high-quality train running control. It was developed by the IEV and implemented as a software product operated in operation control center.

In case of deviations, the dispatching system regulates train movements collaborating with the modules of the driving time calculation and movement forecasting. On the one hand, a single delayed train shall try its best to catch up the timetable; on the other hand, the constraints of the interdependence with other trains shall be considered.

With the assistances from the high-quality train running control, the dispatching instructions (in PULZURE, they are generated manually) are converted to driving recommendations to realize the optimized looking-ahead driving. The optimization of running also minimizes the delay by driving with the maximum permitted speed. Hence the capacity of the network is increased and the operation costs are reduced as additional benefits. The screen shot of partial control panel in PULZURE is shown in Fig. 2-3.

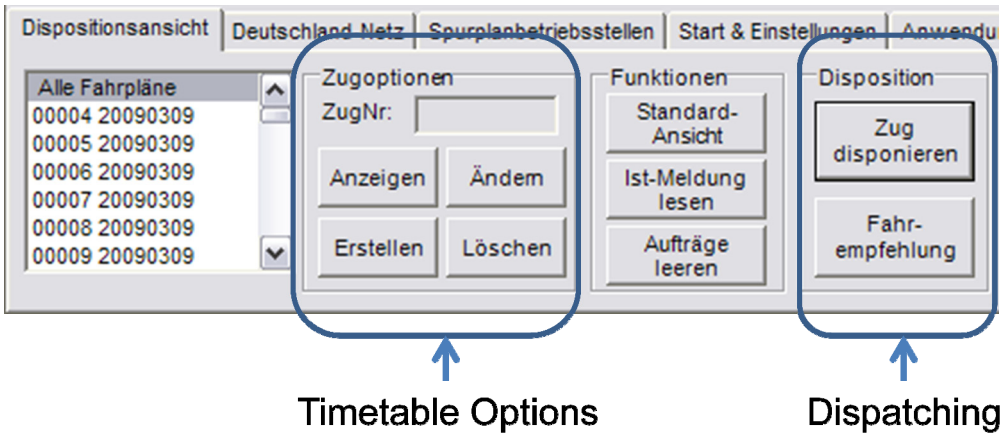



**Fig. 2-3 The Control Panel of PULZURE**

There are two areas: timetable options (*Zugoptionen*) and dispatching (*Disposition*) on the control panel. In the “timetable options” area, the original timetable of a train may be modified or deleted, and a new train may be introduced into the current operation. All of these operations involve location-related dispatching, such as detouring or shortening scheduled route. In the “dispatching” area, only the time-related dispatching and overtaking are allowed. Train movements can be regulated within a dispatching window (shown in Fig. 2-4) or by mouse-controlled dispatching.

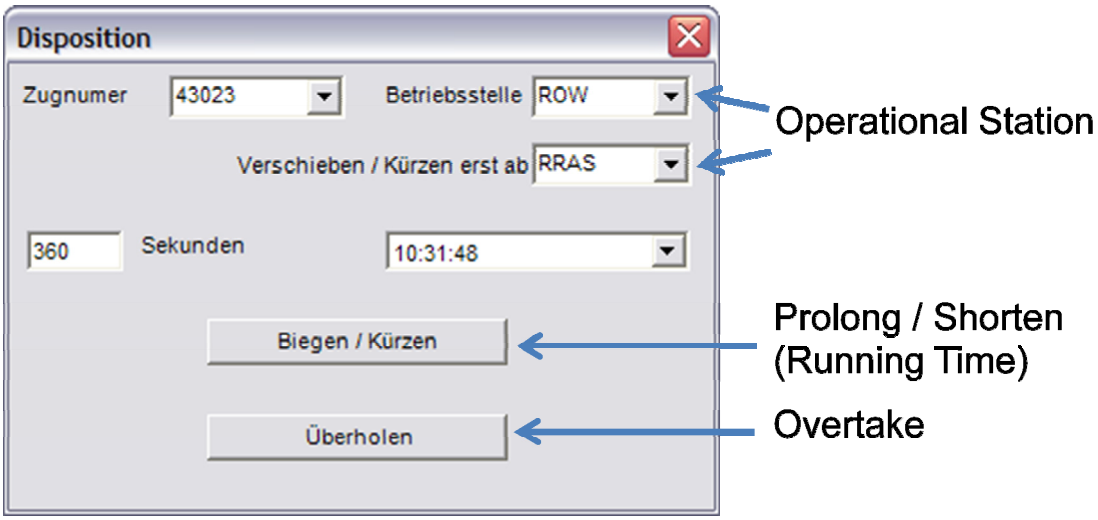



**Fig. 2-4 The Dispatching Window of PULZURE**

The different train regulation areas in PULZURE reveal a phenomenon that dispatching measures are differentiated in more than one dispatching processes in the current practices. By categorizing dispatching measures, the complexity of the system will be reduced. The basic dispatching functions, including time-related dispatching and overtaking, can be separated from other sophisticated processes that require more manual intervention. The separated design will also be adopted in optimization process (see Section 6.1).

## 2.2 Research and Applications on the Field of Automatic Dispatching

The purpose of designing an automatic dispatching system is not to replace dispatchers completely, but to relieve dispatchers from routine and tedious, computational workloads especially in case of deviations.

It is not reasonable to regulate train movements and handle all conflicts and accidents with a fully-automatic dispatching system. The domain models used in dispatching software are abstracted from reality and are not able to reflect all the various aspects of railway infrastructure, rolling stocks, and operations. The decision process is impossible to be completely documented and programmed, and some of the mental activities are even not perceivable. Additionally, if an optimal solution is demanded, practices show that an optimal result is difficult to be resolved in many cases due to the computational complexity, even working with a simplified data model.

Although the capability of automatic dispatching is limited, a computer supported dispatching system provides a great potential to improve the productivity and efficiency of railway operation control. A computer-based dispatching system can take over many routine and regular tasks involving much computational workloads, such as extrapolation of running time, forecast of train movements, identification of conflicts, and recovery from minor deviation by time-related dispatching or overtaking. In a highly centralized traffic control center, the benefits of applying a computer-based train regulation are considerable (see [OETTING, 2006]). With the support of an automatic dispatching system, dispatchers are able to concentrate on more challenging and sophisticated dispatching tasks that still highly rely on manual intervention. These tasks include emergency operations, accidents management, personnel/rolling stocks dispatching, and empiric-based optimization. The service quality is therefore improved by sharing the workload between dispatchers and software tools.

The technique of automatic train dispatching is still in development, and most of the solutions have not yet been put into real operations. According to the model definition and the algorithm used for resolving the solution, automatic dispatching models can be categorized into three types: simulative models, analytical models and heuristic models (the former graphical models are not longer in use).

### 2.2.1 Simulative Models

With simulative approach, all train movements and operations are executed in a computer model that simulates the reality. The dispatching system is initialized with scheduled operations, and updated with real-time train movement messages. The process of forecasting future train movements and identifying potential conflicts are executed iteratively. Once conflicts occur, a reschedule process based on a certain (synchronous or asynchronous) mechanism will be activated, and a new conflict-free schedule will be generated.

The simulation models for railway dispatching of real-time train running should be based on a very detailed microscopic infrastructure model, which includes the most detailed infrastructure information, because of the very strict safety requirements in railway operation. A technique of infrastructure modeling is given in [RADTKE, 2008].

The simulative methods can be classified as synchronous simulation and asynchronous simulation. Synchronous models process all train movements simultaneously, and asynchronous models rank trains according to their priorities and insert trains into simulation process successively based on the ranks.

Synchronous simulation is the most often used simulation model (see [SIEFER, 2008]). Many software tools are developed based on this model, such as RailSys (by the Leibniz Universität Hannover and Rail Management Consultants) and OpenTrack (by Eldgenössische Technische Hochschule Zürich). Other commercial synchronous simulation tools are also enumerated in [SIEFER, 2008]. The biggest challenge of a synchronous simulation model is the unsolved deadlock problem. In Chapter 3 and Chapter 4 the synchronous simulation model and the solution for resolving deadlocks will be explored in details.

A put in used asynchronous simulation approach is developed by JACOBS and implemented in the software tool ASDIS (Asynchronous Dispatching, in German: *Asynchrone Disposition*). The rescheduling process in the ASDIS method is similar to the procedure of constructing timetable. All trains are ranked according to the train priorities, and introduced by the rank, starting from the group with the highest priority.

When a group of trains belonging to a certain rank is imported, conflicts with the previously imported trains, which have the higher or the same priority, may arise. In principle, only the new introduced trains are adjusted. The conflicts are resolved in the order of time by certain dispatching measures (time-related, location-related, or both). After a conflict is solved, knock-on conflicts may occur due to the adjustment of the

original schedule. These conflicts will be identified and resolved as well. A new group of trains with lower priority ranking will then be introduced after all conflicts of the current group are solved. The rescheduling process is executed repeatedly until all trains are processed, and a conflict-free timetable is derived finally.

The train with a higher rank will take priority over others in the ASDIS's method. When partial priority is applied, a low-priority train may get the precedence if a certain tolerance threshold is reached (see [JACOBS, 2008]).

In synchronous simulation, all train schedules are constructed gradually at intervals. At each interval, all the trains are processed concurrently. Deadlocks may happen in the lines or in the station limits with bidirectional operations (see Chapter 4). In contrast to synchronous simulation, deadlocks do not take place in asynchronous mode. In a simulation process with asynchronous mode, the complete train schedule of a train is built up immediately once the train is introduced into the scheduling process (with further possible adjustments if conflicts exist). Therefore all the train movements are ensured to be completed without deadlocks.

In an asynchronous simulation, each train schedule can be built completely without execution of each single step at each interval. Besides, the processing time for resolving deadlocks will be saved. In these senses, asynchronous simulation is faster than synchronous simulation. However, in the case of some very busy lines, the amount of possible conflicts and knock-on conflicts are so considerable that the additional time for conflict resolving will bring negative impacts to the performance of asynchronous mode.

Another feature of asynchronous simulation is quite controversial. In an asynchronous simulation, the train priorities are not determined in real time (like in a synchronous simulation), but are determined before starting the simulation process. The criticism of asynchronous simulation is that the anti-discriminatory principle may be violated, if train priorities are defined earlier based on a possibly biased foundation.

Today's research shows that the two modes of simulation, synchronous and asynchronous, can be incorporated with each other. "Asynchronous components" (see [JACOBS, 2008] and [DB NETZ AG, 405.0105 III]) can be introduced in a synchronous simulation, and a synchronous simulation can be executed for a certain train group in an asynchronous simulation. An example for the combination of the two modes can be seen in Chapter 5.

### 2.2.2 Analytical Models

With analytical approaches, train operations are abstracted to a mathematic model, and the solution is resolved based on a certain operations research algorithm. Many different approaches are proposed, including event tree analysis, summation method, linear programming, or optimization with a system of differential equations (see [MARTIN, 2002]).

In an analytical model, an objective function has to be defined with a specified optimization objective. The optimization objective reflects specific dispatching strategies. Meanwhile, the model should be restricted by many constraints (a set of equalities or inequalities) that map the real train operations to the abstracted model.

A linear programming model is used to find the optimal solution for the problem with linear objective function and constrains at the IEV. The model was first developed by MARTIN in 1995. SCHLAICH and HLAWENKA contributed further development with alternative route search. Rail operations can be exactly defined and mapped within the linear model, conforming to the strict safety requirements.

Originally, the objective function was to minimize the total weighted delays (see [MARTIN, 1995]). After introducing alternative routes (virtual course), the overall weighted trip time should be minimized (see [SCHLAICH, 2002]). The objective function is defined as:

$$\sum_{j=1}^{nges} C_j \cdot \left\{ \sum_{k=1}^{nvZ} \left[ \sum_{i=1}^{iges} \left( tw_{i,j,k} + hH_{i,j,k} \cdot (tBr_{i,j,k} + tAn_{i,j,k}) \right) \right] + hvZa_{j,k} \cdot \sum_{i=1}^{iges} tI_{i-1,j,k} \right\} \quad (2\text{-}1)$$

$$\rightarrow \min$$

Notations used:

| | |
|---|---|
| $i$ | index of the block section or scheduled stop |
| $j$ | index of the train |
| $k$ | index of the virtual course |
| $tw_{i,j,k}$ | unscheduled waiting time of train $j$ at block section or scheduled stop $i$ for virtual course $k$ |
| $tI_{i,j,k}$ | scheduled running time of train $j$ in block section $i$ for virtual course $k$ |
| $hvZa_{j,k}$ | binary switch variable to indicate whether train $j$ selects a virtual course $k$ |

| | |
|---|---|
| $tBr_{i,j,k}$ /$tAn_{i,j,k}$ | additional braking/acceleration time of train $j$ in block section $i$ for virtual course $k$ |
| $hH_{i,j,k}$ | binary switch variable to indicate whether an unscheduled brake happens for of train $j$ in block section $i$ with virtual course $k$ |
| $C_j$ | train related constant for weighting of trip time for train $j$ |
| $neges$ | total number of trains |
| $nvZ$ | total number of the virtual course for a real train |
| $iges$ | number of the destination for a virtual course |

For a certain virtual course $k$ of train $j$, the departing time or passing time of block section $i$ (denoted as $tB_{i,j,k}$) can be calculated recursively as a set of constraints generated automatically.

$$tB_{i,j,k} = tB_{i-1,j,k} + hvZa_{j,k} \cdot tI_{i,j,k} + tw_{i,j,k} + hH_{i,j,k} \cdot (tBr_{i,j,k} + tAn_{i,j,k}) \quad (2\text{-}2)$$

Conflicts at track sections or routes can be avoided with occupancy conflict-free constraints. The conflicts between two trains with “following” or “filter out” movements can be prevented by regulating the departing time for the following train. By introducing priority decision variables, the constraints for resolving conflicts between two trains with “merging” routes or “opposite” movements can be set up as well.

Deadlock conflicts are avoided by restricting the sum of waiting time for all the trains, and the sum of the waiting time per train should not exceed 24 hours in regular public railway service:

$$\sum_{k=1}^{nvZ} \sum_{i=1}^{iges} tw_{i,j,k} < 24h \quad (2\text{-}3)$$

Comparing with the algorithm for resolving deadlocks in synchronous simulation (see Chapter 4), the deadlock-free constraint is concise and graceful within the linear programming model. The method for deadlock-free also demonstrates the attractiveness of the analytical models: within a succinct mathematic model, the optimization objective and operations characteristics are formulated exactly and efficiently. A complete description of the linear model and further development are discussed in [MARTIN, 1995] and [SCHLAICH, 2002].

The most critical challenge of analytical models is the capability of resolving the optimal solution. Many analytical models are limited by system performance and computational complexity. The strict safety requirements of rail operation result in a model with very detailed infrastructure information and operational controls. It is difficult to

find a solution with analytical models for a large scale network with many train movements. Since the binary decision variables are used in the linear programming model, the optimization problem of the model becomes mixed integer programming (MIP) problem that is classified as NP-hard. The computational complexity will be increased exponentially when the scale of the problem increases. The statistic data of the system performance for an implementation of linear programming model is given in [CUI, 2005]. With high computational complexity, the cost of analytical models is unacceptable for a large dispatching territory (especially for centralized traffic control).

It is meaningful to distinguish a model's *definition* and a concrete *optimization technique*. Taking the example of MARTIN's model: even though it is a linear programming model, it is not necessary to bind it with a linear programming technique. If the size of the problem it too large, other optimization algorithms can be adopted. One of the approaches is to employ heuristic models.

### 2.2.3 Heuristic Models

Heuristic models balance the quality of the solution and the computational complexity. Based on experiences or a search algorithm, heuristic models are developed to find new solutions toward the direction of the optimal solution. The method for evaluating candidate solutions should be defined previously, and according to a certain algorithm, new solutions are generated, evaluated, and selected. The quality of the solution is improved gradually with further executed searches. A relatively optimal solution will be derived by the end of pre-defined time or after certain number of iterations.

In heuristic models, the evaluation method can be based on a single objective like in analytical models (e.g. weighted delays or overall trip time). It also can be a combination of multiple objectives. How to define the objective of an automatic dispatching system is not covered in this dissertation.

Many heuristic algorithms are developed to generate candidate solutions and to search the optimal solution. These algorithms include genetic algorithm, tabu search, simulated annealing, expert system and problem space search.

A laboratory software solution has been developed and tested for a small experimental network in the IEV. It is based on genetic algorithm developed by THOMA (see [THOMA, 2008]). In this solution, the fitness function is constructed with total trip

time, which is used to evaluate the quality of the solution. Two levels, train paths and train sequences, are optimized with evolutionary algorithm in sequence. Through mutation and recombination, alternative routes are searched, and train sequences are exchanged as well. The optimization process is repeated in a pre-defined time period and an optimized dispatching schedule is finally generated. Since the deadlock problem has not been solved with this solution, generated unrealistic solutions are identified and discarded, and in some cases the system fails due to deadlocks. The deficiency of the tool reveals that it is difficult to find even one feasible, basic dispatching solution in an automatic dispatching system with population-based heuristic algorithm.

A knowledge-based expert system is another option for heuristic models. A software solution for accident-management was developed by the IEV for the project RUDY (Regional Enterprise-spreading Dynamic Sampling of Timetable Information, Reservation and Operation in Public Transport, in German: *Regionale Unternehmensübergreifende Dynamisierung von Fahrplaninformation, Buchung und Betrieb im ÖPNV*) that intended to improve service quality of regional public transport (see [TRITSCHLER et al., 2005]). In the project RUDY, the historical dispatching decisions made by dispatchers are logged and referenced by the location of each accident or traffic jam. Once a new accident occurs, solutions (detour routes) may be proposed based on the matched historical decisions. It is possible that several historical decisions are available for a specific accident location. The suggested detouring routes are ordered according to the combination of a set of weighted indicators, including the length of the route, the trip time of route, the frequency of the route usage, the recency of the route usage, number of missed stations (comparing with original timetable), and the number of the additionally served stations. Similarly, an expert system can also be used to determine detouring route in railway dispatching system. This method relies heavily on manual dispatching, since the historical data are collected either in real operations or by experienced dispatchers in a simulated environment.

To improve the efficiency of train scheduling process, an automatic scheduling system is developed by University of South Australia and TMG International (see [PUDNEY and WARDOP, 2008]). Combined with a fast dispatch heuristic, which will generate a single basic solution, the problem space search is utilized to find a best solution from many randomly perturbed alternatives. Two principles, first-to-start or first-to-finish, can be used to guide the heuristic process to determine the timing of

choosing train movements to construct a basic solution. The generated basic solution is not an optimal solution, and will be further optimized in a probabilistic search algorithm. By setting the rules of perturbations, many random alternatives are generated and evaluated with certain assessment criteria (e.g. delay or costs) to find out the optimal solution. This method has been put in use in several real Australian rail networks for train planning and timetable construction. It can also be used in operation control center for automatic dispatching as an application of “dynamic rescheduling”. A synchronous process is used to generate timetable and alternatives. The process may derive results with deadlocks. The solutions ending in deadlocks will be simply discarded if the proportion of such solutions is not high.

## 2.3 Simulation Based Hybrid Approach for Automatic Disposition

The models presented in Section 2.2 are compared in Table 2-1. The criteria used here are: level of optimization, computational complexity, and implementation costs.

| Models | Level of Optimization | Computational Complexity | Implementation Cost |
|---|---|---|---|
| **Simulative Models** | Low | Low | Low |
| **Analytical Models** | High | High (may be NP-Hard) | Medium |
| **Heuristic Models** | Medium | Medium | Medium or High |

**Table 2-1 Comparison of Automatic Dispatching Models**

The level of optimization reflects the solution quality. The computational complexity indicates the system reliability. In the worst case, a model cannot find a solution due to high computational complexity. A higher level of optimization (positive) is often accompanied with a higher computational complexity (negative). Implementation costs are evaluated from the point of view of software development. For both analytical and heuristic models, an optimization module is required, and the integration process is still necessary even although many commercial software tools are available. For some heuristic models (e.g. the model based on expert system), additional data bases are required to generate candidate solutions. The process of data collection itself in many cases incurs much more (time and human) costs comparing with the costs of software development.

A hybrid approach is recommended to balance the solution quality and the system performance. As the most reliable and the least-cost approach, a simulative model can be used to generate a basic dispatching solution. If it is necessary, further opti-

mization will be executed by using an analytical model or a heuristic model. Therefore, at least a basic dispatching solution can be ensured as a dispatching solution, and the quality of the solution will be improved within an acceptable time period. A hybrid approach is proposed in Chapter 6. The basis of this approach, simulative model (based on synchronous mode), will be introduced in Chapter 3.

# 3 Synchronous Simulation in Railway Operation

Operational simulation is widely used in the fields of railway planning and operation such as timetable constructing, operation control, and capacity research. There are two different kinds of simulation methods: synchronous simulation and asynchronous simulation. In synchronous simulation the movements of all trains are simulated simultaneously; on the contrary, in asynchronous simulation trains are grouped according to their priority and put into simulation in the order of priority gradually.

A simulation process of a single train in an asynchronous simulation workflow can be regarded as a synchronous simulation. For this reason, the synchronous simulation is the basic form of simulation. Synchronous simulation is driven following the real time sequences by fixed time intervals in a time-driven simulation, or by flexible time intervals in an event-driven simulation. Time-driven simulation is very suitable to demonstrate train movements and operational activities in reality. Event-driven simulation is utilized to improve system efficiency. The mechanism of event-driven simulation will be discussed in Section 3.3. In this chapter, the model of synchronous simulation used in this dissertation is explained. If not otherwise indicated, the term “simulation” used in Chapter 3 and Chapter 4 refers specifically to “synchronous simulation”. The asynchronous simulation is highly related with train priorities, which are the key elements for rule-based dispatching. A solution based on the synchronous model combined with asynchronous simulation will be discussed in Section 5.3.

There are two perspectives to model the simulation system for railway operation: the structural perspective and the behavior perspective. In Section 3.1, the depiction of the components for railway simulation describes the structural elements composing the whole system. The workflow of synchronous simulation, which reflects the behavioral features of the system and business process, will be illustrated in Section 3.2. In addition, the special characteristics for railway operation, including running dynamics, the safety requirements, and the signaling technique, are taken into consideration in the model.

## 3.1 The Components of Synchronous Simulation

The components involved in the synchronous simulation workflow are infrastructure resources, simulation performers, and simulation tasks, which are mapped to the key elements of rail system – infrastructure, vehicles, and operations – respectively.

### 3.1.1 Infrastructure Resources

In a railway simulation, the infrastructure network is broken down into many single infrastructure resources. An infrastructure resource (from this point on in this dissertation, the term “resource” will refer to “infrastructure resource”) is a basic unit of the infrastructure in railway operation and simulation. The principle to define an infrastructure resource depends on how the resources are requested and allocated. Therefore, the definition of the infrastructure resource depends on the application context: it can be defined based on the block sections in a signalized network, or be defined based on the basic infrastructure elements such as a track section or a point.

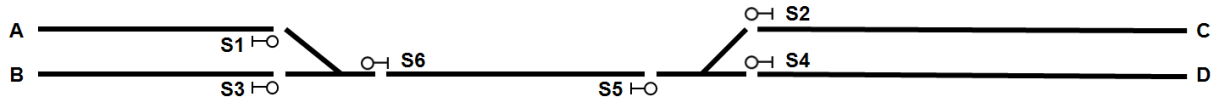



**Fig. 3-1 Resource Definition Based on Block Sections**

Fig. 3-1 illustrates a simple signalized network with 4 entry/exit points and 6 signals, and the infrastructure resources are defined based on the block sections. A block section is named by its starting and ending points. For example, the block section from entry point A to signal S1 is named as A-S1. The infrastructure resources (block sections) inside the network shown in Fig. 3-1 are: A-S1, B-S3, S1-S5, S3-S5, S5-C, S5-D, C-S2, D-S4, S2-S6, S4-S6, S6-A, and S6-B.

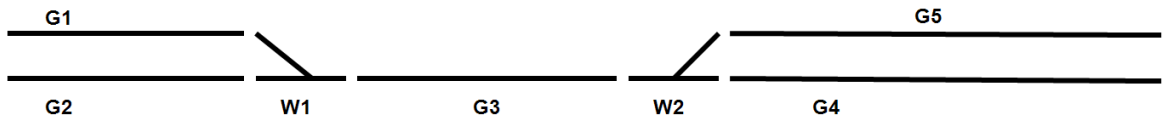



**Fig. 3-2 Resource Definition Based on Infrastructure Elements**

The definition of infrastructure resources based on infrastructure elements is mostly used in the non-signalized operation or the situation that the information of signaling system is not available. A network without signaling system is shown in Fig. 3-2. The infrastructure resources can be defined simply based on the infrastructure elements, which are categorized as track sections and junctions. A *track section* can be physically one track or several connected tracks, and it has only two entry/exit points. A *junction* can be a point or a crossing, and the element has more than 2 entry/exit points. In Fig. 3-2, the infrastructure resources with the element type of track section are G1, G2, G3, G4, and G5; and W1 and W2 are the infrastructure resources de-

fined by a junction-type element. In this dissertation, tracks are named with the prefix “G” (the first letter of the word *Gleis*, German for “track”), and points are named with the prefix “W (the first letter of the word *Weiche*, German for “point”).

Although infrastructure resources can be defined based on block sections in a signalized network, it is still possible to determine if an infrastructure resource includes a junction type element or not. Therefore an infrastructure resource can be classified as a *junction type resource* (JR, if a junction-type element is included) or a *non-junction type resource* (NJR).

An infrastructure resource is requested and allocated as a basic unit in the simulation. An important principle is that a train is not always free to enter an infrastructure resource. For instance, in a signalized network with regular operations, one block section is only allowed to be occupied by one train. That means the infrastructure resource is blocked when it is occupied or pre-allocated by a train. Additionally, other infrastructure resources (block sections) conflicting with the block section to be driven (according to the internal logic of the interlocking system) are also blocked. These additional blocked resources are called exclusive resources. Therefore, a list of exclusive infrastructure resource should be maintained for avoiding possible conflicts. Table 3-1 shows the list of the exclusive resources for each infrastructure resource of the network in Fig. 3-1.

| Infrastructure Resource | Exclusive Resources |
|---|---|
| A-S1 | S6-A |
| B-S3 | S6-B |
| S1-S5 | S6-A; S6-B; S3-S5; S2-S6; S4-S6 |
| S3-S5 | S6-A; S6-B; S1-S5; S2-S6; S4-S6 |
| S5-C | C-S2; S2-S6; S4-S6 |
| S5-D | D-S4; S2-S6; S4-S6 |
| C-S2 | S5-C |
| D-S4 | S5-D |
| S2-S6 | S5-C; S5-D; S4-S6; S1-S5; S3-S5 |
| S4-S6 | S5-C; S5-D; S2-S6; S1-S5; S3-S5 |
| S6-A | A-S1; S1-S5; S3-S5 |
| S6-B | B-S3; S1-S5; S3-S5 |

**Table 3-1 List of Exclusive Resources for the Network in Fig. 3-1**

The blocking rule is application-dependent. Taking an example in shunting operation, a train can enter an infrastructure resource occupied by another train when these two trains are supposed to be coupled together later. Although the blocking rule of a resource varies, the authorization of entering an infrastructure resource is always required. The core of the simulation workflow is related with the strategy of requesting and allocating resources according to certain blocking rules.

Currently, signalized operation is the most common form of operation. The definition of infrastructure resources based on block sections is more practical compared with the definition based on infrastructure elements. However, the same principle of the simulation workflow is applied for both definitions. To illustrate the workflow clearly with simple diagram and little number of infrastructure resources, in this dissertation the definition based on infrastructure elements is utilized for most of the examples.

### 3.1.2 Simulation Performers

A simulation performer (from this point on in this dissertation, the term “performer” refers to “simulation performer”) can be a train, a set of railroad cars, or a shunting set. A train consists of one or more locomotive with or without railroad cars, and a railroad car is a vehicle for carrying passengers or cargo. In passenger transport, railroad cars also can be single or multiple units. A shunting set is a set of vehicles with or without a locomotive, which is possibly coupled with other shunting sets or decoupled into more shunting sets in shunting operations.

In a simulation workflow, simulation performers act as the basic entities to request and occupy infrastructure resources. Therefore, an infrastructure resource requester/occupier is a simulation performer. During the simulation process, each infrastructure resource maintains a queue of requesters and a list of occupiers, which are updated regularly by a certain time interval according to the simulation logic. The details of requesting and allocating infrastructure resources will be discussed in Section 3.2 and Chapter 4.

Simulation performers are generated when they are entering the observed railway network. The performers existing in the network at the beginning of the simulation are also created when the simulation is initialized. The lifecycle of a performer is terminated when this simulation performer is out of the network or the simulation process is finished. Particularly, when a simulation performer is out of service in operation, not only the simulation performer but also the infrastructure resource occupied by that

performer is not available. In this situation, the setting of unavailable status for the infrastructure resource can be achieved by inputting an unavailable time frame with the same time period as the damage of the performer (the concept of time slot is explained in Section 5.3).

When a train runs inside the network, it has a route that specifies the sequence of the infrastructure resources the performer will pass by and the destination of the movement. It is possible that one train has many alternative routes to reach its destination. The route with the shortest running time has the highest priority.

### 3.1.3 Simulation Tasks

A simulation task describes the action expected to be completed for a simulation performer in the operation. It is derived from a predefined timetable or other operation concepts. Simulation tasks are initialized before a simulation process starts, and they are dispatched by the simulation workflow. A simulation task always includes one simulation performer. The time-related attributes usually are recorded, including starting time, accumulated execution time of the simulation task and the delay compared with the scheduled time.

There are two kinds of simulation tasks: movement tasks and non-movement tasks. A movement task specifies the movement of a train with a pre-defined destination, and a non-movement task defines the action for a simulation performer without changing its location.

In a normal operation, usually the whole movements of a train in the observed area are defined as one movement task, and all the planned stops are included in this movement task. For example, when a train is planned to run from station A to station C and halt in stop B. The stop B and the stopping time on stop B are specified also in the movement task. During the execution of a movement task, there might be also some unplanned stops due to deviations from timetable. These unplanned stopping actions will not be defined in the simulation tasks since they are unpredictable before a simulation starts. The unplanned stopping time will be added to the accumulated execution time of the movement task.

In a shunting operation, a shunting set is often decoupled or coupled during the whole process. It will be convenient to define a movement task based on each leg of the movement without introducing any predefined stops. For example, a shunting set S0 runs from track section A to track section B, then it will be divided into 2 parts:

shunting set S1 and S2, at last the shunting set S1 runs to track section C and the shunting set S2 starts to load goods. The simulation tasks of the whole process will be defined as:

- S0 runs to track section B (Movement task).
- S0 is decoupled into S1 and S2 (Non-movement task).
- S1 runs to track section C (Movement task).
- S2 loads goods (Non-movement task).

The lifecycle of a simulation task includes three states: created, running and terminated. The state diagram of the lifecycle is shown in Fig. 3-3:

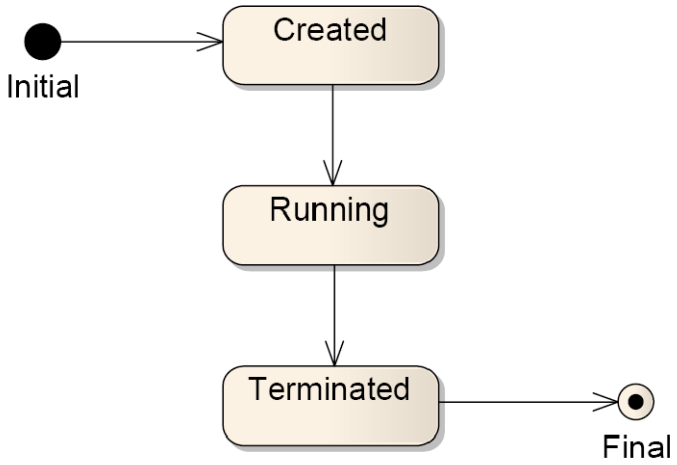



**Fig. 3-3 The State Diagram of a Simulation Task**

#### 3.1.3.1 Task State – Created

A simulation task is pre-defined before the simulation, and it is loaded when simulation starts. At this moment the task state is set as “created”. The simulation task with the state of “created” is not allowed to be executed. The preconditions to execute a simulation task will be checked to determine if this simulation task is ready to run. The preconditions are application dependent, the most common preconditions for starting simulation tasks are:

- A simulation task can only start after an absolute time (according to the timetable).
- A simulation task can only start until the performer is entering the simulation area.
- A simulation task can only start until the performer finishes other simulation tasks those are defined to be executed before this simulation task.

- A simulation task only can start if the shunting set is built correctly (for shunting process).

#### 3.1.3.2 Task State – Running

As soon as the preconditions to execute the simulation task are satisfied, the task state is set as “running”. That means the simulation task can be executed and the attributes of the performer, such as the position of the train, the accumulated execution time of this simulation task, the route of the train or the weight of the goods of the train, is updated during the execution.

Sometimes a simulation task has to be suspended if the further execution is not possible. One example is a train trying to enter an infrastructure resource that is not free to be entered. The train is waiting until the requested infrastructure resource is free again. The suspending and resuming state are not defined here, while they are integrated with the running state. The details of the execution of a simulation task with running state will be discussed in the Section 3.2.

#### 3.1.3.3 Task State – Terminated

The conditions for terminating a simulation task are checked before each execution of the simulation task. Once these terminating conditions are satisfied, the task state is set as “terminated” and the execution of the simulation task is accomplished. The terminating conditions depend on the type of the simulation task and the application context. The most common terminating conditions are:

- A simulation task will be terminated if the simulation performer is arriving at its destination (for a movement task).
- A simulation task will be terminated if the weight of the goods of the performer reaches a specified value (for a non-movement task, loading/unloading goods for freight transport).
- A simulation task will be terminated if the accumulated execution time reaches a specified value. (for a non-movement task)
- A simulation task will be terminated at an absolute time point. (for non-movement task)

Infrastructure resources, simulation performers and simulation tasks are the data basis of synchronous simulation. These components reflect the static aspect of the simulation model. Once a simulation starts, the components interact with others, and

the status and attributes of the components are updated driven by the simulation logic. The process of the synchronous simulation is described in Section 3.2.

## 3.2 The Workflow of Synchronous Simulation

The diagram of the workflow for synchronous simulation is shown in Fig. 3-4. A simulation starts after all simulation components are initialized. It is also necessary to set the simulation terminating conditions. The introduction about starting and terminating a simulation is given in Section 3.2.1. A synchronous simulation is composed of a series of single process steps triggered by a certain time interval. In Section 3.2.2, the execution of a single processing step is discussed in details.

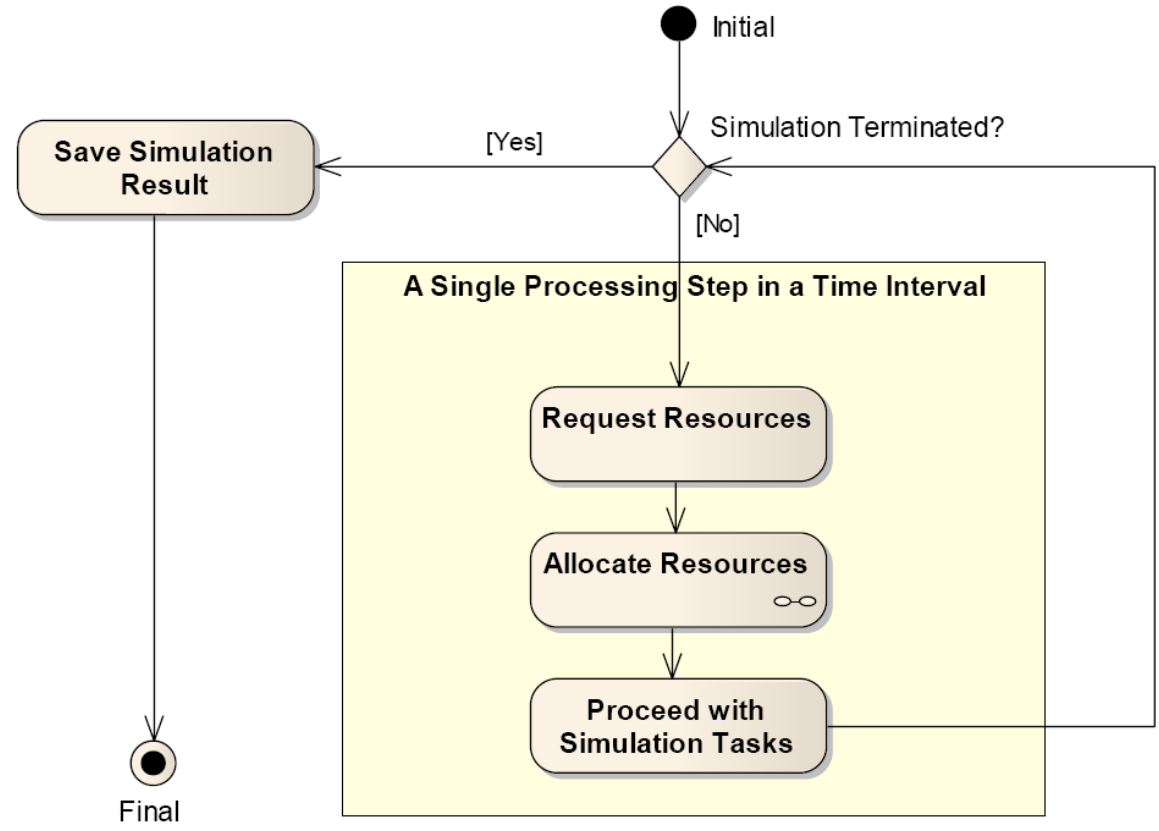



**Fig. 3-4 The Workflow of Synchronous Simulation**

### 3.2.1 Initialization and Termination of Simulation

The inputs of a simulation process are the simulation components described in Section 3.1. The initialization starts from constructing infrastructure network, in which all the infrastructure resources are initialized. At this moment, the requester queue and the occupier list of each infrastructure resource are empty. After the infrastructure network is built up, the simulation performers are loaded, and each of the simulation

performer is added to the occupier list of its blocked resource. At last the simulation tasks are initialized and all the states of these simulation tasks' are set as "created".

A simulation process will be terminated if the pre-defined terminating conditions are satisfied. A simulation process can be terminated:

- At an absolute time point;
- After the total simulation time exceed the predefined time period;
- When all the simulation tasks are terminated;
- When none of the simulation tasks can be executed furthermore (simulation failed);
- By the user.

The outputs of a simulation process can be a new generated timetable, a time-distance diagram, a blocking time "stairway", or other forms of blocking diagram for infrastructure resources. The events occurred during the simulation process are also logged.

### 3.2.2 A Single Processing Step of Synchronous Simulation

Once a simulation starts, the process is triggered by a certain time interval. A single processing step is executed in each time interval iteratively until the whole simulation is terminated (see Fig. 3-4).

The execution of a single processing step is simulation task oriented. Inside a single processing step, the states of all the simulation tasks are checked, and only the simulation tasks with the state of "running" are put into execution. It is possible that the state of a simulation task is changed after the execution of one single processing step. Therefore, the states of all simulation tasks will be updated at the end of each time interval or at the beginning of the next time interval.

Before proceeding with a movement task in the current time interval, it is necessary to ensure that all the required resources (the resources supposed to be occupied) are possible to be allocated to the performer of the simulation task. The activities in a single processing step are: request resources, allocate resources and proceed with simulation tasks. They are executed inside of a single processing step in sequence.

#### 3.2.2.1 Request Resources

Only movement tasks are concerned in requesting resources, since the performers of non-movement tasks do not intend to occupy new infrastructure resources. For a

movement task, the resources expected to be occupied or be allocated in the current time interval should be found out, and the performer of the simulation task will be added to the requester queue of the requested resources. Meanwhile, the requested resources are recorded in the simulation task (It will be useful in “proceed with simulation tasks”, see Section 3.2.2.3). A performer may request to occupy more than one resource. If a resource has already been blocked by a performer, it is not necessary to add this performer to the requester queue.

To find out the requested resources in the current time interval, the route of the simulation performer should be determined at first. Since the destination of a movement task has been defined, the route of the train can be obtained by a route searching algorithm or by the predefined train path in the timetable. It should be noted that the route may be altered if necessary. The decision of changing the current route will be made when allocating resources. Hence the requested resources are also changed. The process of using an alternative route will be discussed in Section 4.4.3.

Obviously, it is not necessary to request and reserve all the infrastructure resources of the whole route. Therefore the least required resources in the current time interval needs to be defined exactly to avoid the idleness of the infrastructure resources. The definition of the least required resources is highly related with the aspect of railway operation.

A very intuitive idea of determining the least required resources is to predict the train running distance in the current time interval with maximum running speed, and the resources covered inside this distance are the least required resources. However, there is a problem in rail operation as shown in the following example:

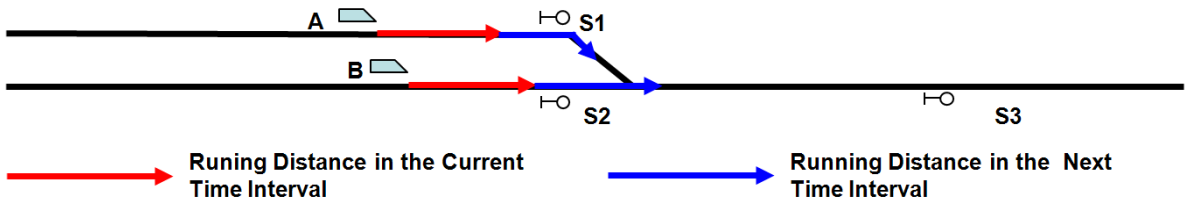



**Fig. 3-5 Requesting Resources without Considering the Braking Distance**

When train A and train B are requesting resources for the current time interval, the required resources are determined based on the distance (marked with red line) that each train runs in the current time interval with maximum speed respectively. At this time interval each train has no new required resource since each of them is before the next signal. That means all trains are running normally without breaking. In the

next time interval, the distance for each train is calculated in the same way and marked with blue lines. Both of the trains are requiring entering the next block section. Assume that train B gets the chance and train A has to wait before the signal S1. But the conflict between train A and train B occurs as long as the stopping distance of train A is too long to keep train A stopped before signal S1!

To solve this problem, the distance for the forecast of train movement should be carefully calculated with the consideration of running dynamics and signaling. A feasible solution is to calculate the distance as:

$$D_{\min} = D_{\mathrm{r}} + D_{\mathrm{s}} \tag{3-1}$$

Notations used:

$D_{\min}$: Predicted minimum requesting distance in the time interval

$D_{\mathrm{r}}$: Running distance in the time interval

$D_{\mathrm{s}}$: Stopping distance at the maximum speed

The distance for requesting resources is extended by introducing the stopping distance. A train requests the additional resources covered by the stopping distance, and it starts to brake at the latest stage when it cannot get the chance to enter the new required infrastructure resources. It can be proven that even in the worst case the safety is still be ensured. When the time interval approaches to zero, the distance to the next stopping point (in a signalized network, it is the next signal) reaches its minimum value that equals the braking distance. It means that the train is still able to stop before the critical area if it does not obtain the permission to enter.

For a simulation task, it is not guaranteed that the requested resources are allowed to be allocated to the performer. It depends on the result of allocating resources (see Section 3.2.2.2), which also influences the behavior in proceeding with the simulation task (see Section 3.2.2.3).

#### 3.2.2.2 Allocate Resources

After requesting resources, a resource may be requested by one or more requesters. Not all the requesters have the authority to enter the requested resources. During allocating resources, the resources allowed to be entered are assigned to their requester according to a certain dispatching strategy. The approach of allocating resources in a simulation workflow is similar to that in railway operation and control, which is centralized and all simulation components are coordinated and dispatched

within one business logic consistently. Fig. 3-6 illustrates the procedures of allocating resources.

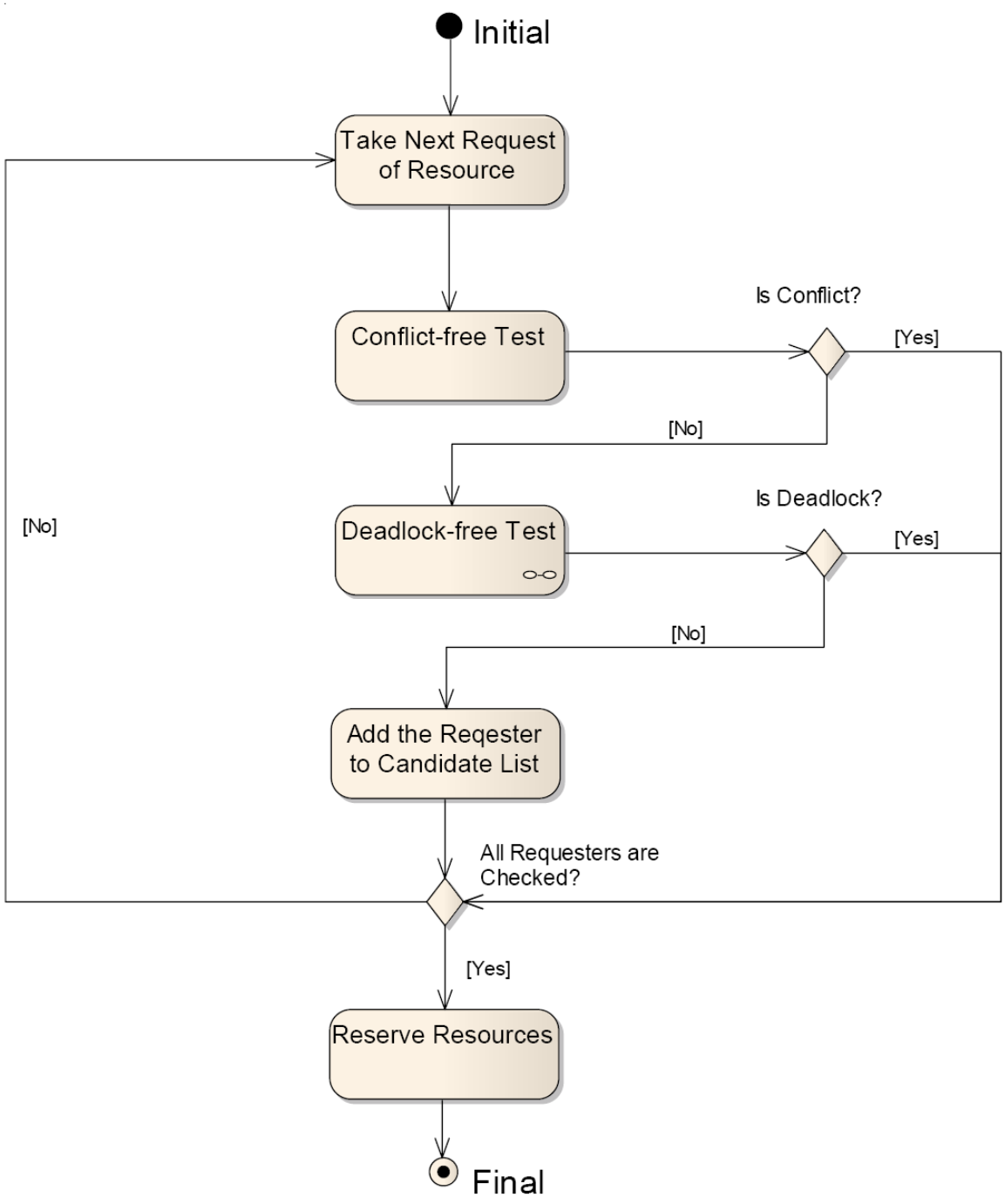



**Fig. 3-6 The Procedures of Allocating Resources**

The purpose of allocating resources is to ensure the system is conflict-free and deadlock-free. Conflict-free is the most basic safety requirement in railway operation, and deadlock-free is a necessary condition for preventing simulation failure, the definition of the deadlock will be given in Section 4.1.

The resource-requester pair is checked and analyzed one by one until all the resource-requester pairs are checked. A requester is the simulation performer of a movement task (see Section 3.2.2.1). Only if the conflict-free test and deadlock-free test are passed, the requested resource can be allocated to the requester.

The criteria for conflict-free test depend on the application context. The most common situations that lead to conflicts are listed as follows:

- The requested infrastructure resource is blocked by another performer, and the requester is not allowed to enter the requested resource.
- The exclusive resources (for definition of exclusive resources see Section 3.1.1) of the requested infrastructure are blocked by other performers, and the requester is not allowed to enter the exclusive resources.
- The requested infrastructure resource and the exclusive resources of the requested resource are not allowed to be entered, since the timeslots (see Section 5.3) for those infrastructure resources are not available at the current time.

Once entering the requested resource leads to conflicts, the current resource-requester pair is ignored. The deadlock-free test will only be carried out for the pairs that passed the conflict-free test. The deadlock-free test will be explained in details in Chapter 4. Similar to the conflict-free test, only the pairs that passed the deadlock-free test are concerned.

A requester that passed both conflict-free test and deadlock-free test is possible to enter the requested resource, and such a requester is called a candidate. A candidate will be added into a candidate list, which is maintained for each infrastructure.

After all the resource-requester pairs are tested and analyzed, the candidate lists for all the resources are also updated. For each infrastructure resource, if the candidate list is not empty, one of the candidates will be selected from the list as the occupier of the infrastructure resource. The rule of selecting an occupier is up to the dispatching strategy. The determination of train priorities will be discussed in Chapter 5.

Finally, the requester queue and the candidate list are cleared when allocating resources for the current single processing step is finished. The requester queue and the candidate list are initialized in each simple processing step, and the occupier list is only initialized at the beginning of a simulation process.

Allocating resources is the most sophisticate activity in the whole simulation process. The deadlock avoidance and train priorities determination are included here. The re-

liability and the efficiency of the simulation model are highly depending on these algorithms. Therefore, allocating resources is the core of the whole simulation process.

#### 3.2.2.3 Proceed with Simulation Tasks

After infrastructure resources have been requested and allocated for all the movement tasks, all the simulation tasks are executed in the current time interval. Essentially, an execution of a simulation task in a simple processing step is to update the information of the simulation task performer, the related infrastructure resources and the simulation task itself.

For the performer of a movement task, the new position and running speed of the performer after the running in the current time interval is the most important attribute need to be updated, which will be calculated by a running time calculation module. The driving style of the performer is required as the basis input for the calculation. If a running train is permitted to enter the next requested infrastructure resources, the new position will be calculated with normal driving style without hindrance; otherwise, the train will be stopped at a calculated position with braking. If a train is already stopping at a position because of not having got the authority to enter the next requested resources in previous processing steps, it will start acceleration if the requested resources are obtained, or keep stopping if the resources are not available. As mentioned in Section 3.2.2.1, each movement task maintains a list of requested resources, and the blocking situation can be determined via comparing the occupier of the requested infrastructure resources and the simulation task performer. It is also necessary to update the blocking situation of infrastructure resources after a single step of execution is finished. When the performer of the simulation task leaves the blocked infrastructure resource, the resource will be released by that performer. The performer will be removed from the occupier list of that resource.

A non-movement task does not influence the attributes of the infrastructure resources, while the status of the performer, such as the weight of the goods or the coupling status of the vehicles, will be updated according to the type of the simulation tasks.

The updating of the simulation task follows after the performer and the related infrastructure resources are updated. The accumulated execution time will be increased by the current time interval, and the delay of simulation task is re-evaluated. A single processing step in a simulation time interval is accomplished at this moment. As

mentioned at the beginning of the Section 3.2.2, the state of a simulation task may be changed at the end of each processing step, and it will be updated here or at the beginning of the next time interval.

## 3.3 Event-Driven Simulation

Event-driven simulation is a very popular technique for synchronous simulation. It reduces the overhead of a simulation system and improves the system performance. A dispatching system is a real-time controlling system, and the system performance is one of the most decisive aspects. Therefore, the event-driven simulation is particularly suitable for the simulative dispatching approach.

### 3.3.1 The Principle of Event-Driven Simulation

As described in Section 3.2, the process of a synchronous simulation is triggered by a certain time interval. An easiest way to define the time interval is to use a fixed time interval, in which each single processing step is executed in a fixed time interval regularly. Such a definition is often utilized in case of visualizing a simulation process with a graphical user interface. In each step the most detailed movements are concerned and demonstrated.

The model of using a regular time interval is inefficient when the most trivial details are not interesting. For instance, the desired output of a simulation in a dispatching system is the rescheduled timetable. The detailed movements of a train in each time interval, such as a smallest position change in the same block section, are not necessarily to be known outside of the simulation. In an execution of a single processing step, the status of the simulation components is updated. But only the updates that change the status of other system components are interesting. Repeating executing some processing steps that do not influence the system status will result in unnecessary system cost.

Event-driven simulation introduces the concept of “event” to realize a flexible time interval definition. The system interactions are predicted, and they are defined in the form of “event”. An event represents an occurrence of a possible system status change in a specific time point. The definition of events is application dependent, and the most common events in a simulation system are:

- The state of a simulation task changes.
- A system performer reaches at a position, at which a requesting for new infrastructure resources is required.
- A system performer passes a position, at which the blocked infrastructure resource can be released.

An event list is maintained in the model (see Section 3.3.2), and the execution of each single process step only occurs at the time that defined by the events. Therefore, the system interactions are organized deterministically, and the simulation system runs in an efficient way.

### 3.3.2 Event List and Event Processing in Synchronous Simulation

To implement an event-driven simulation, an event list is required to store system events in the order of the occurring time. In a simulation process, the execution of a single processing step is triggered based on the event list, and the event list is updated after each execution of one step. The workflow of a synchronous simulation using the event-driven approach is shown in Fig. 3-7.

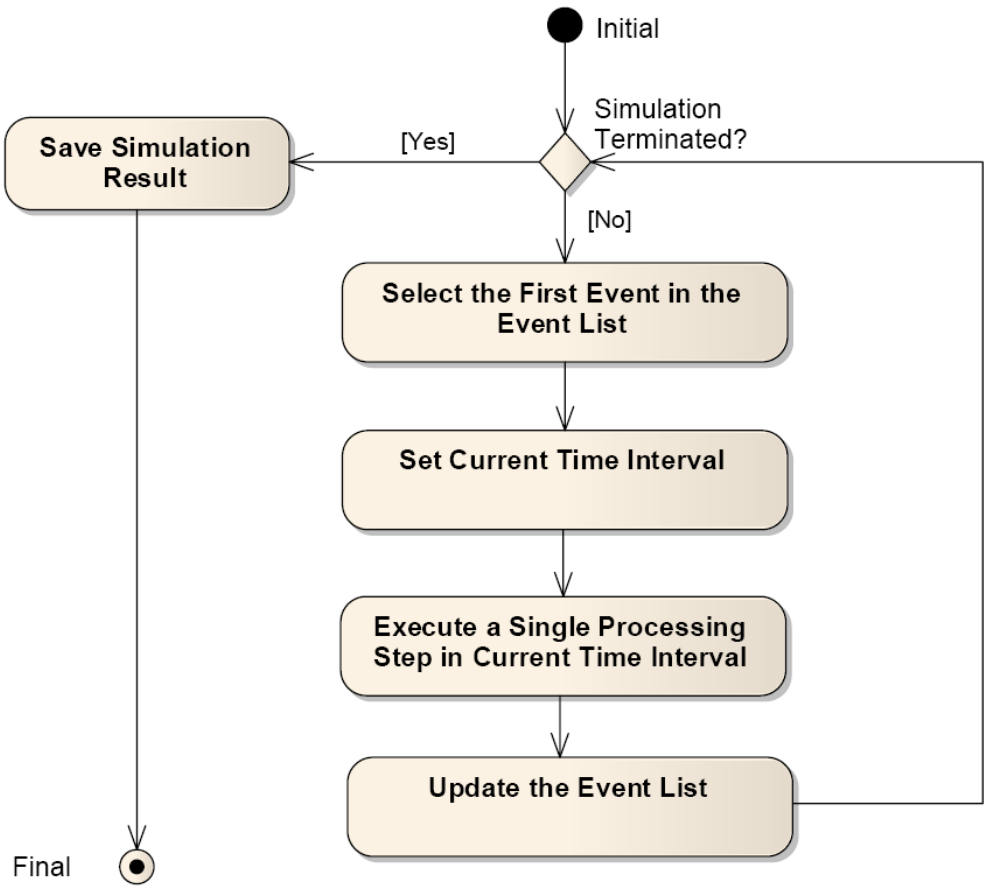



**Fig. 3-7 The Workflow of Event-Driven Simulation**

The initialization of the event list starts after the simulation components are initialized (the initialization of simulation components is described in Section 3.2.1). The simulation tasks will be scanned. For a simulation task with the state of “running” (that means the task state has been changed from “created” to “running”), an event will be added to the event list respectively.

In an event list, the times of the events are sorted in ascending order. The time of the first event will be set as the current simulation time. A single processing step for the current time interval is executed as described in Section 3.2.2. The status of all the simulation components will be updated as well.

It is necessary to update the event list at the end of each processing step, in order to ensure the events are managed correctly for next iterations. It includes to remove processed events and to insert new events.

- To remove processed events: It can be achieved by removing the events with the occurring time that is same as the current simulation time. The events with an earlier occurring time should be removed from the event list in previous processing steps.

- To insert the new events: The simulation tasks with the state of “created” or “running” are examined. The new events will be derived from changing of task state, requesting of new infrastructure and releasing of blocked infrastructure. These new predicted events will be inserted into the event list. For a movement task, the module of running time calculation is required to forecast the time of the event. For the same event, the prediction of the event time should be repeated in each single processing step, since the current system status may be different with the system status that is used for the calculation of the event time in last processing steps.

After the event list is updated, the system is preparing to execute the next processing step until the terminating condition of the simulation is satisfied. The terminating condition can also be determined according to the status of the event list: the simulation will be finished if the event list is empty.

Synchronous simulation is the basic form of simulation. The involved components of the synchronous simulation are infrastructure resources, simulation performers and simulation tasks. The simulation process is driven by a certain time interval, while the performance can be improved with the mode of event-driven simulation. Allocating resources is the most sophisticate activity in a simulation process, and deadlock should be avoided before an infrastructure resource is allocated to a simulation performer. The algorithm of deadlock avoidance will be discussed in Chapter 4.

# 4 Resolving Deadlocks in Synchronous Simulation

In a synchronous simulation, to resolve deadlock problems is a necessary feature to prevent simulation failure. In the field of railway operation, a deadlock is defined as follows (see [PACHL, 2007]):

“A deadlock is a situation in which a number of trains cannot continue their path at all because every train is blocked by another one.”

A simple example of a deadlock situation is shown in Fig. 4-1 (as mentioned in Section 3.1.1, to reduce the complexity, the infrastructure of the example used in this chapter is defined based on infrastructure elements, and the definition based on block sections is not used here.).

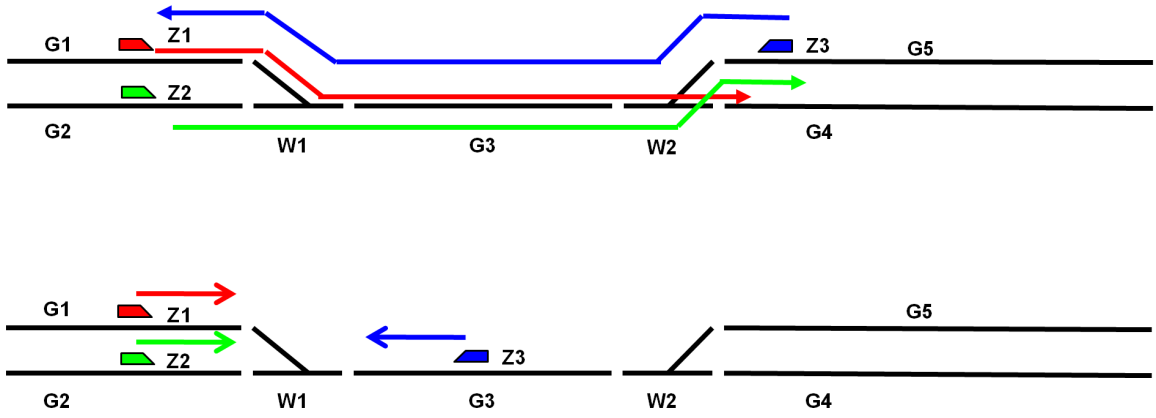



**Fig. 4-1 An Example of Deadlock**

In this dissertation, trains are named with the prefix “Z” (the first letter of the word *Zug*, German for “train”). Train Z1, Z2 and Z3 are going to run along the route marked in red, green and blue lines respectively. The correct sequence of the train movements should be: Z1, Z3 and Z2. Assume they ran in another order as shown in Fig. 4-1: train Z3 instead of train Z1 gets the chance to reach track G3. Train Z1 is waiting train Z3 to leave track G3 whereas train Z3 is waiting train Z1 to leave track G1. Hence train Z1 and train Z3 are blocked by each other and a deadlock situation occurs.

It is a challenge to find a practical solution to resolve deadlock problems for railway simulation of the operating process. In this chapter, the necessary conditions that lead to deadlocks in synchronous railway simulation are listed, with the introduction

of possible approaches for resolving deadlock problems. Different deadlock avoidance algorithms, including movement consequence analysis (MCA), dynamic route reservation (DRR), Petersen and Taylor algorithm, and labeling algorithm, are introduced. In this dissertation, the Banker's algorithm, which is developed originally for computer operating system, is suggested and adapted for railway synchronous simulation (especially for the dispatching system in complex network). Some further improvements of the Banker's algorithm with special considerations for railway operation are given as well. The software implementation and the system performance for synchronous simulation and the Banker's algorithm are evaluated at last.

## 4.1 Deadlock Problem and Approaches for Resolving Deadlocks

In contrast to asynchronous simulation, deadlocks may be produced if there are track sections with bidirectional operations in synchronous simulation (see [PACHL, 2007]). In addition to railway synchronous simulation, the deadlock problem also occurs in other fields such as computer science and telecommunication. There are four necessary conditions (known as COFFMAN conditions, see [COFFMAN et al., 1971]) that lead to deadlocks. These are:

1) The "mutual exclusion" condition: Tasks claim to the requested resources exclusively. In a railway simulation, that means the infrastructure resources are not allowed to be entered when it is blocked by another simulation performer (except for shunting process, e.g. when two shunting sets are supposed to be coupled, they are allowed to enter the same infrastructure resource.).
2) The "wait for" condition: Tasks hold the blocked resources when waiting for the new required resources. In a railway simulation, a simulation performer may wait until the required resources are released, while the resources blocked by the performer are not allowed to be allocated to other trains.
3) The "no preemption" condition: A resource cannot be forcibly removed from the occupier until it is released by the task explicitly. This condition is also fit in railway operation.
4) The "circular wait" condition: Two or more tasks form a circular chain, in which each task holds one or more resources requested by other tasks. In the example shown in Fig. 4-1, train Z1 and Z3 form such a circular chain, each train requests the resource blocked by the other one.

The approaches for resolving deadlock problem can be categorized as deadlock detection, deadlock prevention, and deadlock avoidance. These approaches are originally developed for computer operating system (see [SILBERSCHATZ et al., 2005]). Some of the principles may also be applied in railway operation. The principles of resolving deadlocks in the context of railway operation are explained and compared in Section 5.2 in [PACHL, 1993].

- **Deadlock detection** is to start and execute a simulation process without considering the deadlock problem at first. A periodical detection is arranged to test if a deadlock has been produced (the detection of an occurred deadlock can be implemented easily by checking the circular wait condition). Once a deadlock situation occurs, the system simulation rolls back and one or more tasks restart again in order to recover the system from deadlocks. In most cases the recovery from deadlock is achieved through removing the resource that leads deadlock from its occupier (preemption). Deadlock detection is not suitable for solving deadlocks during railway operation, in which the “no preemption” condition is always in effect. In addition, a rollback operation is hard to be supported in a synchronous simulation. Therefore the approach deadlock detection will not be discussed in this dissertation.
- **Deadlock Prevention** is to examine the four necessary conditions of deadlocks and to ensure that at least one of them will never be satisfied. For a railway synchronous simulation, the conditions 1), 2), and 3) are always valid, and then the key of deadlock prevention is to prevent the system from falling a circular wait situation. To prevent the circular wait situation systematically, additional constraints and rules will be introduced. The possible constraints and rules include to forbidden bidirectional operations, or to reserve all the required infrastructure resources in advance for every trains. Although these additional introduced constraints and rules can eliminate deadlocks absolutely, it is unpractical to be applied in railway operation because of the low efficiency and low infrastructure utilization level. The applications of deadlock prevention will not be covered in this dissertation.
- **Deadlock Avoidance** is to examine the system state dynamically before allocating a resource to a requester. A request can only be granted if the system will still keep in a safe state. A safe state means the resources can be allocated to each task in some order without leading to deadlock. In this disser-

tation the test for system state is also called “deadlock-free test”. To execute a deadlock-free test, the potential usage of resources must be known in advance, this is the most important pre-condition for deadlock avoidance. Many different deadlock avoidance algorithms will be explained in Section 4.2 and Section 4.3.

The approach of deadlock prevention only concentrates on eliminating at least one of the four necessary conditions of deadlocks, while these conditions do not sufficiently lead to deadlocks. Compared with deadlock prevention, deadlock avoidance is more flexible. As long as a request does not lead the system to an unsafe state, the requested resource can be allocated to the requester, even if all four necessary conditions of deadlock are satisfied. Therefore, deadlock avoidance can deal with deadlock problems with a higher resource utilization level than deadlock prevention. The difference between deadlock prevention and deadlock avoidance is “similar to the difference between a traffic light and a police officer directing traffic” (see [AMIR, 2000]).

Although deadlock avoidance can improve the network throughput, it is still not a perfect solution. The Information of potential usage of infrastructure resources is not always available. More important, a typical characteristic of deadlock avoidance should be considered: a safe state can ensure the operation being out of deadlocks, however, an unsafe state does not always sufficiently lead to deadlocks. For deadlock-free tests, the situation called “*false positive*” may occur: a deadlock-free test result that is read as positive but actually is negative. A false positive test shows evidence of a deadlock when it not actually present. In Section 4.2 and Section 4.4 some examples of false positive are given.

There is not a universal solution to solve deadlock problems entirely for railway synchronous simulation. An overview of the research is given in [PACHL, 2007], with the conclusion that “the development of solutions for deadlock avoiding in synchronous simulation or real-time dispatching system is still a big challenge”. In the following sections of this chapter, the algorithms and applications for deadlock avoidance are explained. The advantages and disadvantages of these algorithms will prove PACHL’s conclusion again.

## 4.2 Algorithms for Deadlock Avoidance in Railway Operation

Movement consequence analysis (MCA) and dynamic route reservation (DRR) are algorithms based on the principle of deadlock avoidance found by PACHL. MCA was developed in early 1990s and DRR was the further development of MCA (see [PACHL, 1993] and [PACHL, 2007]). To keep the terminology unified, the terms used in Section 4.2 follow the definitions for simulation components defined in Chapter 3. Other deadlock avoidance algorithms (developed for Australia's railway), including Petersen and Taylor algorithm, and labeling algorithm, are introduced in Section 4.2.3 and 4.2.4.

### 4.2.1 Movement Consequence Analysis (MCA)

The basic idea of MCA is to analyze the consequences of train movements and to avoid circular wait situation based on the analysis. In MCA two types of consequences are defined:

- Primary Consequences: Movements that must be made to enable a performer to enter an infrastructure resource are called primary consequences.
- Secondary Consequences: Movements that must be made after a performer has entered an infrastructure resource are called secondary sequences.

According to the definition, the events for a performer to enter a new resource take place in the following order: first the primary consequences, next the allocation of the resource and finally the secondary consequences. Each movement of primary/secondary consequences will lead to new primary and secondary consequences. A request of entering a resource will be observed as a consequence tree shown in Fig. 4-2.

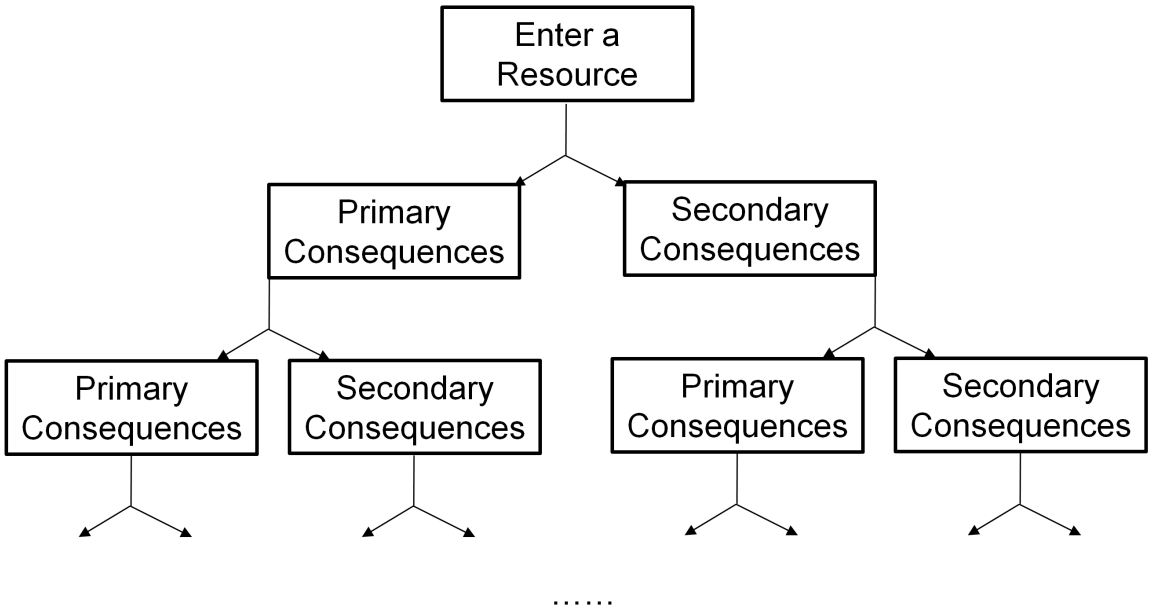



**Fig. 4-2 The Consequence Tree in MCA**

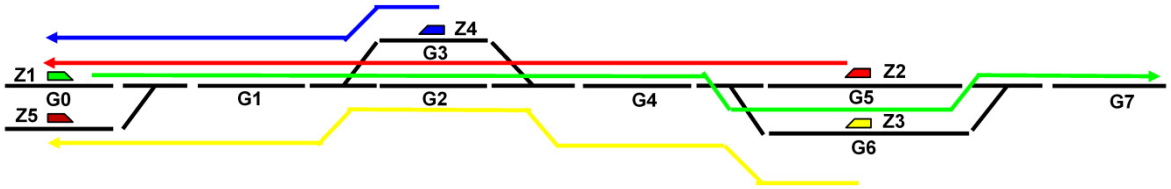
Once a request to a resource occurs twice in the tree, a circular wait condition is satisfied. The request for the resource should be denied to avoid a possible deadlock. Take the example shown in Fig. 4-1, when performer Z3 is requesting for resource G3 (with MCA, a request is made for the next track section), the analysis for the movements is:

Z3 (G3) → Z3 (G1)

Z3 (G1) → Z1 (G3) → Deadlock (G3)!

MCA can avoid every potential deadlock strictly. However, it has some disadvantages need to be overcome. A predefined route is required for each performer to construct the consequence tree, while in many cases it is impossible to determine the route in advance as the simulation operation is very flexible. This problem may be solved by introducing some rules as described in [PACHL, 2007]. Particularly, in a dispatching system, the train path has been defined in timetable, and a default route can be determined by some route searching algorithms.

A disadvantage of MCA is due to the strict rule during the deadlock-free test: a situation that fits the circular wait condition does not always lead to deadlocks. For this reason, many requests are refused unnecessarily and many extra stopping are introduced. It can be categorized as false positive problem, which results in low utilization level of infrastructure resources. An example of false positive test is shown in Fig. 4-3.




| Performer | Route |
|---|---|
| Z1 | G0, G1, G2, G4, G6, G7, out |
| Z2 | G5, G4, G2, G1, G0, out |
| Z3 | G6, G4, G3, G1, G0, out |
| Z4 | G3, G1, G0, out |
| Z5 | Stop |

**Fig. 4-3 An Example of "False Positive" in Deadlock-free Test**

If Z1 requests G1, the analysis of the movement consequences is:

Z1 (G1) → Z1 (G2)

Z1 (G2) → Z1 (G4)

Z1 (G4) → Z1 (G6)

Z1 (G6) → Z3 (G3)

Z3 (G3) → Z4 (G1) → Deadlock (G1)!

The result shows a cycle of requesting G1 occurs in the consequence tree. Therefore, Z1 is not allowed to enter G1. However, it will not necessarily lead to deadlock if Z1 enters G1. The feasible train movements can be arranged as follows:

1) Z1 enters to G2 via G1 and wait.
2) Z4 runs away, and Z3 runs away.
3) Z1 continues its trip and runs away, and finally Z2 runs away.

The problem of false positives may occur in other deadlock avoidance algorithms as well (the discussion of the problem for the Banker's algorithm can be seen in Section 4.4). In MCA it is also difficult to decide, how far of the train route should be observed in advance in a deadlock-free test. On the one hand, if the complete train route is taken into consideration, the possibility of false positives will be high; on the other hand, if too few track sections ahead are observed, it may lead to deadlocks. Further research is still required in order to use MCA to avoid deadlocks in a complex and large network.

From the point of view of software implementation, many synchronous simulation models do not fit with MCA very well. It is very complicate to introduce an additional

consequence tree to the simulation model. In addition, constructing and traversing the consequence tree may lead much processing time.
To reach a high resource utilization level and implement good integration with simulation software, dynamic route reservation (DRR) is designed based on some principles of MCA. This algorithm will be introduced in Section 4.2.2.

### 4.2.2 Dynamic Route Reservation (DRR)

Adopting some basic principles from MCA, DRR is developed to reduce the possibility of false positives within a route reservation model. To provide more flexibility than MCA in arranging train movements, the following simplifications are introduced in DRR (see [PACHL, 2007]):

- When a simulation performer requests to enter an infrastructure resource with bidirectional operation, except for a station track, the secondary consequences criterion is always in effect regardless of the actual locations of any opposing performers.
- When a simulation performer requests to enter a station track section, the secondary consequences criterion is never in effect regardless of the actual location of any opposing performers.

Station tracks are the main tracks and loops for passing or overtaking trains inside or between station limits. With the simplifications of DRR, the performers outside of the nearest station track section are not considered for deadlock avoidance. Therefore such a performer does not have to wait for an avoidable deadlock. This is based on the assumption that a deadlock can be handled within a station track section.
Another important characteristic of DRR is route reservation that is suitable for most of synchronous simulation models. The principle of route reservation is that a performer may be allowed to enter an infrastructure resource only if the resource is reserved for the performer. An infrastructure resource can be reserved for many requested performers, and only the performer at the first position in the stack is allowed to enter the resource. A consequence tree is not required here anymore. Instead a stacked reservations model is built to define the train sequence for each infrastructure resource. A set of rules is required to determine the train sequences and prevent deadlocks. The detailed rules used for the examples are described in [PACHL, 2007].
Compared with MCA, DRR provides a better integration with synchronous simulation software, and reduces the possibility of false positives for avoidable deadlocks. A

possible solution of the reserved routes for the same example shown in Fig. 4-3 is demonstrated in Fig. 4-4.

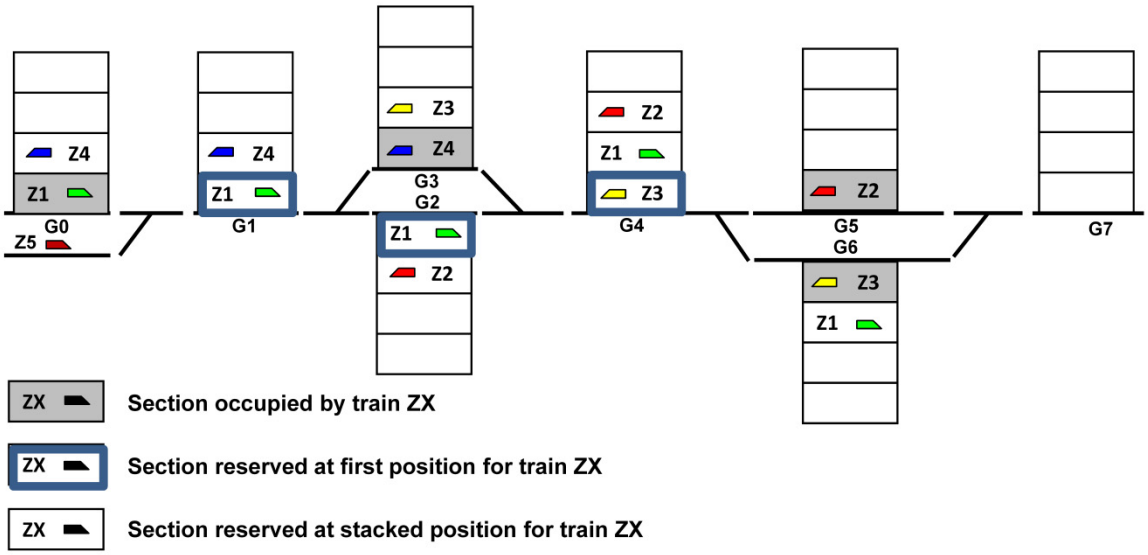



**Fig. 4-4 An Example of Deadlock Avoidance with DRR**

With the simplifications in DRR, the secondary consequences are handled flexibly if a train enters to station tracks. The operation can be preceded as long as train routes are reserved at each track appropriately. Therefore, the possibility of false positives will be reduced and the system efficiency will be improved. It should also be noticed that the risk of deadlocks still exists in DRR, since sometimes the simplifications may lead to deadlocks between the train inside station tracks and the opposite train ahead of the station tracks. Additional rules are developed to overcome the problem in the route reservation model. These rules can be applied flexibly according to different application contexts. The implementation cost of the rules should be considered, especially if these rules are combined with other principles (for instance, the decision principles in a rule-based dispatching system).

DRR can handle deadlock problems for regular train movements with high efficiency. It is suitable to be used in synchronous simulation for dispatching purpose. However, for the operations in a complex network (e.g. the shunting movements in a shunting yard), it is difficult to differentiate exactly station track sections and track sections in a microscopic simulation. In such a situation, it is worth to combine with the solution (such as MCA or the Banker's algorithm) that works in a general network.

### 4.2.3 Petersen and Taylor Algorithm

PETERSEN and TAYLOR developed their algorithm for railway line dispatching and simulation models (see [PETERSEN und TAYLOR, 1983]). It is a simplified solution for deadlock avoidance, in which only two opposing train fleets are considered in a network with single line tracks and crossing loops (similar to the station tracks in DRR, see Section 4.2.2). An arbitrary set of trains moving on one direction is called an inbound train fleet, and then the other set of trains is an outbound fleet. Train fleet types are defined as following:

"A train fleet is *simple meetable* if the trains in that fleet can be moved onto track segments in their direction of travel so that the opposing fleet can feasibly reach its destination.
A train fleet is *second-order meetable* if one of the trains in the opposing fleet can be moved so that the forward fleet becomes simple meetable".

As long as one of a train fleet is simple meetable, second-order meetable or higher-order meetable (a higher-order meetable can be identified by examining the deeper level of train movement consequences), the system will not be blocked by deadlocks. To avoid deadlocks, Petersen and Taylor algorithm is based on the following principle: a (simple/second-order/higher-order) meetable fleet will ensure that opposing fleet can reach its destination, and no circular wait situation occurs finally. The idea to identify simple, second or higher order meetable fleet is also similar to examining movement consequences in MCA. However, the time complexity increases exponentially for high-order meatability tests. The simple meetable test is executed by counting the trains in the fleet and released resources in the path (see [PETERSEN und TAYLOR, 1983]), which is also similar to the principles of the Banker's algorithm (see Section 4.3).
Petersen and Taylor algorithm is suitable to determine deadlock situation in a simple network with only single tracks and loops. The algorithm to determine simple meetability is efficient with low implementation costs. However, it is not able to deal with a complex networks, where more than two directions of train fleets exist. The false positives problem is also applied for Petersen and Taylor algorithm, especially only simple meetability is tested. In many cases, a fleet that is not simple meetable does not sufficiently lead to deadlocks.

### 4.2.4 Labeling Algorithm and Single-path Problem

The labeling algorithm (see [MILLS und PUDNEY, 2003]) is initiated from the ideas of job shop scheduling, developed by MILLS et al. in the Cooperative Research Center for Railway Engineering and Technologies (Rail CRC). In the labeling algorithm, the resources are the infrastructure resources, and the jobs are the movement tasks. Similar to the Petersen and Taylor algorithm, only simple networks with single lines and crossing loops are considered. By labeling the order of passing single lines, a feasible train schedule can be developed. The single lines are treated as nodes. The nodes represent critical resources that may influence system status.

In labeling algorithm, a safe move test is executed to examine the feasibility of a move. A safe move in the labeling algorithm is defined as:

1) Move to a loop that contains two or more unoccupied tracks.
2) Move to a loop that contains one unoccupied track and one or more opposite trains.
3) Move to a loop that contains one unoccupied track and no opposite trains, but the next move of the trains at the loop will lead to a safe move.

The definitions 1) and 2) for safe move ensure that all movement tasks are feasible. Either at least one free resource is left in the loop, or an opposite train is swapped with the tested train. Definition 3) is another form of potential state transition analysis (see Section 4.4.1).

Labeling algorithm is easy to be implemented with high computational efficiency. However, it is only suitable for typical networks with only single lines and crossing loops, and not applicable for a complex and general network structure (e.g. in a shunting yard).

In [BRAAKSMA, 2008], deadlock problems are analyzed, especially concerning the approaches of state transition graph and precedence constrains, which are used to represent the single-path deadlock problems. The job shop scheduling problem is investigated with the consideration of computational complexity and possible solving approaches. The results show that further research is required to find practical solutions in order to manage the realistic deadlock problems in these directions.

## 4.3 The Banker's Algorithm

The Banker's algorithm was developed by DIJKSTRA (see [DIJKSTRA, 1982]), and it is based on the approach of deadlock avoidance. As described in Section 4.1, the

principle of deadlock avoidance is to grant a request only if that request will not lead the system to an unsafe state. The deadlock-free test for the Banker's algorithm is shown in Fig. 4-5. The Banker's algorithm is used to test the system state for deadlocks in order to determine whether an infrastructure resource is allowed to be allocated to a requester.

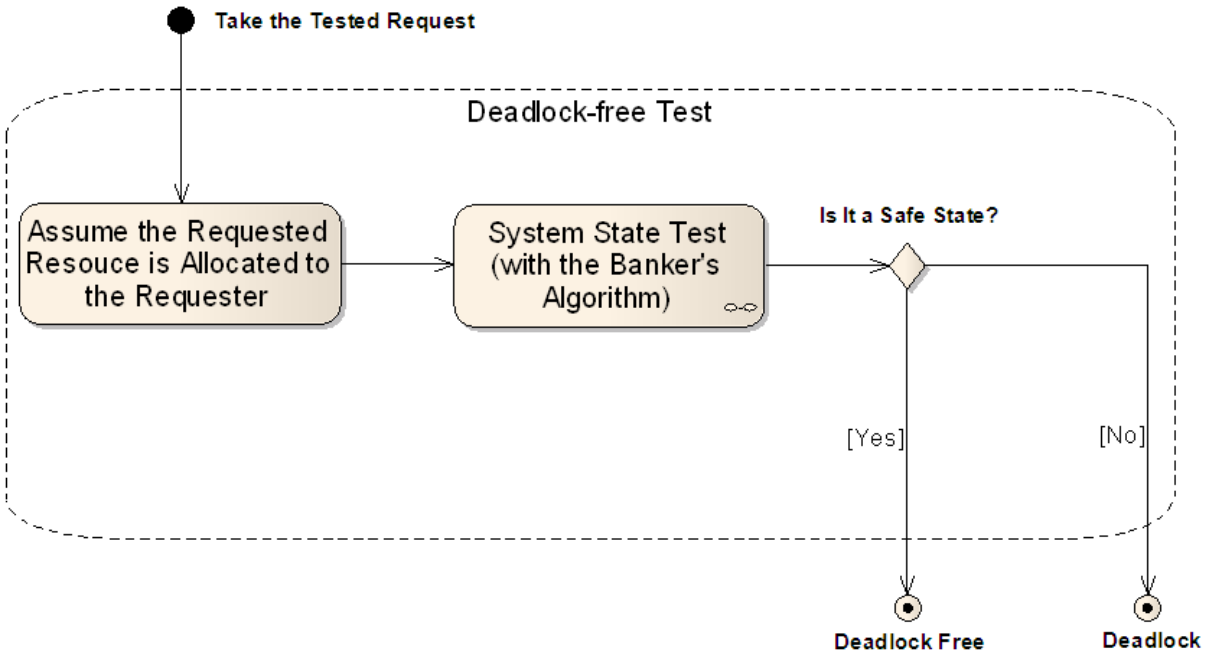



**Fig. 4-5 System State Test for Deadlock Avoidance**

In this section, the principle of the Banker's algorithm is introduced at first; then the description of the algorithm for railway synchronous simulation is given; at last some examples will demonstrate the usage of the algorithm.

### 4.3.1 The Principle of the Banker's Algorithm

The algorithm is named as "Banker's algorithm", because it can be used for a banking system to avoid the risk that the available money is no longer able to meet the demands for all its customers.

Considering a loaning system, there are many debtors requesting money from bankers. A loan program for each debtor is created to state the total amount of the money the debtor expects to borrow for completing a whole transaction. In the Banker's algorithm, a loan program is called a *process*.

It is impossible to satisfy all the requests from all the debtors at one time, and a debtor usually does not have to obtain the total amount of the loan to continue the whole transaction. For this reason, the huge amount of the loan for a debtor is divided into many small pieces of units, and one or more units are lent to a debtor in many steps.

The debtor returns money to the bankers also in many times, so that the debtor does not have to pay high interests and the bankers can hold enough money to keep the banking system running smoothly. In the Banker's algorithm, a unit of money is called a *resource*.

In each step of requesting loan, if the bankers simply approve every loan request without making any risk evaluation, such a situation might be happen: after a unit of money was lent out, the bankers cannot allocate enough money for the next minimal request, and all of the debtor are not able to return the money to the bankers yet unless they can get further investment (to generate benefit) from the bankers. All the transactions are blocked and the whole operation cannot be carried out any more. This situation fits exactly the definition of deadlock!

To avoid deadlocks, the system state should be tested carefully before approving a loan request in each step. This could be done by accounting and checking the loan programs. Assume a requested loan is granted to the requester, the available money of the bankers and maximum required money in future for each debtor will be calculated. All the loan programs are tested iteratively. If there is a loan program that can proceed with the currently available money, the loan program is considered as a feasible loan. It means that the maximum required money of that loan program is less than the currently available money. The system will calculate all the money supposed to be borrowed by that feasible loan program and return that money back to bankers (the available capital is added with the returned money). This process will be repeated until all the loan programs are proven to be feasible. If this process is executed successfully, the loan request can be approved; otherwise, the approval of the request should be postponed.

The precondition for applying this method is that the loan programs and maximum required money for each debtor are known in advance. Otherwise it is impossible to check the system state. This is also the precondition required by the approach of deadlock avoidance. Originally the Banker's algorithm was designed for computer operating system to avoid deadlocks among processes and computer resources (hardware or software). However, in many cases it is impossible to estimate the possible requested resources for a process since an execution of a process is very dynamic and unpredictable.

In railway synchronous simulation, the structure of the Banker's algorithm fits the simulation model quite well. The money of a banking system (resource) is mapped to

the infrastructure of railway network. A debtor is mapped to a simulation performer that sends requests of infrastructure resources to the railway operating control system. A loan program (process) is similar to a simulation movement task. In addition, the prediction of the maximal requested resources is easier to be handled in a railway simulation system than in a computer operating system. A movement task always specifies its target, from which the maximum requested infrastructure resources can be predicted via its route (especially in a dispatching system, the default route and the request resources can be obtained based on the predefined train path in a timetable). The detailed description of the Banker's algorithm in railway synchronous simulation is given in Section 4.3.2.

### 4.3.2 Banker's Algorithm in Railway Synchronous Simulation

As shown in Fig. 4-5, the Banker's algorithm is designed for deadlock avoidance algorithm. A request can be granted if the system will still be in a safe state in case of the requested resources are allocated. For this reason, the Banker's algorithm can be regarded as an algorithm to test system state. Two sets of data structure need to be managed for the Banker's algorithm: the *processes* and the *resources*.

The purpose of the Banker's algorithm is to try to allocate resources for each process in some orders. If a process (a movement task in "running" state) can get all the required resources, it can be regarded as a feasible task. A request can be approved if all the processes are proven as feasible tasks.

In a railway synchronous simulation, the processes are the movement tasks in "running" state (see Section 3.1.3). Since a movement task is always related to one simulation performer and a simulation performer can only execute one movement task in a certain point of time, the performer of a movement task is chosen to identify a process. Two performer sets (P) are used to represent two groups of processes (movement simulation tasks) in the Banker's algorithm:

- $P_1$: The performer with a feasible task will be included.
- $P_0$: The performer, of which the simulation task has not yet been proven as a feasible task, will be included.

For a microscopic simulation, the resources in the Banker's algorithm are the infrastructure resources as defined in Section 3.1.1. Each resource is initialized with exact one instance. Most of the examples in this chapter are running in this mode. For a macroscopic simulation, a resource may be defined based on a large section, e.g. a

station track section or a line section (see [RADTKE, 2008]). A resource in a macroscopic infrastructure model can be initialized with many instances according to its capacity. Supporting a resource type with multiple instances is one of the most important features of the Banker's algorithm. An example for using the Banker's algorithm in a macroscopic model is demonstrated in Section 4.3.3.2.

In the Banker's algorithm, three sets of resources are defined:

- Currently Available Resources (*AR*): The resources, those can be entered by a simulation performer.
- Currently Occupied Resources (*OR*): The resources, those are already blocked by the performer of the current concerned process (movement task).
- Maximal Required Resources (*MR*): The predicted maximal required resources for completing the currently concerned process (movement task).

The key to check the system state in the Banker's algorithm is to find a feasible task (similar to find a feasible loan program in a banking system). If such a feasible task does not exist, the system state is unsafe. Otherwise, when at least one feasible task is found, the feasible task will be able to be completed since it obtains all required resources. More important, the resources blocked by that feasible task will be returned to AR after the simulation task is completed. Therefore, the currently available resources are increased and the other tasks may get more chances to be completed. More feasible tasks are found, more resources are returned. A safe state will be proven when all the simulation tasks are feasible tasks. The detailed steps of the Banker's algorithm are:

Step 1: Initialize an empty performer set $P_1$.

Step 2: Initialize a performer set $P_0$, in which the performers with a movement task in "running" state is are included.

Step 3: Initialize a set AR, in which all of the currently available resources are included.

Step 4: Take a performer p from $P_0$, and initialize MR and OR for that performer.

Step 5: If there is a resource in MR that does not exist in AR or OR: go to Step 9 if there is no performer in $P_0$ can be moved to $P_1$, otherwise go back to step 4.

Step 6: If all resources in MR exist in AR or OR: all the resources of OR are released and returned to AR; if the destination of the movement task is still inside the observed network and the resource of the destination is in AR,

the resource of the destination should be removed from AR; at last the current performer p is added to $P_1$ and removed from $P_0$.

Step 7: If not all the performers in $P_0$ are moved to $P_1$, go back to Step 4; otherwise go to Step 8.

Step 8: All the performers are able to complete their processes. It means that the approval of the request will lead to a safe state.

Step 9: There is at least one performer that is not able to get required resources to complete its process. It means that the approval of the request will lead to an unsafe state.

The flow of the Banker's algorithm is shown in Fig. 4-6. The procedure will be ended at Step 8 (with the result of a safe state) or Step 9 (with the result of an unsafe state). According to the result of the system state, a request will be approved with the result of a safe state, since it is sufficient to avoid deadlocks. In case an unsafe state is derived, the situation is complicated. Although in some cases the request should be rejected, an unsafe state will not necessarily lead to deadlocks. Further discussion of such situations will be given in Section 4.4.

A special process is required in the implementation of the Banker's algorithm. For a request that has not been proven to lead to a safe state, the requester itself might be irrelevant with the unsafe state. It means that the incompetence of accomplishing all the simulation tasks does not result from the tested performer and the request. In this case, rejection of the request will prevent the "innocent" performer from continuing its simulation task. The possibility of deadlocks will be increased as long as more and more such "innocent" performers are blocked. Therefore, it is necessary to identify the irrelevant requesters and approve the request of valid movements.

After a deadlock with an unsafe result, a requester p can be sufficiently identified as an *irrelevant requester* if it satisfies all the following conditions:

- The requester p must be in the $P_1$ set, in other words, all the required resources of p in MR are in AR or OR.
- The target resource of p is *not* requested by other performers that cannot accomplish their simulation tasks (the performers are not in $P_1$).
- The target resource of p is requested by a performer is not in $P_1$, but it is occupied by another performer. The situation indicates that the incompetence of allocate the target resource is not due to the tested requester p, but the current occupier. Therefore, the requester p is still irrelevant to the result of

unsafe state. This situation is specially designed for shunting movements, when the tested requester is going to couple with the occupier on its destination.

Identification of irrelevant requesters can be regarded as an improvement of the Banker's algorithm. However, the identification process is so strongly depending on the current execution of the deadlock test (the current list of $P_1$ is always needed) that it is hard to be separated out as a standalone implementation of improvements. In addition, the practice (a software implementation is introduced in Section 4.5) shows that the system will be frequently blocked without "amnestying" the irrelevant requesters. Therefore, identifying irrelevant requesters is integrated within the implementation of the Banker's algorithm tightly as a standard function.

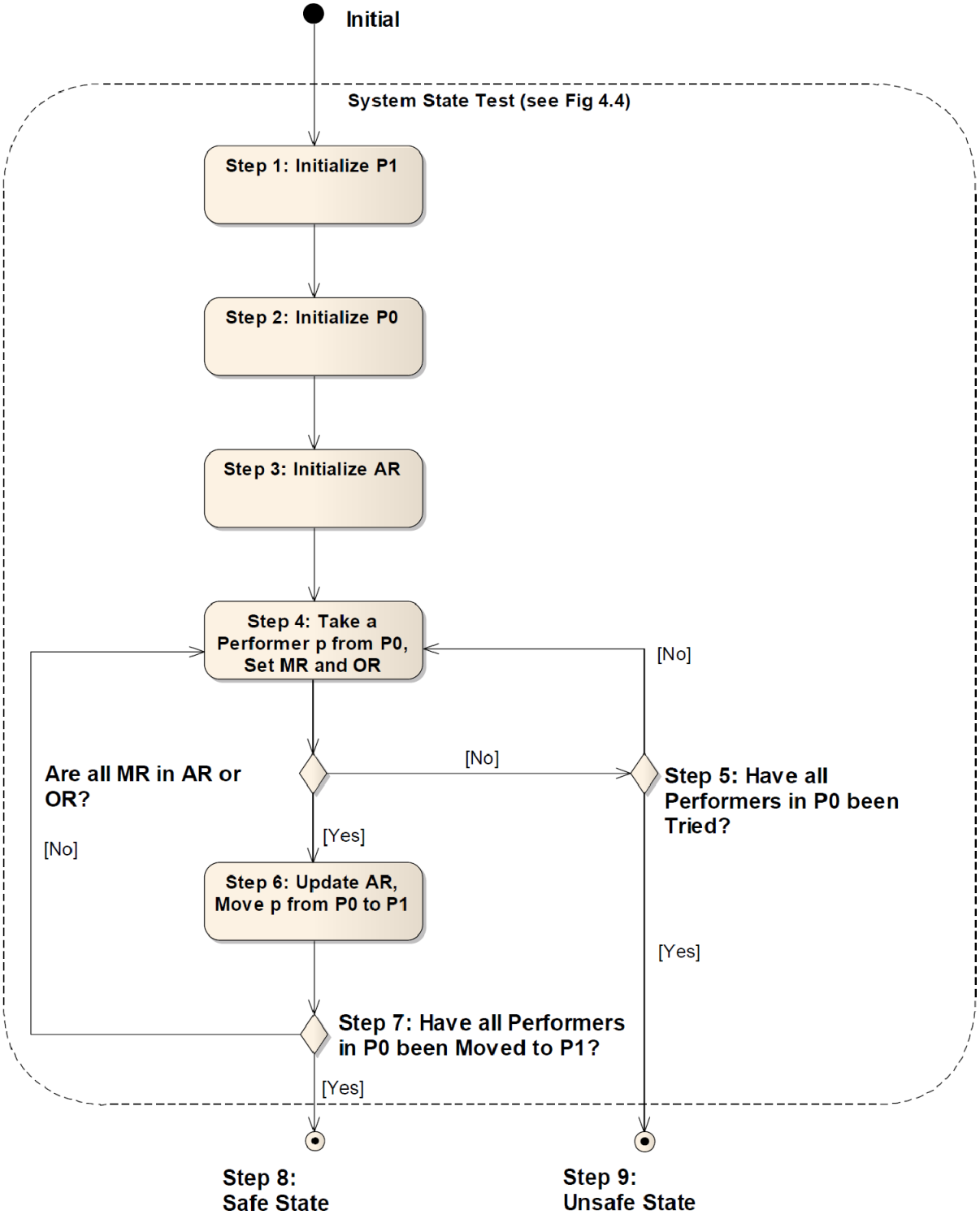



**Fig. 4-6 The Banker's Algorithm for System State Test**

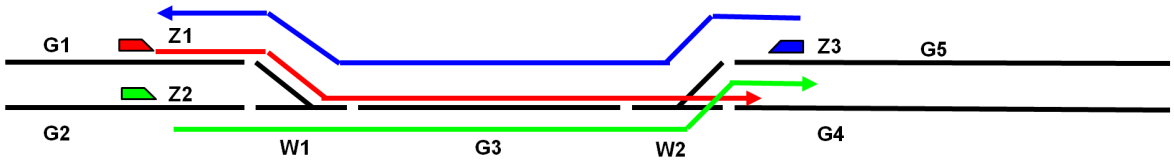
### 4.3.3 Examples of the Banker's Algorithm

Two examples are shown to demonstrate the procedure of the Banker's algorithm in different circumstances in this section. The description of the detailed procedure is based on the diagram shown in Fig. 4-6. Each step is also named with a step number

as defined in Section 4.3.2. There are several iterations from Step 4 to Step 7. As a convention, the number after a step number represents the iteration number.

#### 4.3.3.1 Example 1: A Simple Network

The example 1 is taken from the simple network shown in Fig. 4-1. The routes of the performers and the requested resources are listed in Fig. 4-7.




| Performer | Route | Requested Resource |
|---|---|---|
| Z1 | G1, W1, G3, W2, G4, out | W1 |
| Z2 | G2, W1, G3, W2, G5, out | W1 |
| Z3 | G5, W2, G3, W1, G1, out | W2 |

**Fig. 4-7 The Routes and the Requested Resources in Example 1**

Using the Banker's algorithm, the system state test for the request of W1 from Z1 is shown in Table 4-1. The result shows that the request will lead a safe state.

| Steps | Analysis for the Request of W1 from Z1 |
|---|---|
| Step 1 | $P_1$ {} |
| Step 2 | $P_0$ {Z1, Z2, Z3} |
| Step 3 | AR {G3, W2, G4} |
| Step 4-1 | Take Z1 from $P_0$<br>MR {G1, W1, G3, W2, G4}; OR {G1, W1}<br>All resources in MR are in AR or OR, go to Step 6 |
| Step 6-1 | AR {W1, G3, W2, G4, G1}<br>$P_1$ {Z1}; $P_0$ {Z2, Z3} |
| Step 7-1 | $P_0$ is not empty, go to Step 4. |
| Step 4-2 | Take Z2 from $P_0$<br>MR {G2, W1, G3, W2, G5}; OR {G2}<br>G5 is not in AR or OR, go to Step 5 |
| Step 5-2 | Z3 has not been tested, go back to Step 4 |
| Step 4-3 | Take Z3 from $P_0$<br>MR {G5, W2, G3, W1, G1}; OR {G5} |

| | |
|---|---|
| | All resources in MR are in AR or OR, go to Step 6 |
| Step 6-3 | AR {W1, G3, W2, G4, G1, G5}<br>$P_1$ {Z1, Z3}; $P_0$ {Z2} |
| Step 7-3 | $P_0$ is not empty, go to Step 4. |
| Step 4-4 | Take Z2 from $P_0$<br>MR {G2, W1, G3, W2, G5}; OR {G2}<br>All resources in MR are in AR or OR, go to Step 6 |
| Step 6-4 | AR { W1, G3, W2, G4, G1, G5, G2}<br>$P_1$ {Z1, Z3, Z2}; $P_0$ {} |
| Step 7-4 | $P_0$ is Empty, go to Step 8 |
| Step 8 | **Safe State** |

**Table 4-1 The Request of W1 from Z1 in Example 1**

The state tests for the request of W1 from Z2 and the request of W2 from Z3 are illustrated in Table 4-2 and Table 4-3 respectively. The results show that these requests may lead unsafe state and cannot be approved.

| Steps | Analysis for the Request of W1 from Z2 |
|---|---|
| Step 1 | $P_1$ {} |
| Step 2 | $P_0$ {Z1, Z2, Z3} |
| Step 3 | AR {G3, W2, G4} |
| Step 4-1 | Take Z1 from $P_0$<br>MR {G1, W1, G3, W2, G4}; OR {G1}<br>W1 is not in AR or OR, go to Step 5 |
| Step 5-1 | Z2 and Z3 have not been tested, go back to Step 4 |
| Step 4-2 | Take Z2 from $P_0$<br>MR {G2, W1, G3, W2, G5}; OR {G2, W1}<br>G5 is not in AR or OR, go to Step 5 |
| Step 5-2 | Z3 has not been tested, go back to Step 4 |
| Step 4-3 | Take Z3 from $P_0$<br>MR {G5, W2, G3, W1, G1}; OR {G5}<br>G1 is not in AR or OR, go to Step 5 |
| Step 5-3 | All performers are tested, go to Step 9 |
| Step 9 | **Unsafe State** |

**Table 4-2 The Request of W1 from Z2 in Example 1**

| Steps | Analysis for the Request of W2 from Z3 |
|---|---|
| Step 1 | $P_1$ {} |
| Step 2 | $P_0$ {Z1, Z2, Z3} |
| Step 3 | AR {G3, W1, G4} |
| Step 4-1 | Take Z1 from $P_0$<br>MR {G1, W1, G3, W2, G4}; OR {G1}<br>W2 is not in AR or OR, go to Step 5 |
| Step 5-1 | Z2 and Z3 have not been tested, go back to Step 4 |
| Step 4-2 | Take Z2 from $P_0$<br>MR {G2, W1, G3, W2, G5}; OR {G2}<br>G5 and W2 are not in AR or OR, go to Step 5 |
| Step 5-2 | Z3 has not been tested, go back to Step 4 |
| Step 4-3 | Take Z3 from $P_0$<br>MR {G5, W2, G3, W1, G1}; OR {G5, W2}<br>G1 is not in AR or OR, go to Step 5 |
| Step 5-3 | All performers are tested, go to Step 9 |
| Step 9 | **Unsafe State** |

**Table 4-3 The Request of W2 from Z3 in Example 1**

It should be noted that AR and OR of each performer may vary in Step 3 for different request tests. This due to the fact that a test is to check the system safe state for a request based on the assumption that the requested resource is allocated to the requester already. Although the requested resource has not been blocked yet, it will be put in OR of the requester instead of in AR.

#### 4.3.3.2 Example 2: Using the Banker's Algorithm for Macroscopic Models

The Banker's algorithm can also be used in a macroscopic model, in which a simulation is carried out without concerning the most detailed infrastructure information. A station section may be abstracted as a node in the network, and a track section links two nodes with each other. An example of such an abstraction is shown in Fig. 4-8. All the operations in the network are bidirectional, with 3 abstracted nodes (S1, S2 and S3) and 3 links (L1, L2 and L3). The trains Z1 and Z3 (marked with green colour) are running from left to right and the train Z2, Z4 and Z5 are running from right to left (marked with blue color).

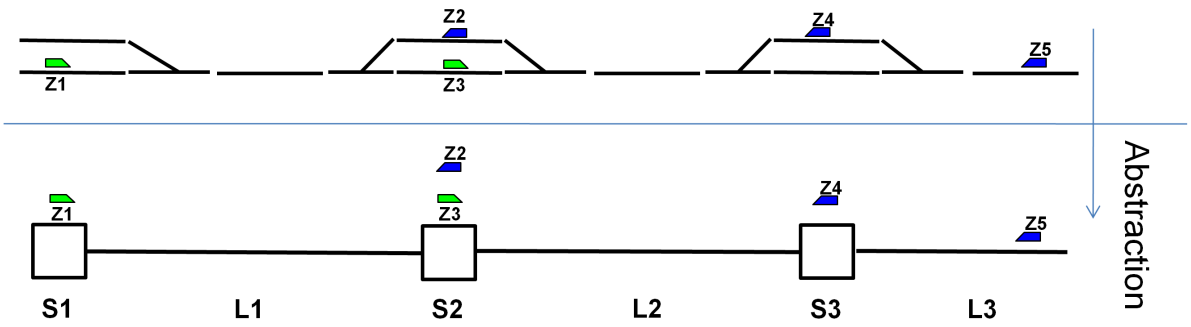



**Fig. 4-8 Abstraction of a Macroscopic network**

In a macroscopic simulation, a node or a link can be regarded as a resource. An important attribute of a node or link is the capacity. It defines the maximum number of performers that can be held in a resource simultaneously. For example, in the station S1, if there are two trains are allowed to stop or pass in the same time, the capacity of the station S1 is 2. The capacities of the resources are list in Table 4-4.

| Resource | S1 | S3 | S3 | L1 | L2 | L3 |
|---|---|---|---|---|---|---|
| Capacity | 2 | 2 | 2 | 1 | 1 | 1 |

**Table 4-4 The Capacities of the Resources in Example 3**

The Banker's algorithm supports a resource type with multiple instances. For executing an analysis of the system safe state for a request, the number of available instances is introduced as an attribute for AR. For instance, the available resource "S1(2)" means that the resource S1 is still available for two trains to stop or pass.

In Table 4-5 the procedure of state test for the request of L1 from train Z1 is demonstrated (with the result of an unsafe state):

| Steps | Analysis for the Request of L1 from Z1 |
|---|---|
| Step 1 | $P_1$ {} |
| Step 2 | $P_0$ {Z1, Z2, Z3, Z4, Z5} |
| Step 3 | AR {S1(1), L2(1), S3(1)} |
| Step 4-1 | Take Z1 from $P_0$<br>MR {S1, L1, S2, L2, S3, L3}; OR {S1, L1}<br>S2, L3 are not in AR or OR, go to Step 5 |
| Step 5-1 | Z2, Z3, Z4 and Z5 have not been tested, go back to Step 4 |
| Step 4-2 | Take Z2 from $P_0$<br>MR {S2, L1, S1}; OR {S2}<br>L1 is not in AR or OR, go to Step 5 |
| Step 5-2 | Z3, Z4 and Z5 have not been tested, go back to Step 4 |
| Step 4-3 | Take Z3 from $P_0$<br>MR {S2, L2, S3, L3}; OR {S2}<br>L3 are not in AR or OR, go to Step 5 |
| Step 5-3 | Z4 and Z5 have not been tested, go back to Step 4 |
| Step 4-4 | Take Z4 from $P_0$<br>MR {S3, L2, S2, L1, S1}; OR {S3}<br>S2 and L1 are not in AR or OR, go to Step 5 |
| Step 5-4 | Z5 has not been tested, go back to Step 4 |
| Step 4-5 | Take Z5 from $P_0$<br>MR {L3, S3, L2, S2, L1, S1}; OR {L3}<br>S2 and L1 are not in AR or OR, go to Step 5 |
| Step 5-5 | All performers are tested, go to Step 9 |
| Step 9 | **Unsafe State** |

**Table 4-5 The Request of L1 from Z1 in Example 3**

In Table 4-6 the procedure of state test for the request of L1 from train Z2 is demonstrated (with the result of a safe state):

| Steps | Analysis for the Request of L1 from Z2 |
|---|---|
| Step 1 | $P_1$ {} |
| Step 2 | $P_0$ {Z1, Z2, Z3, Z4, Z5} |
| Step 3 | AR {S1(1), L2(1), S3(1)} |
| Step 4-1 | Take Z1 from $P_0$<br>MR {S1, L1, S2, L2, S3, L3}; OR {S1}<br>L1, S2 and L3 are not in AR or OR, go to Step 5 |
| Step 5-1 | Z2, Z3, Z4 and Z5 have not been tested, go back to Step 4 |
| Step 4-2 | Take Z2 from $P_0$<br>MR {S2, L1, S1}; OR {S2, L1}<br>All resources in MR are in AR or OR, go to Step 6 |
| Step 6-2 | AR {S1(1), L2(1), S3(1), S2(1), L1(1)}<br>$P_1$ {Z2}; $P_0$ {Z3, Z4, Z5, Z1} |
| Step 7-2 | $P_0$ is not empty, go to Step 4 |
| Step 4-3 | Take Z3 from $P_0$<br>MR {S2, L2, S3, L3}; OR {S2}<br>L3 not in AR or OR, go to Step 5 |
| Step 5-3 | Z4, Z5 and Z1 have not been tested, go back to Step 4 |
| Step 4-4 | Take Z4 from $P_0$<br>MR {S3, L2, S2, L1, S1}; OR {S3}<br>All resources in MR are in AR or OR, go to Step 6 |
| Step 6-4 | AR {S1(1), L2(1), S3(2), S2(1), L1(1)}<br>$P_1$ {Z2, Z4}; $P_0$ {Z5, Z1, Z3} |
| Step 7-4 | $P_0$ is not empty, go to Step 4 |
| Step 4-5 | Take Z5 from $P_0$<br>MR {L3, S3, L2, S2, L1, S1}; OR {L3}<br>All resources in MR are in AR or OR, go to Step 6 |
| Step 6-5 | AR {S1(1), L2(1), S3(2), S2(1), L1(1), L3(1)}<br>$P_1$ {Z2, Z4, Z5}; $P_0$ {Z1, Z3} |
| Step 7-5 | $P_0$ is not empty, go to Step 4 |
| Step 4-6 | Take Z1 from $P_0$<br>MR {S1, L1, S2, L2, S3, L3 }; OR {S1}<br>All resources in MR are in AR or OR, go to Step 6 |

| | |
|---|---|
| Step 6-6 | AR {S1(2), L2(1), S3(2), S2(1), L1(1), L3(1)}<br>$P_1$ {Z2, Z4, Z5, Z1}; $P_0$ {Z3} |
| Step 7-6 | $P_0$ is not empty, go to Step 4 |
| Step 4-6 | Take Z3 from $P_0$<br>MR {S2, L2, S3, L3 }; OR {S2}<br>All resources in MR are in AR or OR, go to Step 6 |
| Step 6-6 | AR {S1(2), L2(1), S3(2), S2(2), L1(1), L3(1)}<br>$P_1$ {Z2, Z4, Z5, Z1, Z3}; $P_0$ {} |
| Step 7-6 | $P_0$ is empty, go to Step 8 |
| Step 8 | **Safe State** |

**Table 4-6 The Request of L1 from Z2 in Example 2**

Example 2 only demonstrates the basic principle for using the Banker's algorithm for a macroscopic network. Since in most cases a macroscopic simulation is not suitable for a dispatching system to operate single train movement, further discussions are not given in this dissertation.

## 4.4 Improvements to the Banker's Algorithm and System Performance

Based on the principle of deadlock avoidance, the Banker's algorithm can guarantee a system against deadlocks when the system is in a safe state. As explained in [SILBERSCHATZ et al., 2005]: a safe state will never lead to a deadlock situation, and a deadlock situation is always related to an unsafe state. However, not all unsafe state will lead to deadlocks. As long as an unsafe state is identified, the request of the resource will be rejected, and the performer should wait for next chances. If an unsafe state not sufficiently leads to deadlocks (some examples will be demonstrated in this section), such a stop will decrease the utilization level of infrastructure resources. As shown in Fig. 4-9, the main purpose of the improvements for the Banker's algorithm is to increase the space of safe state and reduce the possibility of false positives. Therefore, system efficiency and infrastructure utilization level have to be improved with specific measures designed for railway operation.

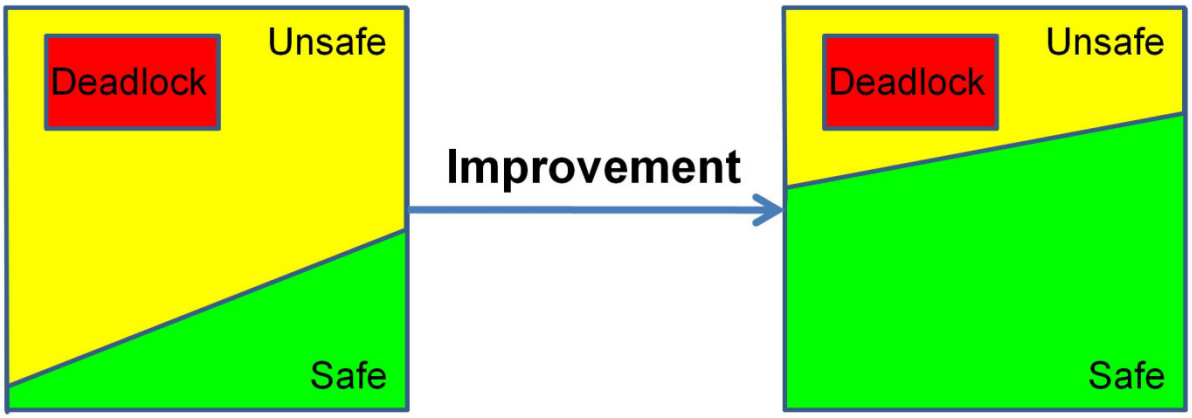



**Fig. 4-9 Improvement for Deadlock Avoidance**

Making a system state test to avoid deadlocks is a time-consuming work. Particularly for a dispatching system, system performance is a decisive aspect for successfully putting the system into practice. Specific measures concerning on improving the system performance are also discussed in this section.

Hence, according to the purposes, further suggested improvements can be classified into two categories: *improvements of the Banker's algorithm* (to increase the space of safe state) and *improvements of the system performance*. In this section, each suggested improvement will be described in this way: the classification of the improvement will be stated at the beginning; then the intention of the improvement will be demonstrated with an example; finally the solution and the possible implementation approach are given.

It is very complicate to implement all the suggested improvements in one application. Measures to increase the safe state space also bring side effects to system performance, since additional processes, normally together with a new round of system state test, are introduced additionally. For these reasons, not all the suggested measures are required to be utilized in one implementation. An improvement will be implemented only if the related problem of that improvement occurs frequently.

If an improvement of the Banker's algorithm is introduced, the timing to apply the improvement should also be carefully designed. Although it is highly depending on the application context, the following principles are most commonly used:

- Longitudinal principle: for a certain request, if the waiting time (due to deadlocks) exceeds a predefined time threshold, an improvement of the Banker's algorithm can be applied.

- Transversal principle: in a single processing step, if all the deadlock-free tests have failed, at least an improvement of the Banker's algorithm should be applied.

Generally, the longitudinal principle is more practical than the transversal principle in a large network, and the transversal principle is easier to be implemented than the longitudinal principle. It is also possible that both of them are integrated together. For example, if the transversal principle is applied in a single processing step, the performer with the highest waiting time (due to deadlock) will be given the highest priority to apply the improvements.

### 4.4.1 Analysis with Potential State Transitions

**Classification**: Improvement of the Banker's algorithm

**Intention**: When future potential state transitions are not considered, unnecessary stops are introduced. In extreme cases, a simulation falls into failure. This improvement is to find more feasible arrangement of resource allocation with consider to further potential state transitions.

A typical problem of the Banker's algorithm is demonstrated in Fig. 4-10:


| Performer | Route | Requested Resource |
|---|---|---|
| Z1 | G1, W1, G2, W2, G4, out | W1 |
| Z2 | G4, W2, G3, W1, G1, out | W2 |

**Fig. 4-10 A Deadlock Situation without Considering Potential State Transitions**

The procedure of the system state test for the request of W1 from Z1 is illustrated in Table 4-7. The request will be rejected since it will lead to an unsafe state.

| Steps | Analysis for the Request of W1 from Z1 |
|---|---|
| Step 1 | $P_1$ {} |
| Step 2 | $P_0$ {Z1, Z2} |
| Step 3 | AR {G2, G3, W2} |
| Step 4-1 | Take Z1 from $P_0$<br>MR {G1, W1, G2, W2, G4}; OR {G1, W1}<br>G4 is not in AR or OR, go to Step 5 |
| Step 5-1 | Z2 has not been tested, go back to Step 4 |
| Step 4-2 | Take Z2 from $P_0$<br>MR {G4, W2, G3, W1, G1}; OR {G4}<br>W1 and G1 are not in AR or OR, go to Step 5 |
| Step 5-2 | All performers are tested, go to Step 9 |
| Step 9 | **Unsafe State** |

**Table 4-7 The Request of W1 from Z1 in the Example in Fig. 4-10**

Similarly, the system state test for the request of W2 from Z2 will also result in an unsafe state. Therefore, both Z1 and Z2 are not allowed to continue their movements, and the system is trapped in failure at this point.

However, the system will not necessarily lead to deadlocks after all the requests are approved as shown in Fig. 4-11. Both Z1 and Z2 can complete their movement tasks without any deadlock problems.

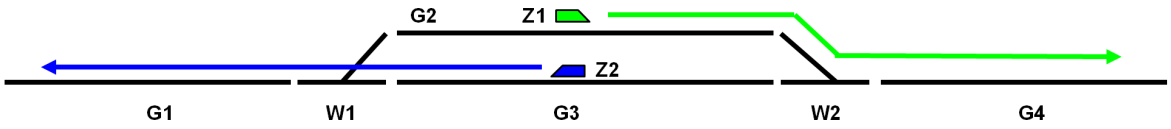



**Fig. 4-11 A Situation without Deadlocks after all the Requests are Approved**

A limitation of the Banker's algorithm can be revealed: the system state test demonstrated in Table 4-7 concentrates only on the current situation, and future potential state transitions – from an unsafe state to a safe state – are ignored. Unnecessary stops for "avoidable" deadlocks are introduced, and in some extreme cases as shown in Fig. 4-10, all performers are blocked with each other and the system is halted finally.

Hence, testing for potential state transitions will improve the performance of the Banker's algorithm. A key point of the improvement is to identify a *feasible track* (FT)

as the new location of the tested requester in future transited state. A feasible track for the tested requester is defined as:

1) The FT is in the route of the tested requester, and all the resources in the route from the current position of the tested requester to the FT are free to enter.
2) The FT is long enough to hold the tested requester.
3) Between the current position of the tested requester and the next junction type resource (JR) behind the FT (JR is searched in the route of the requester), at least one alternative route exists when the FT is occupied by the tested requester.

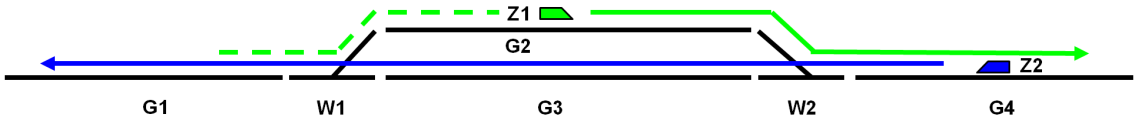
The condition 3) defines a feasible track as a “station track” (a loop, see [PACHL, 2007] and Section 4.2.2). After moving from the current position to the feasible track, the tested requester will give a way for other performers to overtake or pass. The improvement with the consideration of future potential state transitions can be applied as follows:

**Improvement A: When a request has been tested with an unsafe state, if a feasible track exists in MR, further state tests can be executed based on the assumption that the requester occupies the feasible track. The new round of tests are not only executed for the current deadlock test, but also for those tests failed in last round of tests, since the assumed state transition will create the chances for other performers to accomplish their trips.** To examine if deadlocks are possibly to be avoided with potential state transitions, a new round of tests is started. The new tests are based on the assumption that the performer has moved to a new position, where the system state is transited. A feasible track that is long enough to hold the whole performer and give a way to other performers is a suitable new position, meanwhile all the resources in OR can be released to produce the possible state transition. An essential implication of the new round tests is to test the system safe state when the tested requester blocks the feasible track and releases all the resources from OR to the feasible track. In additional, it is necessary to ensure that the partial route from OR to the feasible track is able to enter (conflict-free). If a test results in a safe state, the resources in the partial route should be reserved as a whole for the tested requester.

Additional alternative route searching should be executed for the performers that are supposed to enter the feasible track. Since the feasible track will be blocked by the

tested requester, other performers are not able to enter the feasible track. An alternative route that excludes the feasible track is required. The discussion of alternative route searching will be given in Section 4.4.3.

For the example in Fig. 4-10, after an unsafe state is concluded with the test for the request of W1 from Z1, the assumed new situation will be changed as follows:




| Performer | Route | Requested Resource |
|---|---|---|
| Z1 | G2, W2, G4, out | W2 |
| **Z2** | **G4, W2, G3, W1, G1, out** | **W2** |

**Fig. 4-12 The New Situation for Second Round State Test**

The requested resource for train Z1 has been changed to W2, based on the assumption that Z1 passes G1, W1 and occupies G2. The new test is to analyze the situation if G2 is blocked by Z1. In this situation, G1 and W1 will be regarded as available resource, and G2 will be put into the OR of Z1. The whole procedure is illustrated in Table 4-8.

| Steps | Analysis for the Request of W2 from Z2 |
|---|---|
| Step 1 | $P_1$ {} |
| Step 2 | $P_0$ {Z1, Z2} |
| Step 3 | AR {G1, W1, G3 } |
| Step 4-1 | Take Z1 from $P_0$<br>MR {G2, W2, G4}; OR {G2}<br>W2 and G4 are not in AR or OR, go to Step 5 |
| Step 5-1 | Z2 has not been tested, go back to Step 4 |
| Step 4-2 | Take Z2 from $P_0$<br>MR {G4, W2, G3, W1, G1}; OR {G4, W2}<br>All resources in MR are in AR or OR, go to Step 6 |
| Step 6-2 | AR { G1, W1, G3, W2, G4}<br>$P_1$ {Z2}; $P_0$ {Z1} |
| Step 7-2 | $P_0$ is not empty, go to Step 4. |
| Step 4-3 | Take Z1 from $P_0$<br>MR {G2, W2, G4}; OR {G2} |

| | |
|---|---|
| | All resources in MR are in AR or OR, go to Step 6 |
| Step 6-3 | AR { G1, W1, G3, W2, G4, G2}<br>$P_1$ {Z2, Z1}; $P_0$ {} |
| Step 7-3 | $P_0$ is Empty, go to Step 8 |
| Step 8 | **Safe State** |

**Table 4-8 Further Analysis with Improvement C**

With the result of a safe state, the analysis shows that the further allocation of W2 to Z2 with the potential state transition is feasible. Although the test is only executed for the request of W2 by Z2, the whole route W1 and G2 should be reserved for Z1. An analysis can also be carried out for Z1 requesting for W2. The result will show that Z1 should not be allowed to enter W2, and Z1 should wait at G2 until Z2 passes. By testing with the potential state transition, the deadlock problem does not exist anymore as shown in Fig. 4-11.

The same example shown in Fig. 4-3 can also be solved with the Banker's algorithm, if the analysis with potential state transition is applied. For train Z1, the next feasible track is G2. A deadlock-free test can be carried out based on the assumption that G2 is occupied by Z1 (see Fig. 4-13). In this situation, it can be proven that the request of W4 of Z3 and the request of W2 of Z4 will lead to safe states.

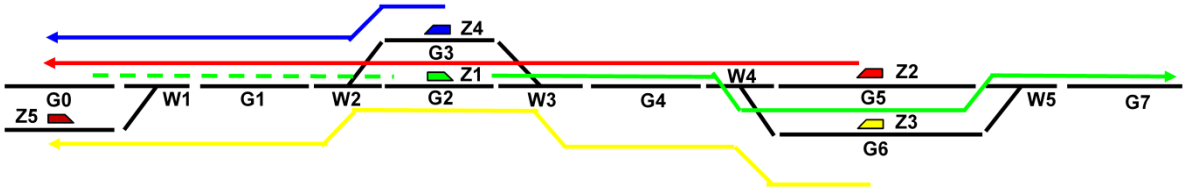



**Fig. 4-13 Analysis with Potential State Transition for Avoiding False Positives**

The new round tests are started for the feasible track at the nearest position to OR. When no test is passed, the resources at further positions can be tested. However, it is not recommended to extend new round tests too far; otherwise too many resources are involved too earlier to keep the route (from the current position to the feasible track) being free to enter, that will influence the requests of the involved resources from other performers. It should also be noticed that the analysis with potential state transition requires additional processing time. In many cases, it is more efficient to give up the failed, current round of tests, and to examine potential state transitions for other performers.

### 4.4.2 Test Processes in a Right Order

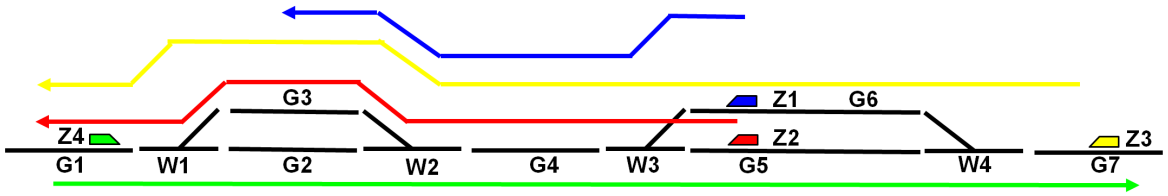
**Classification**: Improvement of the Banker's algorithm

**Intention:** A state test unnecessarily leads to an unsafe statec when an inappropriate order of processes is used in the test. This improvement is to increase the space of safe state when executing the state test for processes in a right order.

Till now all the trains in the examples of Chapter 4 will leave the network finally. It is not always the case. In some situations the destination of a movement task is still inside the observed infrastructure network, and such a destination is called an internal destination. The possible examples can be:

- A train is operated locally, and the whole train path is inside the observed network.
- A train comes from the outside of the network, and the destination of the train is a depot that is inside the observed network.
- A shunting set is operated in the observed shunting yard, and the shunting set is supposed to reside in the shunting yard until the whole simulation is finished.

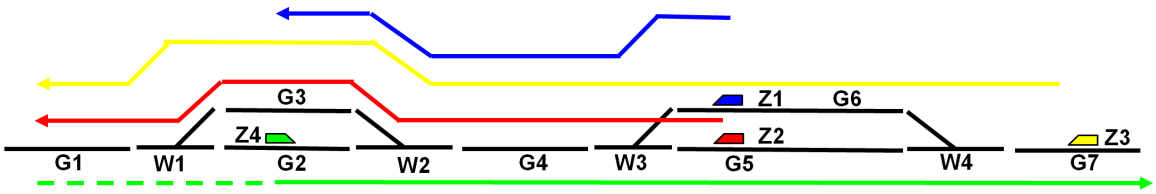
Special attention should be given in system state tests when performers with an internal destination exist. The order of testing performers in $P_0$ will influence the result of a state test. Consider the following example in Fig. 4-14:




| Performer | Route | Requested Resource |
|---|---|---|
| Z1 | G6, W3, G4, W2, G3 | W3 |
| Z2 | G5, W3, G4, W2, G3, W1, G1, out | W3 |
| Z3 | G7, W4, G5, W3, G4, W2, G3, W1, G1, out | W4 |
| Z4 | G1, W1, G2, W2, G4, W3, G5, W4, G7, out | W1 |

**Fig. 4-14 An Example of a Performer with an Internal Destination**

For the request of W1 from Z4, it can be proven that it will lead to an unsafe state (the procedure is ignored here). When applying the improvement A, the request of G2 from Z4 will be considered as shown in Fig. 4-15.




| Performer | Route | Requested Resource |
|---|---|---|
| Z1 | G6, W3, G4, W2, G3 | W3 |
| Z2 | G5, W3, G4, W2, G3, W1, G1, out | W3 |
| Z3 | G7, W4, G5, W3, G4, W2, G3, W1, G1, out | W4 |
| **Z4** | **G2, W2, G4, W3, G5, W4, G7, out** | **G2** |

**Fig. 4-15 The New Situation when Applying Improvement B for Z4**

In the new round of system state test for the request of G2 form Z4, the original $P_0$ includes all the performers (Z1, Z2, Z3 and Z4). A performer will be taken from $P_0$ to test the possibility of resource allocation.

Assume that Z1 is chosen as the first tested performer. All the required resources in MR ({G6, W3, G4, W2, G3}) are in AR ({W3, G4, W2, G3, W4}) or OR ({G6}). Z1 will be moved from $P_0$ to $P_1$ and its current blocked resources will be returned to AR. Since the route of Z1 is supposed to be ended at an internal destination (G3), the G3 will be taken out from AR (see Section 4.3.2 Step 6). Continuing the deadlock-free test, all the performers in $P_0$ (Z2, Z3 and Z4) are not able to complete their simulation tasks. The state test leads to an unsafe state and the request of G2 from Z4 will be rejected.

However, if assumed that Z2 instead of Z1 is chosen as the first tested performer, a safe state will be concluded from the test. The sequences of the train movements can be arranged as Z2, Z3, Z4 and Z1 (see Fig. 4-16).

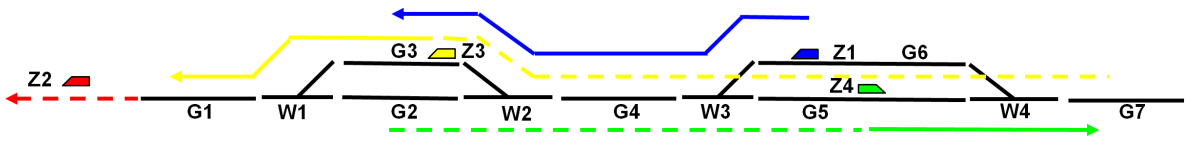



**Fig. 4-16 A Possible Movement Arrangement after Z4 Blocked G2**

Starting to test a simulation task with an internal destination will decrease the possibility to find a right arrangement for all performers. If such a simulation task can be completed in the beginning, the destination resource will be removed from AR and reserved for the performer of that simulation task permanently. This reserved resource may also be required by other performers, and these performers will never be able to complete their simulation tasks eventually. Therefore, testing simulation tasks in an inappropriate order may result in an unsafe state unnecessarily and reduce the space of safe state. An improvement concerning on the order of testing performers is introduced here:

**Improvement B: Testing the performers in $P_0$ for resource allocation should be arranged in the following order:**

1) **At first all the performers supposed to leave the observed network should be tested until no such a performer can be moved from $P_0$ to $P_1$.**
2) **Then the performers with an internal destination lying at a dead-end track will be tested. If no such a performer can be moved from $P_0$ to $P_1$, go to 3); otherwise go back to 1).**
3) **The rest of the performers will be tested at last.**

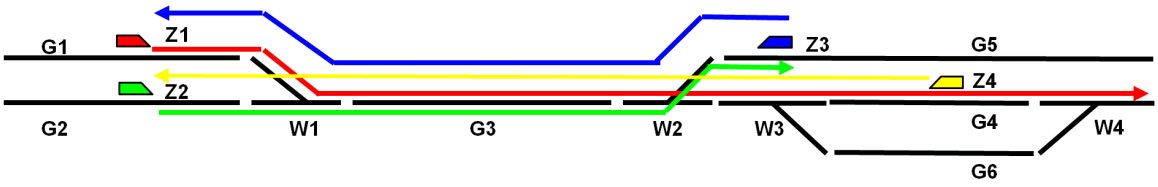
If a performer with an internal destination lying at a dead-end track, it will influence other performance very rare. For this reason, these performers are treated as a special group with a higher priority compared with other performers supposed to be ended in the observed network. This is particularly useful in some situations when several trains are operated inside the observed network.

### 4.4.3 Alternative Route

**Classification**: Improvement of the Banker's algorithm

**Intention:** A state test will be failed with the original scheduled route. However, a feasible solution exists when an alternative route is used. This improvement is to increase the space of safe state with alternative routes.

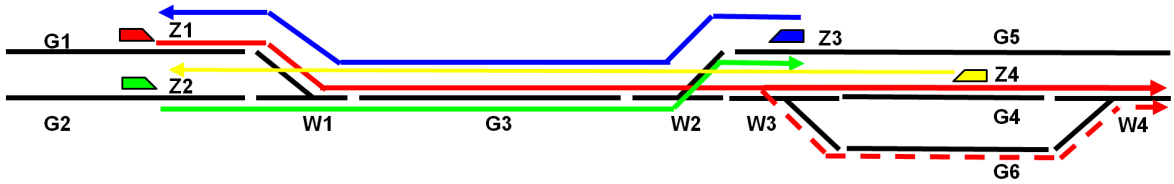
The most important precondition of the Banker's algorithm is to know the required resources of each process in advance. In railway synchronous simulation, it means the route information of each train should be known before a system state test starts. Using fixed routes for all performers would be the easiest implementation. However, this method is not flexible enough for deadlock avoidance.




| Perform | Route | Requested Resource |
|---|---|---|
| Z1 | G1, W1, G3, W2, W3, G4, W4, out | W1 |
| Z2 | G2, W1, G3, W2, G5, out | W1 |
| Z3 | G5, W2, G3, W1, G1, out | W2 |
| Z4 | G4, W3, W2, G3, W1, G2, out | W3 |

**Fig. 4-17 An Unsafe State with Fixed Routes**

An example is shown in Fig. 4-17: with predefined fixed routes, an unsafe state will be concluded in the system state test (the steps of the system state test are similar to the example shown in Section 4.3.3.1 and ignored here). Although all train movements are not allowed due to deadlocks, the deadlock situation can be avoided if Z1 change its route to use G6 instead of G4 (as shown in Fig. 4-18, the alternative route is marked with dashed line and bold font).




| Perform | Route | Requested Resource |
|---|---|---|
| Z1 | **G1, W1, G3, W2, W3, G6, W4, out** | W1 |
| Z2 | G2, W1, G3, W2, G5, out | W1 |
| Z3 | G5, W2, G3, W1, G1, out | W2 |
| Z4 | G4, W3, W2, G3, W1, G2, out | W3 |

**Fig. 4-18 A Safe State with Alternative Routes**

Therefore, an important improvement is introduced:

**Improvement C: The alternative routes will be taken into consideration if a train with alternative routes is in a deadlock situation (in an unsafe state).**

Train routes are determined by certain route searching algorithms. In a train schedule, the station-relevant destinations are the *scheduled destinations* that should be respected in a fixed sequence during route searching processes. Any modification of scheduled destinations should be carefully evaluated during the dispatching and optimization process. The rest of the destinations in the original train route are the *relative destinations*. The original relative destinations may be excluded when an alternative route is utilized from a scheduled destination to the next scheduled destination. In the procedure of searching alternative routes, the determination of scheduled destinations starts at first, and alternative routes including a series of relative destinations between every two adjacent, scheduled destinations are searched afterwards. The route searching procedure with the consideration of scheduled/relative destinations is given in [MARTIN, 1995] and [SCHLAICH, 2002].

If all the possible routes of a performer were available before a simulation task is in "running" state, it would be easy to improve the Banker's algorithm with alternative routes. The route with the shortest running time can be selected as the alternative route. In many cases it is not practical to get all the possible routes for all performers. According to the predefined absolute destinations, a route with the shortest running time can be searched by some kind of route searching algorithm, and DIJKSTRA's Algorithm is most often used (see [DIJKSTRA, 1959]).

When an alternative route is required, the route with the secondary shortest running time can also be searched with DIJKSTRA's algorithm in such a way: each time an edge in the shortest route is removed from the whole network, and the optimum solution to this network is searched. Repeating the operation for all the edges, a ranked list of less-than-optimal solutions is generated. The route with the secondary shortest running time can be obtained from the list. This approach requires many times of route searching, and possibly deadlocks still occur when applying the route with the second shortest running time.

A more practical solution can be utilized based on the principle of the Banker's algorithm. The improvement with alternative routes is applied only if the request cannot pass the first round deadlock-free test with original route. Considering the steps of the Banker's algorithm described in Section 4.3.2, the test must be ended at the Step 9 when all the performers in $P_0$ are not able to get all required resources in MR. For these performers, there must be at least one resource that is in MR but not in AR or OR. This resource is removed out of the graph, and the optimum solution is searched

within the new collection of infrastructure resources. If such a solution exists, the route can be used as the alternative route. In this way, it is not necessary to execute route searching for many times, and the resource that leads problem in previous test is excluded from the route. The process of the improvement with alternative routes is illustrated in Fig. 4-19.

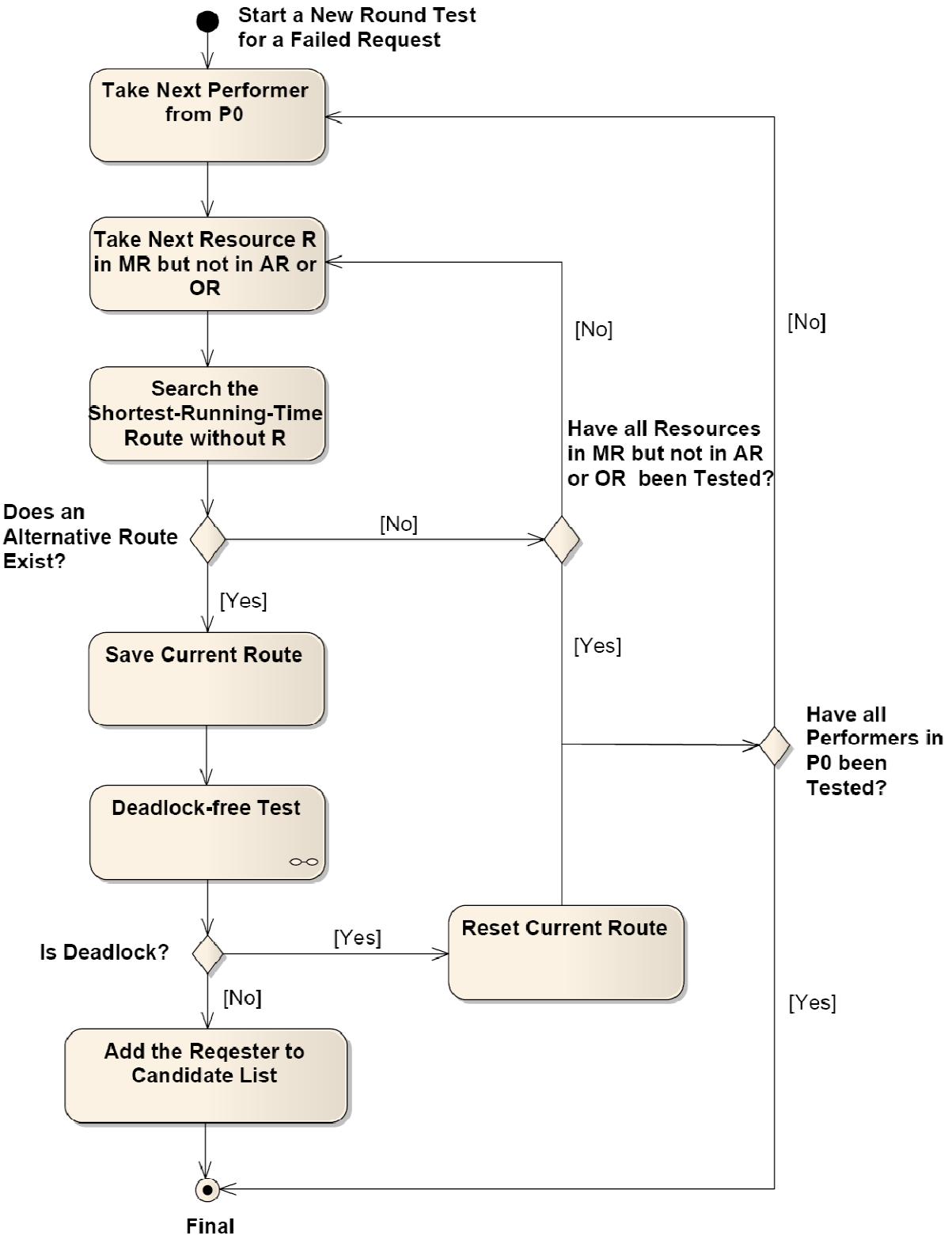



**Fig. 4-19 Improvement of the Banker's Algorithm with Alternative Routes**

### 4.4.4 Timing of State Tests

**Classification**: Improvement of the system performance

**Intention:** A state test for a request is unnecessarily executed when the result of test does not influence the system state. This improvement is to skip those unnecessary state tests.

As described in Section 3.2.2.2, deadlock-free tests are carried out when allocating new requested resources for performers. When the Banker's algorithm is applied, the test will go through all the processes to determine the system state. In some cases such a time consuming job can be skipped to gain a higher system performance.

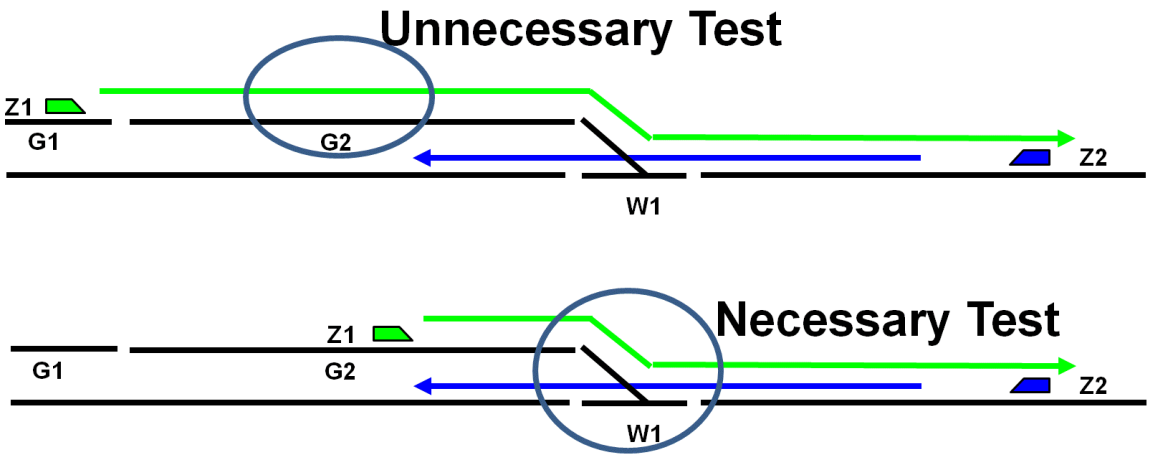



**Fig. 4-20 Examples of Unnecessary and Necessary Deadlock-free Tests**

The example in Fig. 4-20 shows the fact that a deadlock-free test is not always required. When Z1 requests to enter G2, a deadlock-free test can be ignored. The decision of the deadlock-free test has no influence to the system safe state, no matter if the request of G2 from Z1 is approved or rejected. However, when Z1 requests to enter W1, the deadlock-free test is necessary. Because blocking W1 may influence resource allocating for Z2, the system safe state is changed after W1 is allocated to Z1.

To improve the system performance, the unnecessary deadlock-free tests can be skipped. As discussed in Section 3.1.1, infrastructure resources can be defined based on block sections or infrastructure elements, and there are two types of infrastructure resources: junction type resources and non-junction type resources. The difference between G2 and W1 is that G2 is a non-junction type resource and W1 is a junction type resource. The following improvement is introduced:

**Improvement D: A deadlock-free test (system state test) will not be carried out for a request of entering a non-junction type resource.**

In a deadlock-free test, the required resources for a requester are established based on its route. From the point of view of the relation among routes, entering a non-junction type resource does not influence the resource allocation for other routes. The approval of the request of a non-junction type resource will not change system deadlock status.

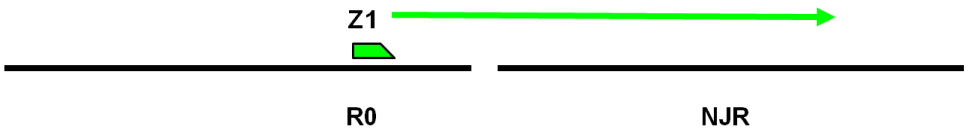



**Fig. 4-21 A Request of a Non-Junction Type Resource**

It can be proven with the network shown in Fig. 4-21. Z1 at resource R0 is requesting for the non-junction type resource (NJR). If the system is in a safe state before the request of NJR is granted, the approval of the request will still leading to a safe state. Assume that the approval of the request of NJR will result in an unsafe state (Z1 cannot completes its simulation task if NJR is allocated to Z1.), it can be concluded that Z1 cannot complete its simulation task at resource R0 either (since the backwards from NJR will necessarily lead to R0, and the resource blocking situation for other performers will not be influenced). The conclusion contradicts with the assumption that the system is in a safe state before the request is approved. In case of the system is in an unsafe status when Z1 is in R0, the approval of request of NJR will also lead to an unsafe status. Hence the system status will never be changed if a request of a non-junction resource is approved.

For a request of a junction type resource, the system state may be changed since the performer is requesting to enter an area that may influence other performers. A junction type resource is not only related with the route of the current requester, but also linked to other routes of other performers. The system state may be changed when other routes are influenced due to blocking a junction type resource.

The change of state can be a state transition from a safe state to an unsafe state, and the system state also can be changed from an unsafe state to a safe state. No matter which kind of state change happens, it is necessary to test the system state

before entering a junction type resource, and it is efficient to check deadlocks in front of a junction type resource only.

## 4.5 Software Implementation and Evaluation for Synchronous Simulation and Deadlock Avoidance

The railway synchronous simulation model (see Chapter 3) and the Banker's algorithm have been implemented and evaluated by the software tool PULRAN (Program to Research the Logistic in the Shunting Operation, in German: *Programm zur Untersuchung der Logistik im Rangierdienst*) developed at the IEV. Originally, PULRAN was designed to support shunting process planning and simulation (see PULRAN SPEZIFIKATION 2009). Different scenario for auxiliary station operations have been simulated and examined by the tool.

After inputting the information of infrastructure network, train group definitions, and operational concepts, which are mapped to the components of a synchronous simulations model (see Section 3.1) respectively, the simulation process is executed in a series of fixed time interval steps. When a simulation process is finished, the occupancy chart of each infrastructure resources will be derived. Since the operations take place in shunting yards and auxiliary stations, the infrastructure resources are defined based on infrastructure elements instead of blocking sections. Operational concepts are modeled as simulation tasks. For a train, several simulation tasks (movement tasks and non-movement tasks) are defined. The integrity and consistency of the simulation tasks are maintained according to certain working sequences.

In PULRAN, bidirectional operations are allowed at all the lines inside an auxiliary station for shunting movements. The Banker's algorithm is applied during the simulation process for deadlock avoidance. The example for demonstrating a deadlock test process in PULRAN is shown in Appendix C. It is implemented for the example shown in Section 4.4.3 (the improvement of using alternative route).

The analysis with potential state transitions and the solution with alternative routes are implemented in PULRAN, applied with longitudinal principle. Only if the deadlock-pending time of request exceeds a certain threshold, these improvements will be used. In PULRAN, most of train movements are inside the tested auxiliary stations, and thus it makes less sense to differentiate trains in different orders. For this reason, the improvement discussed in Section 4.4.2 has not been implemented. However, the implementation decision for the improvements is application-dependent. In other

applications other than shunting operations, additional improvement measures would need to be investigated.

Special considerations for shunting process are considered, with modified rules for authorizing train movements. The possibility of coupling for two trains or several railroad cars on the same track have to be taken into consideration. For this reason, a train is allowed to enter a blocked resource, when the resource is blocked by the supposed to be coupled train or the track is long enough for several railroad cars. Accordingly, the identification of available resources (AR) in deadlock tests is also adjusted. The details will not be discussed deeply here.

The performance of the simulation process is shown in Appendix D, tested at a normal computer (Intel Core2CPU 2.00 GHz, 1G RAM). For 10 minutes operation with 125 infrastructure resources, the average execution time is around 0.133 seconds (the time interval is taken as 4 seconds, implemented with analysis of potential state transitions and alternative route searching). It is hard to measure the required time for one step of deadlock test. The number of trains and the scale of the infrastructure are not large enough in the tested scenarios, and the required time of a deadlock test is less than 1 millisecond. Since the bottleneck of the system performance in the proposed synchronous simulation model is the process of deadlock avoidance, the system performance can be evaluated by the time complexity of the Banker's algorithm. The time complexity of the Banker's algorithm is $O(m \cdot n^2)$ for a system with $n$ processes and $m$ different kind of resources (see [BELIK, 1987]). Therefore, the execution of the synchronous simulation with the Banker's algorithm can be solved in polynomial time with acceptable system performance.

Deadlock problems can be avoided in synchronous railway simulations. The Banker's algorithm is used to test system safe state based on the approach of deadlock avoidance. Possible improvement measures are developed to increase the space of safe state and improve system performance. However, the false positives problem of the Banker's algorithm cannot be solved absolutely even with the proposed measures. It is impossible to identify the space of unsafe state that is the same as the space of deadlocks. In addition, the measures will also bring side effects to system performance. With the consideration of the implementation costs and the impaction to system performance, it is worth to explore more practical improvements under different operational conditions.

As described in Section 3.2.2.2, a request for entering a resource is authorized only when the conflict-free test and deadlock-free test are passed. After passing these tests, a requester is added to the candidate list of that resource. When there are many requesters that pass the tests are competing to enter an infrastructure resource, the requester with the highest priority will get the chance to enter. The determination of train priorities is discussed in Chapter 5.

# 5 Train Priorities and Allocating Infrastructure Resources in Simulative Dispatching

Dispatchers can be supported by an automatic dispatching system. Different automatic dispatching approaches (simulative models, analytical models, and heuristic models) have been discussed in Chapter 2. No matter which kind of automatic dispatching approach is utilized, the determination of train priorities is the key factor that influences the dispatching decisions. Once train priorities have been determined, the conflicts between two trains can be solved by giving precedence to the train with a higher priority value.

The influences of train priorities to the dispatching decision can be implicit or explicit within different automatic dispatching models. Both analytical and heuristic models are implicitly related with the strategy of determining train priorities. In an analytical model, the strategy of determining train priorities is integrated within the objective function. In a heuristic model, the strategy is integrated within the evaluation system. For the automatic dispatching system with simulative models, train priorities are determined explicitly. If a conflict occurs between two trains, a dispatching decision will be made explicitly after the train priority value of the concerned trains has been calculated.

From the point of view of the timing for determining train priorities, the determination of train priorities can also be differentiated as pre-determined and post-determined. In simulative models, train priorities are always determined before a conflict between two trains is solved, or before the train schedule is imported. Therefore, the train priorities in simulative models are pre-determined. In analytical models or heuristic models, train priorities are post-determined. The post-determined train priorities can only be obtained after optimization and evaluation processes have been finished. The comparison of train priorities for different automatic dispatching models is given in Table 5-1:

| Models | Influences to Decision | Timing of Determination |
|---|---|---|
| **Simulative Models** | Explicit | Pre-Determined |
| **Analytical/Heuristic Models** | Implicit | Post-Determined |

**Table 5-1 Comparison of Train Priorities Determination**

The implicit determination of train priorities in analytic models and heuristic models is not covered in this chapter. In analytical models, train priorities are usually defined in the form of decision variables, for example, binary priority decision variables are used to indicate train priorities in MARTIN's model (see Section 2.2.2). A similar approach is also used in the heuristic model proposed in Chapter 6.

In this chapter, only the determination of train priorities (explicit and pre-determined) in a synchronous simulation model is discussed. It is also possible to use the procedure in an asynchronous model. The determination of train priorities highly depends on the calculated train priority value, for which a practical calculation approach will be proposed. The calculated train priority value is used for allocating infrastructure resources in a synchronous simulation model as described in Section 3.2.2.2. In addition, the determined train priorities can also be in effect in requesting resources, which will be discussed in Section 5.3.

To determine priorities between two trains, a set of rules according to the operational disciplines and practices will be used. In this sense, the determination procedure can be regarded as a part of rule-based dispatching. The principle of rule-based dispatching is to build a complete set of dispatching rules and algorithms according to the predefined dispatching objectives. To realize the framework of a rule-based dispatching, the project "KosiDispo" (Consistent Dispatching in Planning and Operation, in German: *Konsistente Disposition in Planung und Betrieb*) is developed by the IEV for DB Netz AG (see [KROHN und LYNCH, 2009]).

In a fully-featured rule-based dispatching system, the strategies for determining train priorities are very comprehensive, with consideration to the optimization purposes based on certain optimization objective(s). The framework for rule-based dispatching is still in development, and it is not the goal of the simulation model implemented in this dissertation work. However, the rules to determine train priorities are always needed. The simplest principle of "first in first out" (FIFO) is not advisable. Therefore, a "limited" process with regard to train priorities is considered in this chapter, and further optimization measures will be adopted separately in a multi-level dispatching and optimization framework (see Chapter 6). For further development, it is worth to introduce the features of rule-based dispatching in the simulation model and the multi-level dispatching and optimization framework.

In this chapter, the basic principle to determine train priorities is introduced in Section 5.1. In Section 5.1.2 and 5.1.3, the proposed practical approach for calculating train

priorities is presented with an example. In section 5.2 the procedure of static priority arbitration for allocating resources and the dynamic resource requesting with priority parameters are discussed. In Section 5.3 a synchronous implementation that partially integrates with asynchronous simulation approaches is introduced.

## 5.1 Train Priorities in Dispatching

### 5.1.1 General Principles for Determining Train Priorities

Train priorities are context-dependent. They are always bound to certain dispatching objectives. However, there are some general principles that are valid for various situations.

One of the most important principles is the anti-discriminatory principle. Today it becomes an essential success requirement in railway operation. It is necessary to allow more competitors to access the railway network freely and equally in the same terms of conditions. As stated by EBA (the German Federal Railway Authority, in German: *Eisenbahn-Bundesamt*) in [FEDERAL RAILWAY AUTHORITY, 2009]:

“Whenever a multitude of competitors utilize the same infrastructure, all these must be offered the same chances and also be submitted to the same technical standards.”

The anti-discriminatory principle can guide to solve conflicts in a simulative automatic dispatching model explicitly. To ensure the principle, infrastructure managers allocate infrastructure to railway enterprises equally, based on unified and unbiased definitions of train priorities.

Some other dispatching principles can also be used to determine train priorities according to certain dispatching objectives. In DB Netz AG, the targets of a dispatching system are defined to recover system regularity as quick as possible through rescheduling a new dispatching timetable. The operational liquidity, punctuality and system capacity should also be taken into consideration. To satisfy these targets, the following guidelines are regulated (see [DB NETZ AG, 420.0105]):

1) Emergency relief trains take priority over other trains, and the operation control center regulates deviations.
2) Premium products (express trains) take priority over the rest of trains.

3) For the trains with the same product level, trains with higher speed in principle take priority over slower trains. Here the passenger tolerated transport/travel time is considered as well.
4) The trains at the lines that are in VzG (permitted speed regulation with special given usage conditions, in German: *Verzeichnis örtlich zulässiger Geschwindigkeiten*), or with respective conditions (e.g. suburban line), take priority over other trains.
5) At defined outlet section of freight trains, priorities should be granted to the freight trains.

If conflicts occur among different trains, the priorities may be determined according to these guidelines. However, it is not sufficient to deal all the complex situations according to these very general principles. A rule-based dispatching system can be employed to handle different scenarios or the combination of them.

An intuitive design for the determination of train priorities is to compare trains with a series of comparison criteria. However, it is not practical to determine train priorities depending on the qualitative comparisons, even if a simplified strategy for train priorities determination is used in this chapter. The qualitative comparisons will lead to an overcomplicated logic structure in software implementations, and further changes of the rules for priorities determination are hard to be adapted. Therefore, a quantitative method is desirable, and a proposed approach is presented in Section 5.1.2.

### 5.1.2 Calculation of Priority Values

With quantitative approaches, train priorities are transferred and calculated to priority values for all the trains. Therefore, train priorities can be determined by comparing the calculated priority values. The strategies for priority determination can be integrated within priority calculation modules flexibly. Equipped with configurable parameters, it is possible to implement a calculation framework, in which indicators for calculation priority values can be added or removed without changing the basic structure of the calculation.

Such a flexible structure is very necessary to be introduced in priorities calculation processes. Since the criteria for comparing train priorities are changed year by year, a calculation procedure with fixed indicators is inflexible when new evaluation standards are applied.

The calculation framework proposed in this chapter follows the methodology used in[MARTIN, 1995], with consideration to the possible changes of evaluation indicators. The following indicators are defined in MARTIN's model as the evaluation criteria of train priorities:

- Train product type
- Passing or stopping criteria
- Punctuality criteria
- Possibility of reduction of delay in further movements

In the procedure of train priorities calculation, the priority value of a train is computed based on two types of indicators:

- Class-oriented indicators: calculated for a train group with common characteristics. The value is determined by absolute value or is categorized into subclasses according to certain evaluation criteria.
- Train-oriented indicators: calculated for a specific concerned train. The value is determined by absolute value for an individual train.

In MARTIN's model, the train product type, passing or stopping criteria, and punctuality criteria are class-oriented indicators, and the possibility of reduction of delay is a train-oriented indicator. Class-oriented indicators should be calculated at first, with respect to the predefined sequence the indicators. The calculation of train-oriented indictors is performed after the calculation for class-oriented indicators is completed. The processes for determining train priorities are shown in Fig. 5-1:

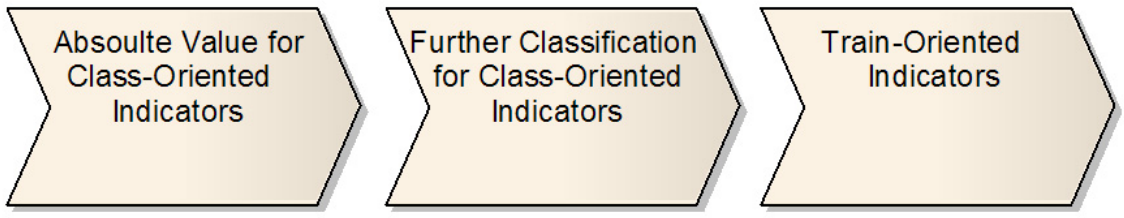



**Fig. 5-1 Determination Processes of Train Priorities**

The calculation of class-oriented indicators starts from the indicators with absolute value that can be usually determined according to some monetary attributes. The weighted costs of delay are suitable benchmarks to evaluate train priorities (see [MARTIN, 2008]). In case the up-to-date values of the overall weighted cost are not available, the line price of each train product can be used as an alternative.

Further calculations of the class-oriented indicators will include several steps. In each step a new indicator will be introduced. The new introduced indicator is used to classify the existing train classes into several sub-classes. For the system with the number of $C$ classes, the $i^{th}$ introduced indicator that categorizes a train class into $C_i$ types will lead to $C \cdot C_i$ sub-classes. The new added sub-classes will be inserted into the existing classes in average. Given $V_j$ $(1 \le j \le C)$ as the priority value for class $j$ before the new indicator is introduced, the priority value $V'_k$ $(1 \le k \le C \cdot C_i)$ for these new sub-classes will be set as:

$$V'_k = \begin{cases} \dfrac{V_1 \cdot k}{C_i}, & 1 \le k < C_i \\ V_j, & k = j \cdot C_i \\ V_{j-1} + \left(V_j - V_{j-1}\right) \cdot \left[\dfrac{k}{C_i} - (j-1)\right], & (j-1) \cdot C_i < k < j \cdot C_i \text{ and } j > 1 \end{cases} \tag{5-1}$$

An example of classifying with new introduced indicators ($C_i = 3$) is demonstrated in Fig. 5-2. For the new class definition, class 1 and 2 belong to the situation $1 \le k < C_i$, class 3, 6, 9 and 12 belong to the situation $k = j \cdot C_i$, and class 4, 5, 7, 8, 10 and 11 belongs to the last situation $(j-1) \cdot C_i < k < j \cdot C_i$ and $j > 1$.

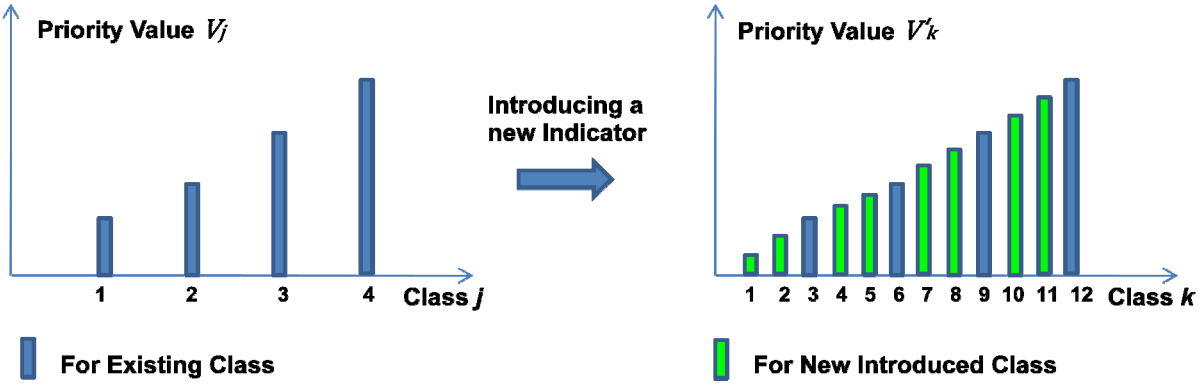



**Fig. 5-2 Illustration of Introducing a New Class-Oriented Indicator**

After the classification and calculation for class-oriented indicators are finished, the train-oriented indicator(s) will be calculated for each individual train. The value of the train-oriented indicator(s) is added (possibly with an empirical constant parameter) to the class-oriented indicator for the train-class the train belongs to. The value of train priority for an individual train is determined finally.

$$P_z = V_k + \alpha \cdot V_z \tag{5-2}$$

Notations used:

$P_z$ priority value for train Z

$V_k$ class-oriented indicator value for the train class $k$

$V_z$ train-oriented indicator value for train Z

$\alpha$ an empirical constant value

The empirical constant value is used to normalize the value of $P_z$ in a reasonable arrange. The principle is to regulate the value between 0 and the maximum value of $V_k$ so that $0 \leq P_z \leq \max(V_k)$.

### 5.1.3 An Example for Calculating Train Priorities

Based on the indicators defined in [MARTIN, 1995] (train product type, passing or stopping criteria, punctuality criteria, and the possibility of reduction of delay), train priorities are calculated in this section.

The line price factors of each train product type are shown in Table 5-1 (see [DB NETZ AG, 2007]). The factors are used as absolute class-oriented indicators:

| Class $j$ | Factor ($V_j$) | Train Product |
|---|---|---|
| 1 | 0.50 | Freight Transport-Feeder-Line |
| 2 | 0.65 | Passenger Transport-LZ-Line, Freight Transport-LZ-Line (LZ: Locomotives or traction units, in German: *Lokzügen*. LZ line enables the operations of dispatched locomotives or traction units) |
| 3 | 1.00 | Passenger Transport Economy Line, Freight Transport Standard line |
| 4 | 1.65 | Passenger Short Distance Transport Cycled Line, Passenger Long Distance Transport Cycled Line, Freight Transport Express Line |
| 5 | 1.80 | Passenger Transport Express Line |

**Table 5-2 Line Price Factor for Train Product in DB Netz AG**

Five classes are defined with different price factors. The higher the value of the factor, the higher the priority a train class should receive. Further classifications are carried out for passing or stopping criteria and punctuality criteria. With the consideration of passing or stopping criteria, the original 5 classes are further classified into 10 classes as shown in Table 5-3 (the new classes are marked with bold font):

| Class k | Value ($V_k$) | Class of Train Product | Stopping/Passing Criteria |
|---|---|---|---|
| **1** | **0.25** | 1 | stopping |
| 2 | 0.50 | | passing |
| **3** | **0.575** | 2 | stopping |
| 4 | 0.65 | | passing |
| **5** | **0.825** | 3 | stopping |
| 6 | 1.00 | | passing |
| **7** | **1.325** | 4 | stopping |
| 8 | 1.65 | | passing |
| **9** | **1.725** | 5 | stopping |
| 10 | 1.80 | | passing |

**Table 5-3 Priority Value with Consideration of Passing/Stopping Criteria**

Furthermore, the punctuality criteria are considered. The classification and the priority values are shown in Table 5-4 (the new classes are marked with bold font):

| Class k | Value ($V_k$) | Class of Train Product | Stopping/Passing Criteria | Punctuality Criteria |
|---|---|---|---|---|
| **1** | **0.125** | 1 | stopping | Delayed |
| 2 | 0.25 | | | Punctual |
| **3** | **0.375** | | passing | Delayed |
| 4 | 0.50 | | | Punctual |
| **5** | **0.5375** | 2 | stopping | Delayed |
| 6 | 0.575 | | | Punctual |
| **7** | **0.6126** | | passing | Delayed |
| 8 | 0.65 | | | Punctual |
| **9** | **0.7375** | 3 | stopping | Delayed |
| 10 | 0.825 | | | Punctual |
| **11** | **0.9125** | | passing | Delayed |
| 12 | 1.00 | | | Punctual |
| **13** | **1.1625** | 4 | stopping | Delayed |
| 14 | 1.325 | | | Punctual |
| **15** | **1.4875** | | passing | Delayed |
| 16 | 1.65 | | | Punctual |
| **17** | **1.6825** | 5 | stopping | Delayed |
| 18 | 1.725 | | | Punctual |
| **19** | **1.7625** | | passing | Delayed |
| 20 | 1.80 | | | Punctual |

**Table 5-4 Priority Value with Consideration of Punctuality Criteria**

In MARTIN's model, the possibility of reduction from delays is used as a train-oriented indicator. The possibility of reduction from delays can be regarded as a "negative" indicator. A train should receive a low priority value if it has a high possibility of reduction from delays along its further train running.

The value of the possibility of reduction from delays will be calculated as follows (for detailed information see Section 4.1.3, in [MARTIN, 1995]).

$$C_{\mathrm{RED}}[-] = \frac{\bar{V}_{\mathrm{zRest}}[\mathrm{km/h}] \cdot T_{\mathrm{RRest}}[\mathrm{h}]}{L_{\mathrm{Rest}}[\mathrm{km}]}\,, \qquad \text{with } L_{\mathrm{Rest}} \geq 1\mathrm{km} \tag{5-3}$$

Notations used:

| | |
|---|---|
| $C_{\mathrm{RED}}$ | value of the possibility of reduction from delays |
| $\bar{V}_{\mathrm{zRest}}$ | average speed in the rest of the route |
| $T_{\mathrm{RRest}}$ | sum of the scheduled recovery time in the rest of the route |
| $L_{\mathrm{Rest}}$ | length of the rest of the route |

Finally, the train priority value can be calculated through Formula 5-2. Train priorities will be determined based on the calculated train priority values.

## 5.2 Static Resource Allocation and Dynamic Resource Requesting

To arbitrate conflicts (or potential conflicts) in synchronous simulation, two modes of train priorities determination can be used: static resource allocation and dynamic resource requesting. The difference between these two modes is the timing for determining train priorities.

With static resource allocation, train priorities should be determined *when* a conflict is arising. The determination of train priorities can be based on the calculated priority values or other principles (the simplest principle would be the first in first out principle). Static resource allocation is also applicable in an asynchronous model to solve conflicts.

In a synchronous simulation model, it is inefficient if only static resource allocation is applied. Considering the situation as shown in Fig. 5-3, a potential conflict may happen between two trains Z1 and Z2 when requesting to a common resource R, and Z2 is designated to take the priority over Z1 according to the calculated priority value. Assume at a certain time interval n in a synchronous simulation model, Z1 reaches its requesting point for requesting R (determined by the minimum requesting distance), while Z2 has not reached its requesting point yet. Train Z1 gets the right for entering R and blocks R to other trains. In the next time interval n+1, Z2 arrives the requesting point, it has to wait until Z1 release R, although Z2 should receive a higher priority than Z1.

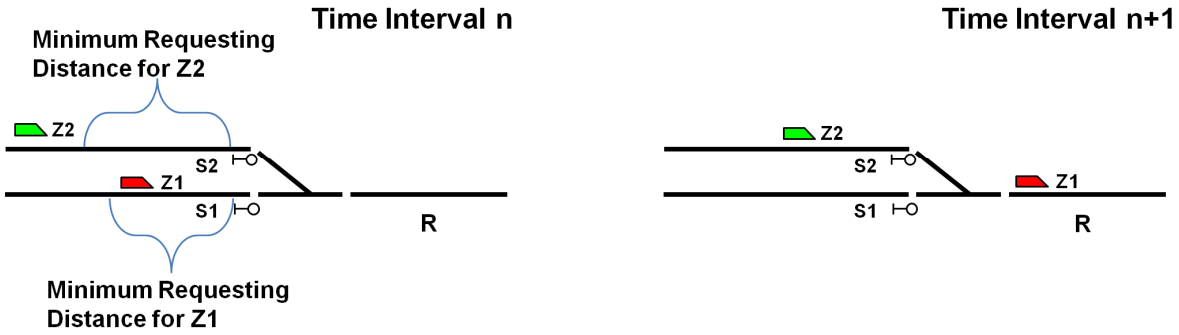



**Fig. 5-3 Static Resource Allocation without Considering Potential Conflict**

In a synchronous model, it is rare that two trains are requesting the same resource at an exact same time interval if it is short enough. Lack of ability to predict potential conflicts will lead to inefficiency in the mode of static resource allocation. A train with a high priority value may lose the chance to enter the requested resource earlier than other low-priority trains, just because it arrives at its requesting point later than others. To solve the problem, the train with a high priority value should able to request resources earlier than other low-priority trains. Therefore, the mode of dynamic resource requesting is used. In this mode, a train with a high priority can acquire the chance to block an infrastructure resource *before* a conflict arises. As discussed in Section 3.2.2.1, the timing of requesting resources is influenced by the requesting distance. With integrating the priority values into the calculation of requesting distance, the train with a high priority value will get more chance to enter the requested resource by shifting the timing to request the resource earlier. The requesting distance in the dynamic resource requesting mode is:

$$D_{\mathrm{req}} = D_{\mathrm{min}} \cdot \left[1 + \frac{P_{\mathrm{z}}}{\max\,(V_k)}\right] \tag{5-4}$$

Notations used:

| | |
|---|---|
| $D_{\mathrm{req}}$ | requesting distance in the dynamic resource requesting mode |
| $D_{\mathrm{min}}$ | predicted minimum requesting distance in the time interval |
| $P_{\mathrm{z}}$ | priority value for train Z |
| $V_k$ | class-oriented indicator value for the train class $k$ |

Since the value of $P_{\mathrm{z}}$ is normalized between 0 and $\max(V_k)$, the maximal requesting distance for the highest priority train will be doubled from the original minimum requesting distance. In the dynamic requesting mode, the train with high priority will be arranged to request the next resource at an earlier time than the low priority trains.

Hence, the solution quality of the synchronous simulation is improved by solving potential conflicts in advance with consideration to train priorities.

## 5.3 Time Frame – A Solution Combined with Asynchronous Simulation

In a synchronous simulation, the movements of all trains are simulated simultaneously, while in asynchronous simulation trains are introduced into simulation ordered by train priorities. Without a well-elaborated rule-based dispatching system, the anti-discriminatory principle may be violated in an asynchronous simulation model by the predefined and possibly biased priorities. On the contrary, the principle can be ensured for all trains equally in a synchronous simulation model with a unified set of calculation rules.

However, there are some situations, where a predefined priority should be respected definitely in synchronous simulation models. These situations can be:

- An emergency relief train should take the precedence over other trains
- A maintenance work is scheduled with related vehicles
- Other high-priority operations that are not allowed to be interrupted (e.g. for political or military reasons) are scheduled on a certain lines with fixed time frame
- A line with fixed time frame has been purchased by a certain client, for whom the operations out of the given time frame are not allowed

The solution with defined time frames is used to ensure pre-defined priorities being complied. According to the application context, there are two types of time frames. If the train with the highest priority is designated, it will be introduced at first with fixed blocking time for each required infrastructure, and the *blocked time frames* are defined to prevent other trains from entering. The *permitted time frames* are defined for a train or a certain group of trains, when the rights of using certain infrastructure are fixed. The train belonging to a certain train group is only allowed to enter the infrastructure resource during the pre-defined time frame. In the project PULRAN, the permitted time frames are used to regulate a train running from an auxiliary station into main tracks (see Fig. 5-4).

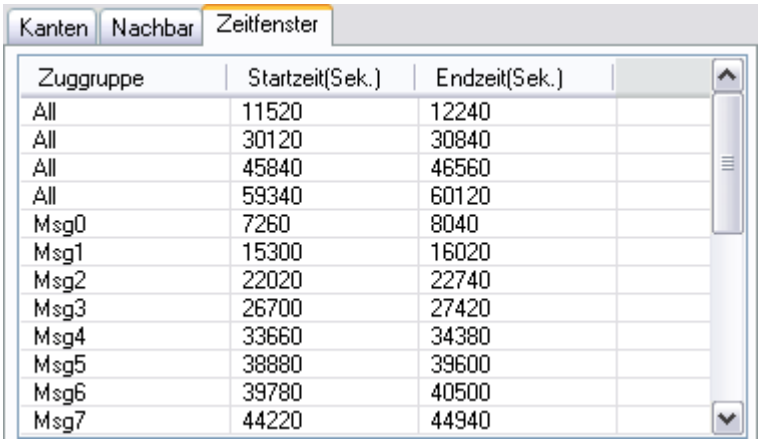



**Fig. 5-4 Permitted Time Frames for an Infrastructure Resource in PULRAN**

After time frames are defined, trains with pre-defined priorities can be introduced into a simulation system asynchronously. The integration of asynchronous features into a synchronous simulation model can increase system flexibility. In this way, synchronous simulation and asynchronous simulation can be used as a complement with each other.

Determination of train priorities is always related with certain dispatching objectives. A good design for priority determination will improve quality of the simulation result. In an optimization process, train priorities are very important decision variables as well. A multi-level dispatching and optimization model that focuses on train priorities on a macroscopic level will be proposed in Chapter 6.

# 6 Multi-level Dispatching and Optimization

A hybrid approach is recommended in Chapter 2 to establish a reliable automatic dispatching model with optimized solutions. After the execution of a simulation process, a basic dispatching solution is obtained as the basis for further optimization process, which is performed to improve the solution quality. In Chapter 3, 4 and 5, a simulation model based on synchronous mode was discussed in details. Further optimization can be carried out with analytical or heuristic method, and the framework of a dispatching and optimization model is proposed in this chapter.

Performance issues should be taken into consideration in an optimization process. For a large scale control-territory with a huge amount of train movements, if a model comprises too detailed information and operations, the computational complexity would be unacceptable. Therefore, special designs should be adopted to reduce the complexity of the optimization model.

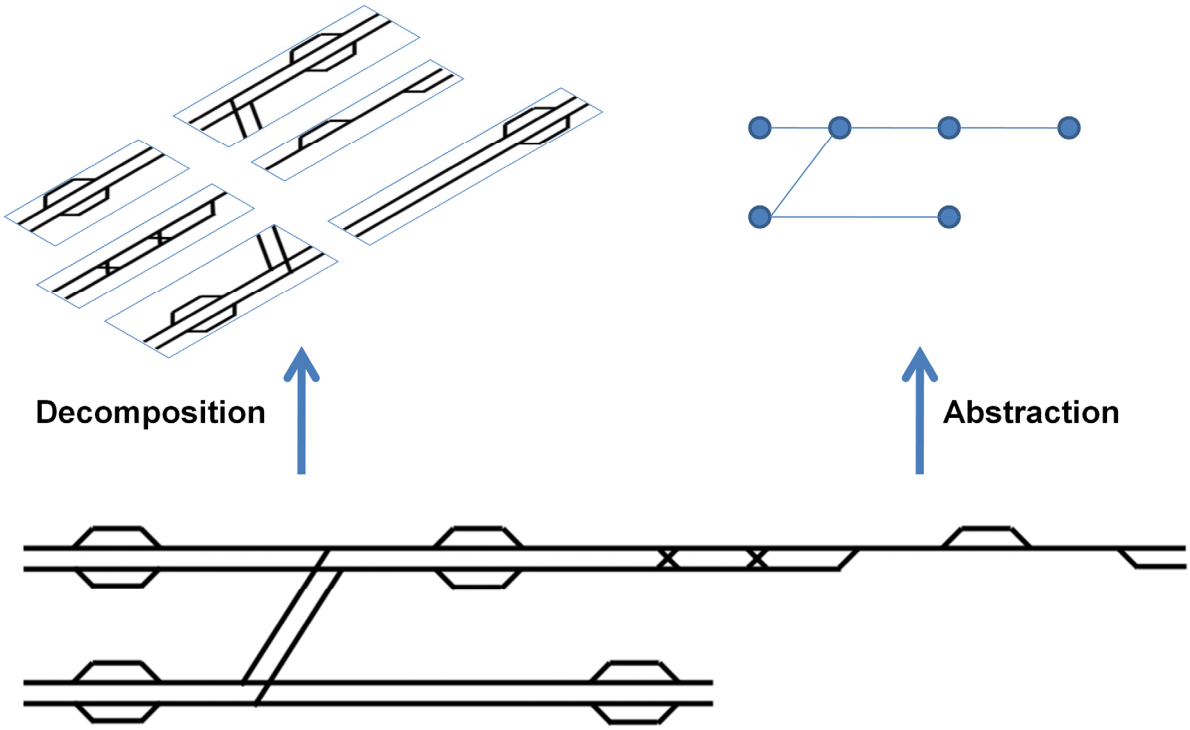



**Fig. 6-1 Decomposition and Abstraction Pattern for Optimization**

Two possible patterns, decomposition and abstraction, are demonstrated in Fig. 6-1. With decomposition pattern, a large network is divided into many sub-networks, and the big optimization problem is decomposed into several small optimization problems accordingly. For a model with abstraction pattern, the microscopic data model is mapped into macroscopic level. An overall optimization process is executed on a

macroscopic level with low computational complexity. A detailed dispatching timetable is generated afterwards on a microscopic level (for the discussions about macro- and microscopic levels in infrastructure model see Section 6.2.1).

The most significant problem for decomposition pattern is the extra coordination requirements, due to the lack of an overview of different decomposed areas. Additional constraints are thus introduced into the model, which results in considerable computation costs. For this reason, the abstraction pattern with multi-level framework is suggested. In Section 6.1 the framework and business process of a multi-level dispatching and optimization is presented. The optimization model on a macroscopic level and the process of generating the dispatching timetable on a microscopic level are discussed in Section 6.2 and 6.3 respectively.

The framework of the multi-level dispatching and optimization proposed in this dissertation should be differentiated from the multi-level railway operation optimization system and method, which is published as a patent (EP 1 687 196 B1) and granted in year 2008 (see [KUMAR et al., 2008]). The levels defined in the patent include railway infrastructure level, railway track network level, individual train level within the network, consist level with the train, as well as the individual locomotive level within the consist. The goal of the patent is to integrate and optimize all the aspects of railway operation with a multi-level, system-wide approach. The “multi-level” proposed in this dissertation concentrates on the macroscopic and microscopic levels in railway infrastructure. It is designed to overcome the problem of the high computational complexity in a large-scale dispatching territory. Within the multi-level dispatching and optimization framework, a derived basic dispatching solution is further optimized and elaborated in macroscopic and microscopic level respectively (see Section 6.1).

There is a wide variety of optimization techniques available. The techniques adopted in this chapter just fit the requirements of the proposed multi-level dispatching and optimization framework, which should be designed according to its optimization objective(s), but not a specific optimization technique.

## 6.1 The Framework of a Multi-level Dispatching and Optimization

In a multi-level dispatching and optimization framework, a synchronous simulation is executed in order to create the *basic dispatching solution*. To reduce the computational complexity, the optimization process only focuses on a macroscopic level. On the macroscopic level, the infrastructure and operations are abstracted. Too detailed

infrastructure information and operations are not considered. The abstraction of the infrastructure data model and the operation concepts on a macroscopic level will be introduced in Section 6.2.1 and 6.2.2. Within the abstracted macroscopic model, a macroscopic optimization is executed, and an optimized dispatching schedule on a macroscopic level, which is called an *optimized macroscopic solution*, is established. Based on such an optimized macroscopic solution, a detailed dispatching timetable can be built up on a microscopic level after a further elaboration process. The final solution generated from the elaboration process is called an *elaborated dispatching solution*. The processes within a multi-level dispatching and optimization framework are demonstrated in Fig. 6-2.

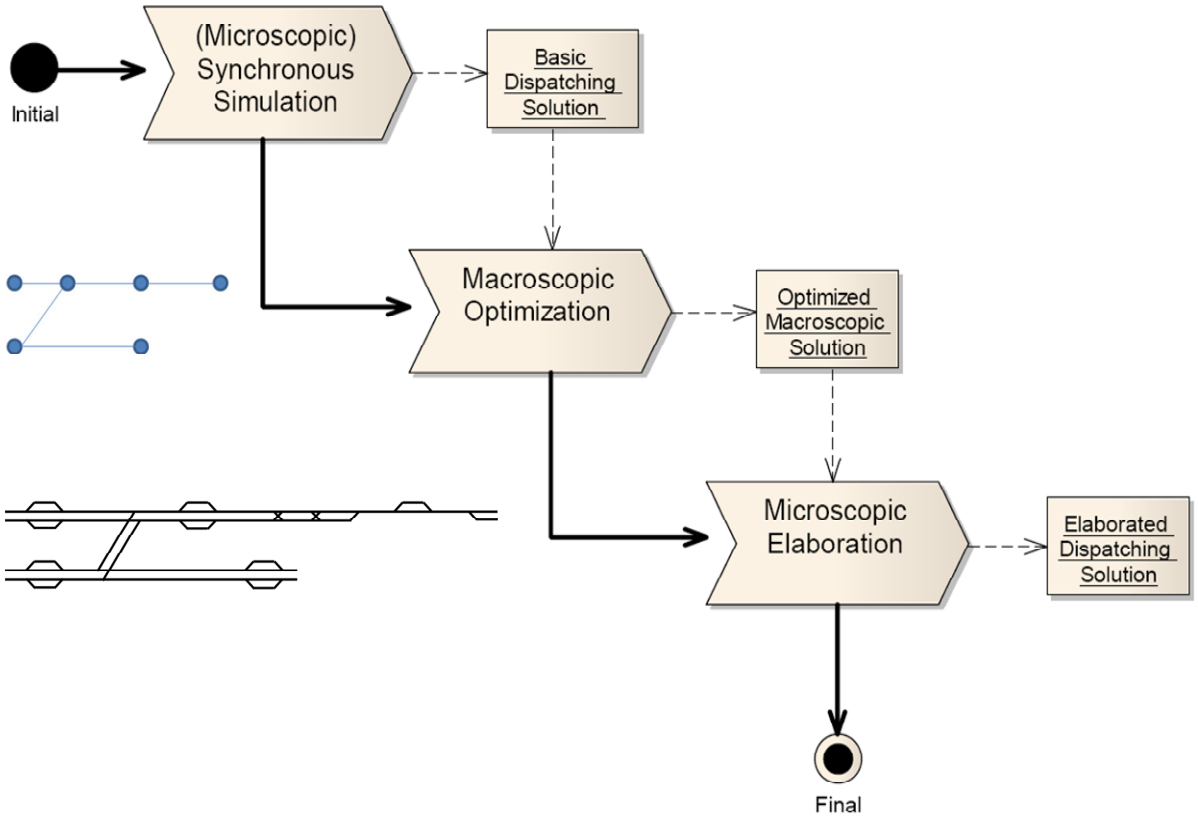



**Fig. 6-2 The Processes of Multi-level Dispatching and Optimization**

The framework and the business process of a multi-level dispatching and optimization exactly fit the structured hierarchy in modern railway operation control center (see Section 2.1.1). Within a multi-level dispatching and optimization framework, a preliminary dispatching schedule can be obtained through an optimized global perspective with acceptable computation expenses. The computation workloads are distributed into different levels, which is similar to the sharing of workloads between dispatchers and operators in operation control center. By isolating the microscopic de-

tails from the strategy optimization process, different concerns are encapsulated at different levels.

From the point of view of software architecture, the decoupled system can provide great flexibility for configuration, maintenance, and further development. A multi-level dispatching and optimization framework can be regarded as an assembly of several independent components that interact with each other according to predefined interfaces. The modification or upgrade of the implementation for a component will not influence others, thus, the system can be configured and deployed at design time or runtime in a flexible way. For example, a new implemented macroscopic dispatching component, based on another optimization technique, can replace the old one without bringing any additional development costs for microscopic elaboration. Furthermore, when several optimization implementations are available for macroscopic optimization, a user can switch the configuration among different implementations dynamically at runtime according to the application contexts: an analytical algorithm may be employed for a small network with few train movements, and a heuristic implementation will be in effect when the problem size reaches a certain threshold. An example of the software architecture for a multi-level dispatching and optimization framework is shown in Fig. 6-3.

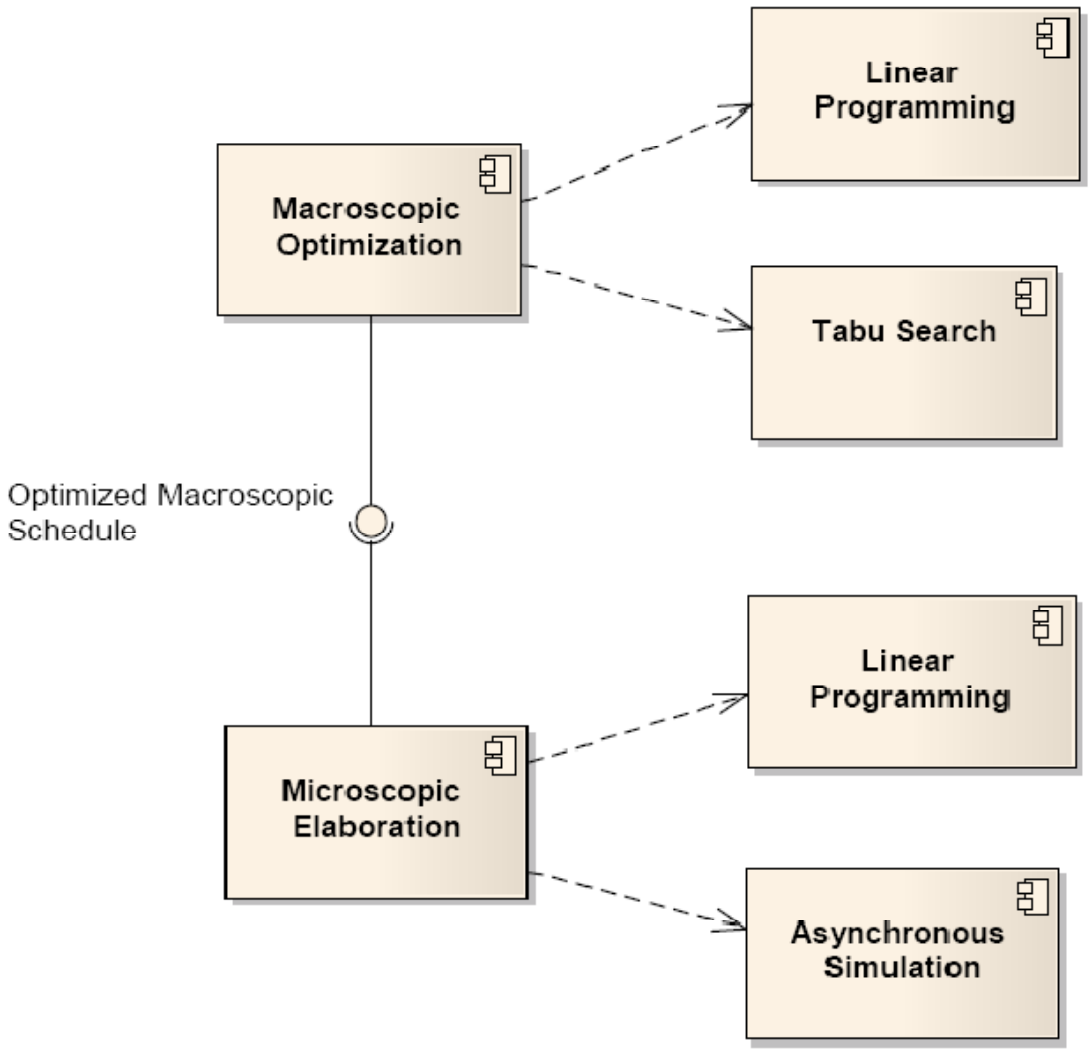



**Fig. 6-3 Possible Software Architecture of a Multi-level Dispatching**

The proposed model in this dissertation is only based on 2 levels (macroscopic and microscopic levels) as a theoretical discussion. More complicated dispatching structure, with more than two levels, is subjected to the concrete management structure in reality. They are not discussed in this dissertation.

As introduced in Chapter 2, the possible dispatching measures of an automatic dispatching system are time-related dispatching and location-related dispatching. At the beginning of the design, it is a very important decision to determine the possible adopted dispatching measures for different dispatching and optimization processes.

During the simulation process, the time-related dispatching is used when a delayed train is trying to catch up its original time schedule. For each single processing step in a synchronous simulation, a delayed train is running with the speed that results in an adjusted running time at a non-junction type resource as:

$$t_{\text{adjusted}} = \max\left(t_{\text{s}} - t_{\text{d}},\ t_{\text{min}}\right) \tag{6-1}$$

Notations used:

$t_{\text{adjusted}}$ adjusted running time

$t_{\text{s}}$ scheduled running time

$t_{\text{d}}$ delay of the train compared with original timetable, must not be negative

$t_{\text{min}}$ required minimal running time

After the simulation is finished, the effects of the adjusted running time will be checked. If a train can be recovered from delays, the scheduled running time will be updated with the adjusted value $t_{\text{adjusted}}$. For each train, the unscheduled waiting time caused by obstructions will also be recorded as a parameter to evaluate the level of delays.

Meanwhile, the measure of location-related dispatching may also be used, if it is necessary to use an alternative route (e.g. for solving deadlocks or searching a new route in case of infrastructure defects). The scheduled destinations should be respected while searching for a new route (see the discussion in Section 4.4.3).

In a dispatching process, overtaking is the most powerful measure along a certain line that can be implemented as a location-related dispatching measure. Other measures, including detouring, shortening the scheduled train path, and rolling stocks/personnel dispatching, involve a great number of aspects and factors in decision making processes. These comprehensive human intellectual activities strongly depend on optimization objectives, which are not covered in this dissertation work. Therefore, only time-related dispatching and overtaking are used in the proposed optimization model. The rest of the dispatching measures are designated to be left for manual dispatching. Some dispatching decisions, e.g. detoured train paths, are utilized as inputs to the optimization model. The design of separating different dispatching measures into different modules can also be seen in the project PULZURE (see Section 2.1.4). In Table 6-1, a summary of the possible dispatching measures for different dispatching processes is given.

| Dispatching Process | | Dispatching Measures |
|---|---|---|
| Manual Dispatching | | Detouring<br>Shortening the Schedule Train Path<br>Rolling Stocks/Personnel Dispatching |
| Multi-level Automatic dispatching | Synchronous Simulation | Time-Related Dispatching<br>Location-related Dispatching (the process of route searching should respect scheduled destinations) |
| | Macroscopic Optimization | Overtaking<br>Time-Related Dispatching |
| | Microscopic Elaboration | Time-Related Dispatching |

**Table 6-1 Dispatching Measures in Different Processes**

It should be emphasized here that the design of the dispatching measures for each process is not fixed. The development progress of automatic dispatching is incremental. More and more dispatching measures will be transferred from manual dispatching to automatic dispatching if the respective algorithms are mature. The design decision will vary with different optimization objectives as well.

## 6.2 Macroscopic Dispatching and Optimization

An optimization process always highly depends on its objective functions. In this dissertation, how to define the objective functions will not be discussed. The objective function used in MARTIN's linear programming model (see Section 2.2.2) is applied. Upon this objective function a conceptual optimization model is proposed to illustrate the framework of multi-level dispatching and optimization. The optimization techniques may vary according to different optimization objectives. However, the basic concepts of multi-level framework can also be applied for other applications.

It is necessary to model the infrastructure network as the basis to start a macroscopic optimization process. The optimization model can be formulated after the principles of macroscopic optimization are specified. To take the advantages of linear programming, the model is designed as a linear system that can be solved by integrating available software tools. A heuristic method based on tabu search is also proposed to find an optimal solution for a large scale network with many train movements.

### 6.2.1 Infrastructure Modeling for Macroscopic Dispatching and Optimization

In principle, the infrastructure model used for macroscopic estimation and optimization follows the definition of macroscopic models given in [RADTKE, 2008]. A macroscopic infrastructure network consists of nodes and links. Stations or junctions are defined as nodes. Two nodes are connected with a link. More detailed definitions are specified to meet the dispatching-specific requirements as follows:

- Loop nodes
- Junction nodes
- Open track sections

A node is the place where dispatching measures can be made. In the automatic dispatching system proposed in this chapter, the dispatching measures include time-related dispatching and overtaking. Not all nodes are able to be arranged for overtaking. Therefore, the nodes are differentiated to two types: loop nodes and junction nodes. Only at a loop node a train may overtake or pass another one if the track is long enough. A junction node includes at least one junction-type resource, but it is impossible to arrange an overtaking operation inside the node.

The concept of links defined for macroscopic dispatching refers specially to the infrastructure resource(s) connecting two nodes without junction-type elements. To make the concept clearly, the term “open track section” is utilized instead of “link”. An open track section is used in a macroscopic model, while a track section (see Section 3.1.1) is used in a microscopic model. By introducing the concept of open track sections, the macroscopic dispatching model can be simplified and established (see Section 6.2.2). A direction attribute is always defined for an open track section. The operations of an open track section can be mono-directional or bidirectional.

The term “node” is used as a general concept for a loop node or a junction node, while term “knot” is used as a general concept for a node or an open track section. The class diagram for macroscopic infrastructure concepts is shown in Fig. 6-4.

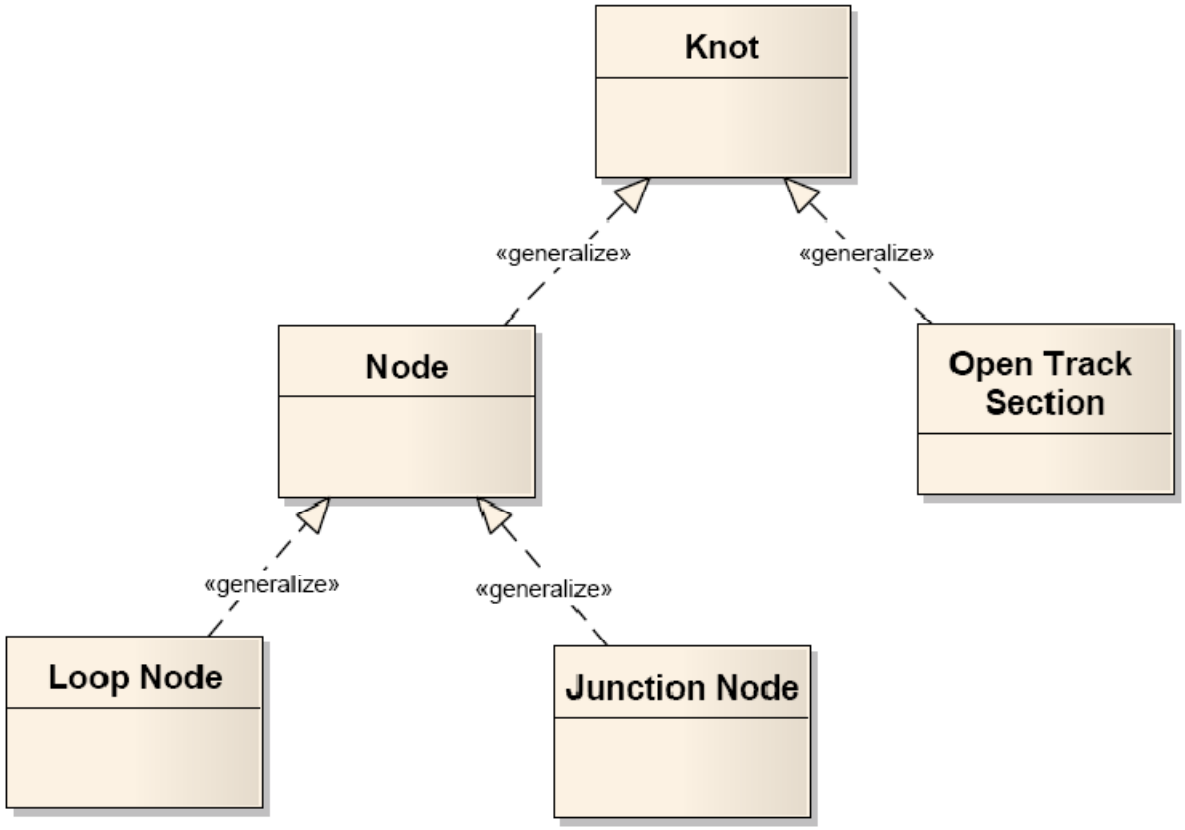



**Fig. 6-4 Class Diagram for Macroscopic Infrastructure Concepts**

Loop nodes, junction nodes, and open track sections are the basic elements of a macroscopic network that can be derived from an existing microscopic or mesoscopic infrastructure model. For example, the infrastructure network of DB Netz AG is divided into several small operational sites (abbreviated as BS, from German term "*Betriebstelle*"). For a BS with only non-junction type infrastructure resources, it can be abstracted as an open track section. The rest of them are nodes. The type of the node can be identified by looking up the predefined operational route (in German: *Betriebsfahrweg*). A node is a loop node if a train can overtake or pass another train by taking a different operational route inside the BS. Therefore the macroscopic network can be constructed from the given microscopic network depending on the definition of BS.

The basic dispatching solution derived from a microscopic simulation can be optimized on a macroscopic level. By introducing the concepts of knots, nodes, and open track sections, a macroscopic infrastructure model is generated from microscopic level by abstraction and aggregation.

### 6.2.2 Principles of Macroscopic Optimization

Several different preferences, such as punctuality, fluency or costs of rolling stocks and staffs, may result in varied definitions for an “optimal” solution. As discussed at the beginning of Section 6.2, the optimization objective in the proposed model still aims to minimize overall weighted trip time.

The essential of a dispatching process is to reschedule a new dispatching timetable. The following components are involved to fulfill a rescheduling process:

- Train routes/Macro-paths
- Sequences among several trains for common infrastructure resources in cases of conflicts
- The scheduled running time and dwell time
- Unscheduled waiting time

Here the concept “macro-path” should be distinguished with the microscopic term “train route” (the discussion of train routes is given in Section 4.4.3). In this dissertation, the concept “macro-path” refers to a list of macroscopic knots ordered according to the given timetable (original or rescheduled timetable). A knot may be a node or an open track section. In German practices, it can be regarded as a BS (see Section 6.2.1). From the given timetable, the macro-path can be obtained at macroscopic level. Within a certain macro-path, there may be various train routes. For example, inside a loop node a train route may be changed if an overtaking takes place.

Train sequences are the most concerned variables in the proposed macroscopic optimization model. They are decisive to determine whether a train should overtake another one inside a loop node, or whether a train should wait before another train passes a certain knot. Before an optimization process starts, the train sequences can be concluded from the basic dispatching solution generated by a simulation process. *The core of the proposed optimization algorithm is to find out an optimal solution to achieve a minimal weighted overall trip time through the adjustment of train sequences*.

More exactly, for a certain knot in one direction, train sequences are described as two train lists ordered by time: one for arriving sequence, and another one for departing sequence. For an open track section in one direction, the departing sequence is always the same as the arriving sequence, as there is not any chance to change train sequences inside an open track section. The sequences of open track sections are sufficient to describe the complete train sequences of the whole network. For an

open track section, the train sequence for one direction reflects the departing sequence of its starting node, as well as the arriving sequence of its ending node.

The possible adjustments of train sequences take place at nodes. The adjustments can happen in the following situations:

- Time-related dispatching through introducing additional unscheduled waiting time before or inside a junction node (see Fig. 6-5)
- Location-related dispatching through overtaking inside a loop node if possible (see Fig. 6-6)

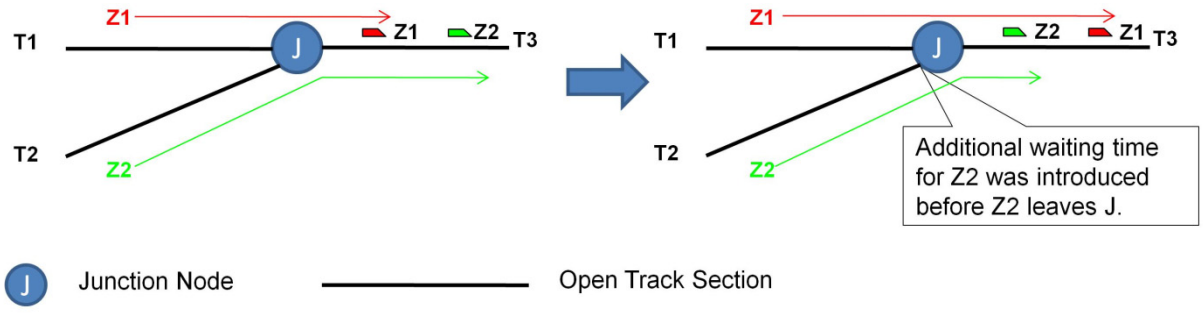



**Fig. 6-5 Change Train Sequence by Introducing Additional Waiting Time**

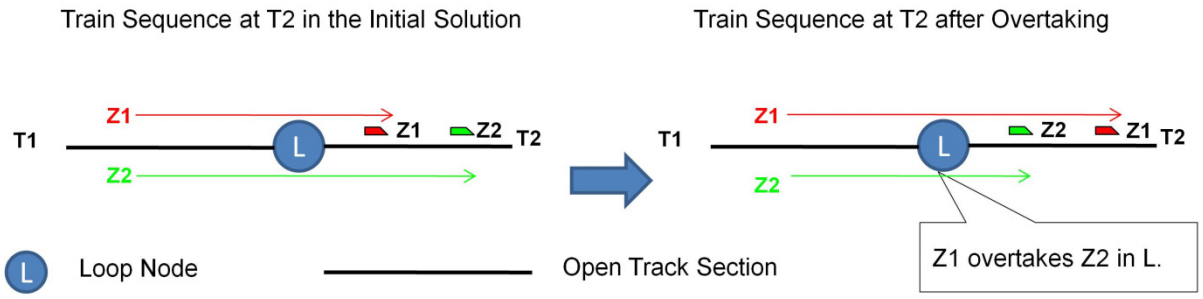



**Fig. 6-6 Change Train Sequence by Overtaking**

In the lines with bidirectional operations, a passing operation at a loop node can be regarded as two operations:

1) A train is waiting for another train to pass at the loop node, where a time-related dispatching takes place.
2) Another train passes the waiting train. A kind of “overtaking” in a different direction takes place.

To simplify the description, the passing operations are considered the same as the operations of time-related dispatching and overtaking in this dissertation. Therefore, the adjustment is not illustrated.

In macroscopic optimization, only the introduction of additional unscheduled waiting time is considered as time-related dispatching. The measure to improve running speed to recover from delays is used in the synchronous simulation at the beginning. The scheduled running time in an open track section will be updated if recovery for a delayed train is possible during the simulation (see equation 6.1).

Compared with MARTIN's theory that is constructed upon blocking time model, the outputs of a macroscopic optimization model are more general with less detail. An exact blocking time calculation will be postponed in the elaboration process after the macroscopic optimization is finished. The scheduled running time and dwell time are collected from the basic dispatching solution, and the additional unscheduled waiting times are only estimated as assistance variables, which will be used to evaluate the result of the solution (see Section 6.2.3 and Section 6.2.4.3).

Unscheduled waiting can take place in a node or an open track section. When overtaking or passing happens inside a loop node, additional waiting time may also be introduced for the train to be overtaken or passed. To simplify the optimization algorithm, the unscheduled waiting time is only modeled inside nodes, even additional stops take place in open track sections, or a train may not be allowed to stop in a junction node.

It is no harm to apply the simplification in a macroscopic optimization, while very detailed running time calculation is not concerned at this phase. Taking the example shown in Fig. 6-5, train sequence at T3 is changed and Z1 takes the priority over Z2 by introducing additional waiting time for Z2. There are many possibilities to achieve the change of train sequence, such as to stop Z2 at some position in T2 before it enters J, or to reduce running speed of Z2 in T2 in order to give the priority to Z1, or a combination of both. No matter which kind of arrangement is applied, the macroscopic optimization is only interested in the fact that Z2 should wait in front of J for a certain time period until Z1 leaves. Since it is impossible to determine a detailed movement arrangement, it is convenient to model all of the unscheduled waiting time inside nodes, as if all of the additional waiting had taken place there. A concrete calculation of distributing additional waiting time will be executed in the microscopic elaboration (see Section 6.3), which is often combined with an energy-efficient driving style and driver support system.

The purpose of changing the train sequence is to give precedence to faster trains, so that obstructions produced by slower trains can be avoided and possible delays can

be reduced. By calculating the total weighted trip time, the effects of the adjustments are evaluated. As all the unscheduled waiting time will be modeled inside nodes, the trip time on open track sections can be easily obtained by assigning scheduled trip time. Attention should be paid to avoid conflicts between two trains. In MARTIN's model, conflict avoidance is implemented by regulating the departing time of the later train based on blocking time models (see Section 2.2.2). It can be simplified in macroscopic optimization by separating trains according to minimum line headways. Differentiated by successive movements and opposite movements, two kinds of line headways – arrive-arrive headway (*AA*) and arrive-depart headway (*AD*) – are considered in the model (see Fig. 6-7, modified from [PT1, 2007]). The definitions of different headway types, as well as the calculation methods of minimum line headways, are given in [PACHL, 2002].

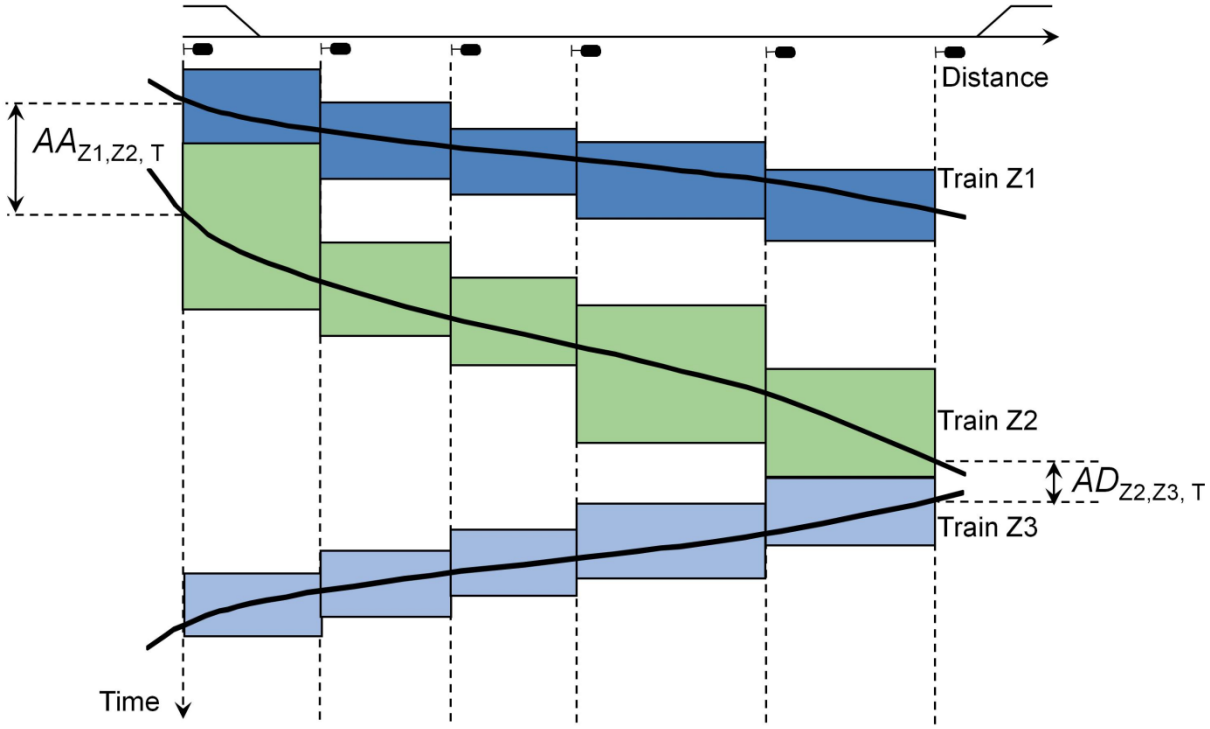



**Fig. 6-7 Minimum Line Headway in Open Track Section T**

The conflicts avoidance based on line headways and other principles of the proposed macroscopic optimization will be formulated in Section 6.2.3. At the end of this section, all these principles are summarized and highlighted as follows:

- The objective of the model is to minimize the weighted overall trip time.
- Adjustment of train sequences is the core of the model by introducing additional waiting time or overtaking.

- Train sequences on open track sections are the most determinant decision variables.
- Unscheduled waiting time is estimated as assistant decision variables. They are modeled only inside nodes.
- Minimum line headways are used in order to avoid conflicts on open track sections.

### 6.2.3 The Optimization Model for Macroscopic Dispatching

Similar to MARTIN's model, the optimization model on a macroscopic level consists of three parts:

- Objective Function
- Recursive computation for departing/passing time
- Conflict-free constraints

The objective function is formulated to minimize the weighted overall trip time.

$$\sum_{j=1}^{nges} C_j \left[ \sum_{i=1}^{iges} (TW_{i,j} + TI_{i,j}) \right] \rightarrow \min \tag{6-2}$$

Notations used (the notations are kept to be identical or similar to those in Section 2.2.2):

| | |
|---|---|
| $i$ | index of the knot |
| $j$ | index of the train |
| $C_j$ | train related constant for weighting of trip time for train $Z_j$ |
| $TI_{i,j}$ | scheduled operation time for train $Z_j$ in knot $i$ |
| $TW_{i,j}$ | unscheduled waiting time caused by obstructions for train $Z_j$ in knot $i$ |
| $iges$ | total number of knots for a given macro-path |
| $nges$ | total number of trains |

In the objective function, the value of train related weight can be assigned according to the value of class-oriented indicators (see Section 5.1.2). As explained in Section 6.2.2, unscheduled waiting times are only modeled inside nodes, and the value of $TW_{i,j}$ for open track sections are simplified as 0 (see equation 6-5). The scheduled operation time $TI_{i,j}$ includes both scheduled running time and scheduled dwell time. The values are derived from the result of initial simulation, with consideration of time-related dispatching (see equation 6.1).

Recursive computation for the departing/passing time on a knot will be considered for open track sections and nodes respectively.

In an open track section

$$TB_{i,j} = TB_{i-1,j} + TI_{i,j} \tag{6-3}$$

In a node

$$TB_{i,j} = TB_{i-1,j} + TI_{i,j} + TW_{i,j} \tag{6-4}$$

Notations used:

$TB_{i,j}$ departing/passing time for train $Z_j$ in knot $i$

Specially:

$$TW_{i,j} = 0 \mid (\text{knot } i \text{ is an open track section}) \tag{6-5}$$

The conflict-free conditions are only considered in open track sections. Considering an open track section T, for the successive movements between two trains $Z_m$ and $Z_n$, the arriving time of the second train should not be earlier than the arriving time of the first train plus the minimal arrive-arrive headway of T. The arriving time of the open track section equals the departing/passing time of the last knot. Hence, the constraints to regulate the departing/passing time at the node before open track section T are:

$$TB_{i_n-1,n} \geq TB_{i_m-1,m} + HVOR_{m,n,T} \cdot AA_{m,n,T} - (1 - HVOR_{m,n,T}) \cdot INF \tag{6-6}$$
$$TB_{i_m-1,m} \geq TB_{i_n-1,n} + (1 - HVOR_{m,n,T}) \cdot AA_{n,m,T} - HVOR_{m,n,T} \cdot INF$$

For an open track section T with bidirectional operations, the arriving time of the second train should not earlier than the departing time of the first train with opposite movements plus the minimal arrive-depart headway, and the conflict-free constraints to regulate the opposite movements are:

$$TB_{i_n-1,n} \geq TB_{i_m,m} + HVOR_{m,n,T} \cdot AD_{m,n,T} - (1 - HVOR_{m,n,T}) \cdot INF \tag{6-7}$$
$$TB_{i_m-1,m} \geq TB_{i_n,n} + (1 - HVOR_{m,n,T}) \cdot AD_{n,m,T} - HVOR_{m,n,T} \cdot INF$$

Notations used:

$AA_{m,n,T}$ minimum arrive-arrive headway when train $Z_n$ is after train $Z_m$ passing T with successive movements

$AD_{m,n,T}$ minimum arrive-depart headway when train $Z_n$ is after train $Z_m$ passing T with opposite movements

$i_m$ knot index of the open track section T for train $Z_m$

$i_n$ knot index of the open track section T for train $Z_n$

$HVOR_{m,n,\mathrm{T}}$ train sequence decision variable for the open train section T. If the value equals 1, train $\mathrm{Z}_m$ takes priority over train $\mathrm{Z}_n$

To avoid inconsistencies, the decision variables for train sequences are defined as follows:

$$HVOR_{m,n,\mathrm{T}}|(m<n)=\begin{cases}1, & \text{train } \mathrm{Z}_m \text{ has a higher priority than train } \mathrm{Z}_n\\ 0, & \text{else}\end{cases} \quad (6\text{-}8)$$

$$HVOR_{m,n,\mathrm{T}}|(m>n)=1-HVOR_{n,m,\mathrm{T}}$$

By this definition, the number of variables is also reduced. For each open track section with $n$ $(n \geq 2)$ trains, the maximum number of priority decision variables is $n \cdot (n-1)/2$.

Further attention should be paid to maintain the integrity of train sequence variables. Consider the situation shown in Fig. 6-8.

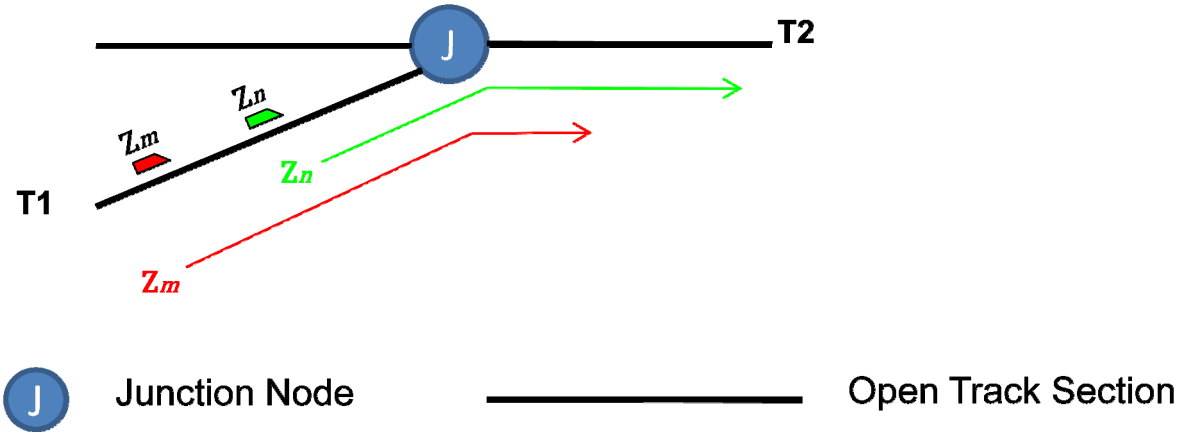



**Fig. 6-8 Train Sequence Integrity for Two Open Track Sections**

The open track sections T1 and T2 are connected via a junction node J. Train $\mathrm{Z}_m$ is after train $\mathrm{Z}_n$ passing T1 with successive movements, with the same macro-path from T1 to T2 via J. Since J is a junction node and overtaking is not allowed, the train order of $\mathrm{Z}_m$ and $\mathrm{Z}_n$ in J cannot be changed. It means that the train $\mathrm{Z}_m$ should also be after train $\mathrm{Z}_n$ Passing T2.

A part of macro-path is called partial macro-path. For two trains have a common partial macro-path with the form of “open track section – node – open track section”, when overtaking between two trains is not possible in the common node, the node may be a junction node, or overtaking route is not available in the (loop) node. The train sequence decision variable between these two trains for the next open track section should equal that of the previous open track section. For the example shown in Fig. 6-3, there is:

$$HVOR_{m,n,\mathrm{T1}} = HVOR_{m,n,\mathrm{T2}}$$

With respect to the integrity of train sequences, consistency is ensured, accompanied with the side effects due to the increased number of constraints. To reduce the additional constraints, a common priority decision variable can be used for the common partial macro-path. The tradeoff between less decision variables and more processes to identify common partial paths should be assessed in design time. In any case, the cost for ensuring consistency is very high when linear programming is used, since every possible change of train sequences is checked. The consistency of train sequences should also be ensured when other optimization techniques are utilized. A detailed discussion of maintaining the priority consistency with tabu search can be seen in Section 6.2.4.2.

At last, the deadlock problems should be taken into account. As the model of macroscopic optimization is linear, the optimal solution can be resolved by linear programming when the computational complexity is acceptable. In this case, deadlocks can be avoided by using the similar constraints applied in MARTIN's model as follows:

$$\sum_{i=1}^{iges} TW_{i,j} < 24h \tag{6-9}$$

It is not necessary to apply the constraint defined in equation 6-9 when linear programming is not adopted. With other optimization techniques, such as tabu search proposed in this dissertation, deadlocks can be prevented by setting train sequence variables with predefined rules. A detailed discussion will be given in Section 6.2.4.2.

### 6.2.4 Optimization with Tabu Search for Macroscopic Dispatching

Although the proposed macroscopic optimization model can be solved by linear programming, the computation capability is limited depending on the scope and traffic volume of the observed network. Heuristic method balances the quality of the solution and the computational complexity. It can be employed to find a relative optimal solution for a large scale optimization problem. Based on local search techniques, tabu search is one of the methods that is especially suitable to optimize the result starting from a basic solution.

#### 6.2.4.1 Introduction of Tabu Search

Tabu search (abbreviated as TS) was firstly introduced by GLVOER in 1986. It is an extension of traditional local search technique. The definition of tabu search is given in [GLOVER und LAGUNA, 1998] as follows:

"Tabu search is a meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality. … One of the main components of TS is its use of adaptive memory, which creates a more flexible search behavior. Memory-based strategies are therefore the hallmark of tabu search approaches."

Tabu search follows the same process as normal local search (neighborhood search) approaches. The basic procedure in a local search is to find a set of new solutions (*neighbors*) from an initial feasible solution, and a new neighbor is selected to start a new round of search until certain termination conditions are satisfied. The best solution of all the iterations is found from all the historical results.

The process can also be expressed with mathematic notations. The solution space is denoted as $X$. For a basic solution $x \in X$, the associated neighborhood $N(x) \subset X$ includes all the neighbors $x'$, and each $x'$ is reached from $x$ by a *move* operation (for the move operation in macroscopic dispatching see Section 6.2.4.2). A new $x'$ is selected to replace $x$ for a new search, and the process is executed repeatedly. Because local search techniques are always used for large scale optimization problems, not all possible $x'$ are able to be enumerated for selecting a new $x$, and the candidate list strategies should be considered to narrow the examination of $N(x)$. For macroscopic optimized dispatching, the strategy (see Section 6.2.4.2) can be set to guide a meaningful search, associated with a certain optimization objective. The procedure for calculating objective value that is used for selection a best new $x$ will be described in 6.2.4.3.

A highlighted feature of tabu search is the capability to avoid local optimality that leads the searching processes to being trapped in a local optimal solution. To depart from a local optimal solution, adaptive memory structures are used. Once a certain solution has been visited, the value of the solution or some characteristics of the solution (*tabu search attributes*) will be recorded and marked as "tabu", and the solutions fit the tabu criteria in further searches are excluded from the neighborhood for

next iteration. Therefore, the algorithm can avoid falling into a local optimal solution repeatedly. The tabu memory structures are updated dynamically in every new round of iterations, and the tabu rules are also evolved respectively. This fits exactly the origin meaning of “tabu” that is always conceived as “a social memory which is subject to modification over time” (see [GLOVER und LAGUNA, 1998]).

Two important principles, intensification and diversification, should be considered in the tabu search algorithm. Intensification encourages that searches are carried out toward the direction with good solution quality in history. Examinations on the solution with high quality (elite solutions) are concerned for intensification purpose, which is often implemented in a short term memory (STM) structure. Diversification encourages widening searches in unvisited areas in order to find potential global optimal solutions. Long term memory (LTM) structure is used for this purpose. The design of STM and LTM in a macroscopic dispatching will be explained in Section 6.2.4.4 and 6.2.4.5.

The framework of the tabu search is shown in Fig. 6-9. Only basic principles of tabu search used in the proposed dispatching model are introduced here. As a solution-based meta-heuristic algorithm, tabu search can be used to find a better result from a basic solution. It fits the structure of the optimized macroscopic dispatching proposed in this chapter. From Section 6.2.4.2 to 6.2.4.5, the search algorithm and memory structures of the model are designed and discussed in details.

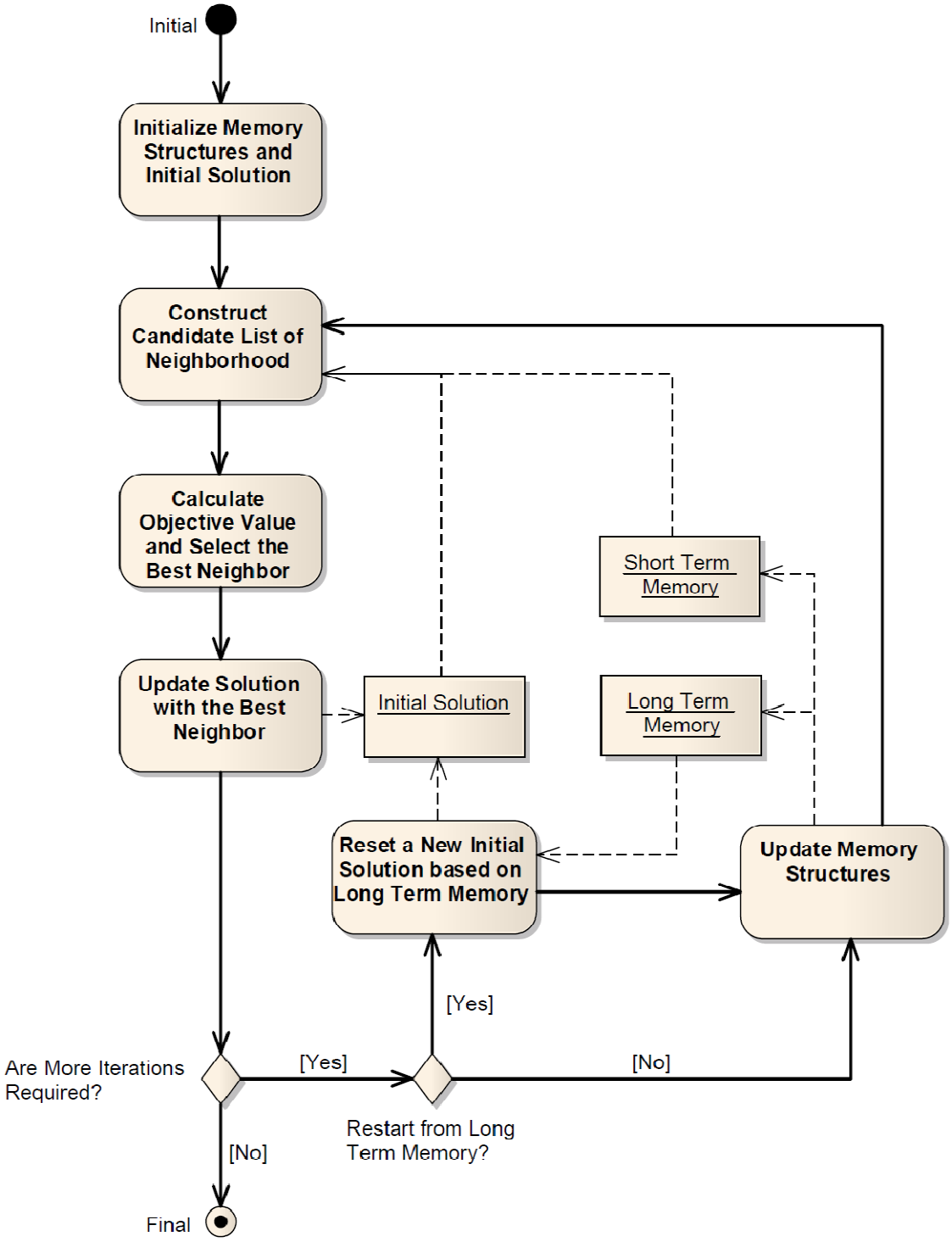



**Fig. 6-9 The Framework of Tabu Search**

### 6.2.4.2 Construction of Candidate List and Tabu Search Attributes

The construction of a candidate list is the starting point in a tabu searching iteration. The key aspects of this procedure are to define move operations and candidate list strategies.

The optimization process starts from a basic dispatching solution, and each bor $x'$ is generated by a move operation. For macroscopic dispatching, a move operation can be defined as an exchange of positions of the sequence in an open track section between a train and its immediately previous train.

The immediately previous train is defined as follows: for an open track section T with more than one train passing, if a train $Z_j$ is not the first train passing T, the immediately previous train $Z_{prev}$ of $Z_j$ is the train passed T before $Z_j$, and there is no other train passing T between $Z_{prev}$ and $Z_j$.

As described in Section 6.2.3, the integrity of train sequence variables should be maintained. For two trains with successive movements, a move operation is only allowed if the consistency of train sequences is ensured. In the example shown in Fig. 6-10, to maintain the priority consistency, move operations should be always executed starting from T1 and ending before L.

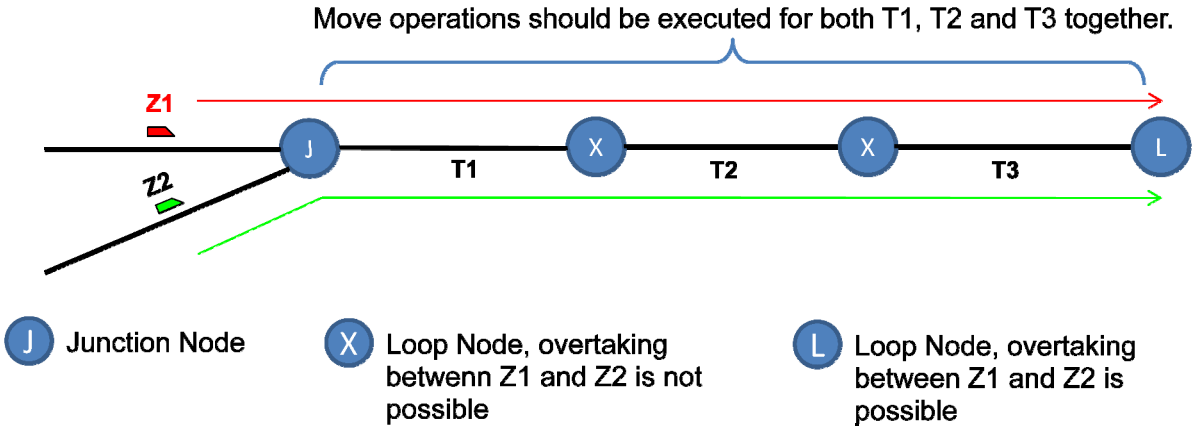


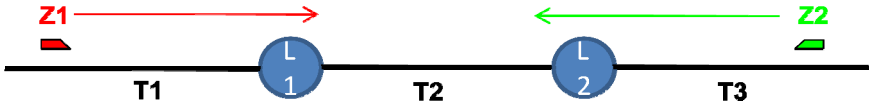

**Fig. 6-10 An Example of Move Operations for Successive Movements**

Therefore, for two trains with successive movements, possible move operations always start after a starting node. These operations are performed for all successive open track sections until the next loop node or the end of the common partial path is reached. Possible starting nodes can be:

- A loop node that allows overtaking
- A junction node, from which trains merge their macro-paths

In the lines with bidirectional operations, deadlocks may happen after a move operation was performed inappropriately. In contrast with deadlocks occurring in microscopic simulation (see Chapter 4), the deadlocks generated from a move operation are caused by violating the logic sequence on a common partial macro-path between two trains with opposite movements. An example is shown in Fig. 6-11. In the basic

dispatching solution, train Z1 passes the common partial macro-path from T1 to T3 before the opposite train Z2. After a move operation performed in open track section T2, the train movements are arranged incorrectly: after two opposite trains Z1 and Z2 met at loop node L1 (the sign of the meeting is that the train sequences are different in the connected open track sections T1 and T2), they meet again in L2! With the incorrect train sequence arrangement, the calculation of the objective value is not able to be completed (see Section 6.2.4.3). Hence, deadlock occurs after the invalid move operation. Train Z1 will not enter T2 since Z1 is designated to pass T2 after Z2; while Z2 will never be able to pass T2 since it has to wait Z1 passes T3.




| Train Sequence | T1 | T2 | T3 |
|---|---|---|---|
| Sequence (before move) | Z1, Z2 | Z1, Z2 | Z1, Z2 |
| Sequence (after move) | Z1, Z2 | **Z2, Z1 (Invalid!)** | Z1, Z2 |

**Fig. 6-11 An Invalid Move Operation for Opposite Movements**

To avoid deadlocks generated from a move operation, the decision variables for regulating train sequences should be checked carefully with the following rules: if two trains are running in a common partial macro-path with opposite running directions, only a move operation is possible that has at most one passing node in the common partial macro-path. A passing node is a loop node where a train is allowed to pass another opposite train. The definition of passing node can be expressed with priority decision variables. For the passing node $\mathrm{K}_i$ and the two connected open tracks $\mathrm{K}_{i-1}$ and $\mathrm{K}_{i+1}$, there must be:

$$HVOR_{m,n,\mathrm{K}_{i-1}} \neq HVOR_{m,n,\mathrm{K}_{i+1}}$$

If two opposite trains meet twice in a common partial path, a “circular wait” situation happens with incorrect precedence logic. Under the rule, deadlocks are avoided against the circular wait condition by controlling the number of passing nodes in a partial path.

After a move operation, the solution is updated with new train sequences, and a new candidate is generated. The next task is to identify the tabu search attributes of the new solution, which will be recorded in memory structures. There are several approaches to define tabu search attributes (see [GLOVER und LAGUNA, 1998]), and

the train-priority related attributes are concerned in the proposed model. It is convenient to use train sequence indexes instead of binary decision variables as tabu search attributes. Hence a move operation can be expressed by train sequence indexes (train sequence indexes are derived from the binary train sequence variables).

Denote attribute $Seq_{j,\mathrm{T}}$ as the position of train $\mathrm{Z}_j$ in the train sequence of open track section T. For two train $\mathrm{Z}_m$ and $\mathrm{Z}_n$ passing through T, and $Seq_{n,\mathrm{T}} = Seq_{m,\mathrm{T}} + 1$, the new sequences for train $\mathrm{Z}_m$ and $\mathrm{Z}_n$ after a move operation are denoted as $Seq'_{m,\mathrm{T}}$ and $Seq'_{n,\mathrm{T}}$, and

$$Seq'_{m,\mathrm{T}} = Seq_{n,\mathrm{T}} \tag{6-10}$$
$$Seq'_{n,\mathrm{T}} = Seq_{m,\mathrm{T}}$$

The candidate list strategies are defined for searching neighbors in the direction to which the solutions quality improves, which depends on the objective function of the model. To minimize overall weighted trip time, the potential for reducing the objective value is considered, and two kinds of trains are selected for generating candidate solutions:

- The train with a high weighted unscheduled waiting time
- The train with a high number of continuous unscheduled stops

If the overall weighted unscheduled waiting time of a train is high, improving the priority of the train can reduce its weighted trip time and possibly gain a lower objective value. Several move operations can be arranged after each unscheduled stop for that train separately, and a group of new solutions can be generated respectively.

A series of continuous unscheduled stops indicates that a faster train is hindered by one or more slower train. It may optimize the overall trip time if the faster train is arranged to take precedence over the previous slower train(s). In this case, only one move operation is executed to generate a new solution.

According to the strategies, the train and its immediately previous train are selected and move operations are performed. A simple implementation is to choose the train with the highest selection value (weighted unscheduled waiting time or number of unscheduled stops) and execute the corresponding move operation. If a move operation is tabu (by short term memory) at the current iteration, the train with the next highest value is selected. The new solution is added into the candidate list respectively and a modified neighborhood $N^*(x)$ is generated. After calculating the objective

value for each neighbor in $N^*(x)$ (see Section 6.2.4.3), the solution with the lowest objective value will be selected and replace the current solution $x$ for next iteration.
It can be adopted to select candidates with configurable parameters, e.g. choose the top $n$ $(n > 1)$ trains fitting the selection criteria for move operations. However, it is not possible to try all of the potential improvements for one solution $x$, the performance of the algorithm and the solution quality should be balanced in the implementation.

#### 6.2.4.3 Calculation of Objective Value

After a train is selected according to the candidate list strategies, a move operation can be executed between the train and its immediately previous train. Thus, the train sequence of one or more open track sections are updated in the new generated solution. The objective value of the new solution (the weighted overall trip time for all trains) will be calculated for evaluating the solution quality. The trip time of a train consists of two parts: scheduled operation time and unscheduled waiting time. The scheduled operation time will be treated as constant value. When an overtaking is required in a loop node, the scheduled operation time should be updated for the new overtaking route. For the rest of the knots, the scheduled operation time is kept the same as in the basic dispatching solution. For the new solution, the unscheduled waiting time is required to be recalculated, and all $TW_{i,j}$ (the unscheduled waiting time for train $\mathrm{Z}_j$ in knot $i$) are initialized as 0.

Essentially, the calculation procedure is a recursive process, and the departing time at each knot is a very important instrumental variable during the calculation. At the beginning of the calculation, the departing time of a knot for train $\mathrm{Z}_j$ at knot $i$ is initialized as:

$$TB_{i,j} = \begin{cases} null, & i > 0 \\ the\ entry\ or\ start\ time\ of\ train\ \mathrm{Z}_j + TI_{i,j}, & i = 0 \end{cases}$$

When the departing time of a knot has not calculated yet, the value will be set as $null$. A macro-path is called a *non-completed macro-path*, if the macro-path has at least one knot with not being determined departing time. A macro-path, in which all departing time of its knots have been calculated, is called a *completed macro-path*.
By completing the departing time for all the non-completed macro-paths, the objective value can be calculated eventually. For a non-completed macro-path, the derivation of departing time starts from the first undetermined knot, and the rest of them will be calculated based on the departing time of the previous knot, the scheduled opera-

tion time and the departing time of the immediately previous train (if available) of the concerned knot.

In some knots, the calculation of the departing time can be simply achieved by adding the scheduled operation time to the departing time of the previous knot. In this case, the knot is called a *non-obstruction knot* for that train, where an unscheduled stop does not take place in the knot. Some non-obstruction knots of a macro-path can be identified as follows:

- An open track section (all the unscheduled waiting has been simplified to nodes, see Section 6.2.2)
- A knot at the end of the macro-path
- A node that is not the end of the macro-path, and the concerned train is the first train passing the open track section next to the node

For the knots other than non-obstruction knots, the possible conflicts should be considered in the departing-time calculation. The minimum line headway between the train and its immediately previous train is utilized, and the departing time of the immediately previous train is required to be obtained in advance. The process is executed repeatedly until all the macro-paths are completed macro-paths. The calculation of object value is listed below and illustrated in Fig. 6-13:

1) Set the updated flag as false.
2) If not all the non-completed macro-path has been checked, take a train $\mathrm{Z}_j$ with a non-completed macro-path; else go to 9).
3) Find the first knot $\mathrm{K}_{i0}$ in the macro-path, where $TB_{i0,j}$ is $null$ and $TB_{i0-1,j}$ is not $null$.
4) If $\mathrm{K}_{i0}$ is a non-obstruction knot, $TB_{i0,j}$ can be calculated by:

$$TB_{i0,j} = TB_{i0-1,j} + TI_{i0,j}$$

5) If $\mathrm{K}_{i0}$ is not a non-obstruction knot, for the open track section $\mathrm{K}_{next}$ next to $\mathrm{K}_{i0}$ in the macro-path of train $\mathrm{Z}_j$, there is an immediately previous train $\mathrm{Z}_{prev}$ of $\mathrm{Z}_j$ (see Fig. 6-12):

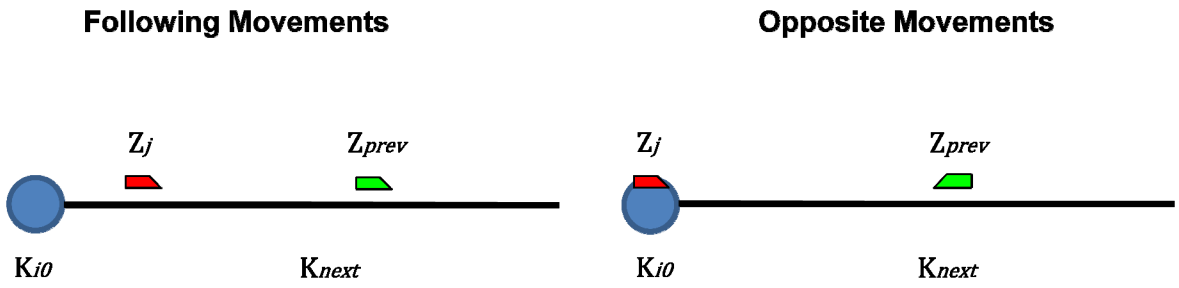



**Fig. 6-12 Immediately Previous Train in the next Open Track Section**

If $TB_{i[next]-1,prev}$ is not $null$ ($i[next]$ is the knot index of $\mathrm{K}_{next}$ in the macro-path of train $\mathrm{Z}_{prev}$, and $TB_{i[next]-1,prev}$ is the arriving time of train $\mathrm{Z}_{prev}$ at open track section $\mathrm{K}_{next}$),

$$TB_{i0,j} = \max(TB_{i0-1,j} + TI_{i0,j}, TB_{i[next],prev})$$

Here $TB_{i[next],prev}$ is the departing time of train $\mathrm{Z}_{prev}$ at open track section $\mathrm{K}_{next}$, and

$$TB_{i[next],prev} = \begin{cases} TB_{i[next]-1,prev} + AA_{prev,j,\mathrm{K}_{next}} & \text{following movements} \\ TB_{i[next]-1,prev} + TI_{i[next],prev} + AD_{prev,j,\mathrm{K}_{next}} & \text{opposite movements} \end{cases}$$

6) If $TB_{i0,prev}$ is $null$, set $TB_{i0,j}$ as $null$ and go to 2).

7) Set updated flag as true, and $TW_{i0,j}$ can be calculated:

$$TW_{i0,j} = TB_{i0,j} - TB_{i0-1,j} - TI_{i0,j}$$

8) If $\mathrm{K}_{i0}$ is not the end knot of the macro-path, assign the next macro-path knot to $\mathrm{K}_{i0}$, and set $i0 = i0 + 1$, then go to 4); else go to 2)

9) If updated flag is false, the calculation failed due to deadlock!

10) If all the macro-paths are completed macro-paths, go to 11); else go to 1).

11) Calculate the objective value:

$$\sum_{j=1}^{nges} C_j \left( \sum_{i=1}^{iges} TW_{i,j} + TI_{i,j} \right)$$

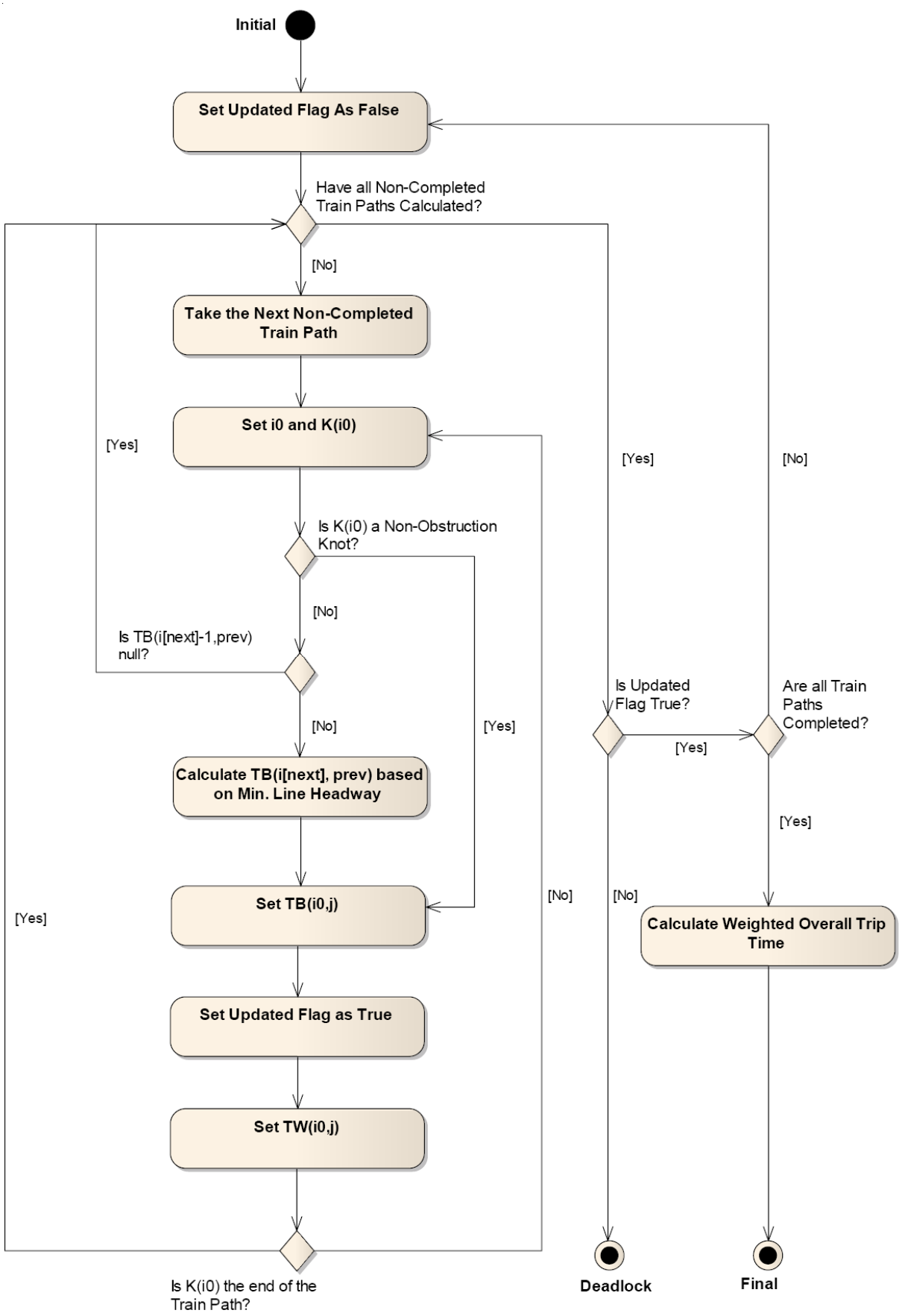



**Fig. 6-13 The Workflow for Calculating Objective Value**

With inappropriate setting of train sequences, the calculation of objective value may lead to deadlocks. The rules for deadlock prevention are given in 6.2.4.2. After the objective values of all the candidates have been calculated, the solution with the minimal value is selected as the base solution for the next round of iteration. During all the iterations, the best solution is recoded as the final result when the optimization process is terminated.

#### 6.2.4.4 Short Term Memory

The purpose of utilizing short term memory is to avoid reversal of moves that lead to local optimality. Recency-based memory is the most common implementation, by which the recent changed attributes and moves are tracked in order to prevent cycling moves. Depending on how to track changes and define tabu, several possible patterns for short term memory structures are utilized, including prevent tabu moves by values, by exchanges, by drop/add actions or by position of sequence. Implementation on the special cases and extensions of recency-based memory are presented in Section 3.5 of [GLOVER und LAGUNA, 1998].

In this dissertation, it is proposed to define tabu moves by identifying the move operation type and the position of train sequences in an open track section, which is categorized as sequencing tabu classification. Other possible patterns, such as by exchanges, are not considered.

The tabu-active status of a move operation depends on the type of a move operation. There are two possible move operations as shown in Fig. 6-14: to move (*delay*) a train to a later position or to move (*expedite*) a train to an earlier position (the terms “delay” and “expedite” are used in [GLOVER und LAGUNA, 1998]).

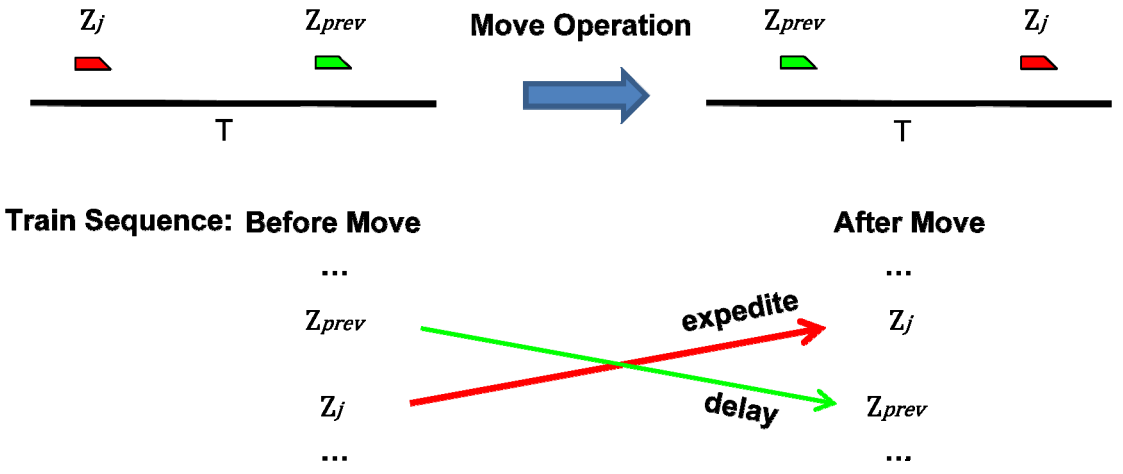



**Fig. 6-14 Move Operation Types**

*Tenure* parameters are defined for regulating number of iterations (time) when the tabu-active status of an attribute ends. Two tenure parameters are defined as follows:

$TabuDelayTenure$ number of iterations to prevent a train from being moved to a later position

$TabuExpediteTenure$ number of iterations to prevent a train from being moved to an earlier position

The tenure parameters can be set dynamically. A practical setting is to consider the current train position in the sequence of open track section T. For a train $\mathrm{Z}_j$ is moved to an earlier position $Seq_{j,\mathrm{T}}$, the $TabuDelayTenure$ can be defined as:

$$TabuDelayTenure = Seq_{j,\mathrm{T}} \tag{6-11}$$

For a train $\mathrm{Z}_j$ is swapped to a later position $Seq_{j,\mathrm{T}}$, the $TabuDelayTenure$ can be defined as:

$$TabuExpediteTenure = n - Seq_{j,\mathrm{T}} + 1$$

That means, if the associated unscheduled waiting time in T is still significant after changing the position for train $\mathrm{Z}_j$, the train is only allowed to be moved to the same direction as in the previous admissible move operation until the train reaches the "best" position (when the previous admissible move operation expedites the train) or the "worst" position (when the previous admissible move operation delays the train).

Two tabu arrays are maintained to record the delay and expedite moves in an open track section T respectively:

$TabuDelayStart(m, \mathrm{T})$ array to record the iteration number where train $\mathrm{Z}_m$ becomes tabu active, and train $\mathrm{Z}_m$ is not allowed to be swapped to a later position

$TabuExpediteStart(n, \mathrm{T})$ array to record the iteration number where train $\mathrm{Z}_n$ becomes tabu active, and train $\mathrm{Z}_n$ is not allowed to be swapped to an earlier position

When a move operation at open track section T is applied between a train $\mathrm{Z}_j$ and its immediately previous train $\mathrm{Z}_{prev}$ in iteration $Iter_0$ (the integer value $Iter_0$ is denoted as the iteration number of the admissible move operation), the tabu arrays are updated as:

$$TabuDelayStart(j, \mathrm{T}) = Iter_0 \tag{6-12}$$
$$TabuExpediteStart(prev, \mathrm{T}) = Iter_0$$

Move operations will be authorized with tabu arrays and tenure parameters. For a move operation between train $\mathrm{Z}_j$ and $\mathrm{Z}_{prev}$ at the open track section T in iteration $Iter$ (the integer value $Iter$ is denoted as the current iteration number), it will be prevented as a tabu move if:

$$Iter \leq TabuExpediteStart(j, \mathrm{T}) + TabuExpediteTenure \tag{6-13}$$

or

$$Iter \leq TabuDelayStart(prev, \mathrm{T}) + TabuDelayTenure$$

Therefore, a move operation can be prevented from falling into cycling moves with the proposed short term memory structure, which can be cooperated with the design of the long term memory structures.

#### 6.2.4.5 Long Term Memory

Long term memory is designed to complement short term memory structures according to the diversification strategies. Guided by the diversification strategies, a searching process is deviated from its previous searching trajectory intentionally in order to find a potential global optimal solution in other searching directions. Through inspecting the unvisited regions, the searching foundation is broadened, and local optimality is avoided by examining solutions that may not be found by original searching strategies.

An efficient approach to apply diversification strategies is to employ restarting mechanisms. The basic idea of the restarting mechanism is to restart a searching process from a new restarting point after certain rounds of (short term memory based) iterations. With simulative approaches, generating a new basic dispatching solution with alternative routes would be a good choice. Once a new restarting point is required, a new round of simulation is executed, where one or more selected alternative routes are adopted. Again, the route searching algorithm should comply with scheduled destinations (see Section 4.4.3). However, the costs for running several rounds of simulation might be high. In this case, it is worth to introduce the design of long term memory structures.

Several different techniques for building long term memory structures are discussed in [GLOVER und LAGUNA, 1998]. A diversification strategy, restarting mechanism with frequency-based memory structure (see Section 4.3.2, in [GLOVER und

LAGUNA, 1998]), is suitable for the proposed optimization model in macroscopic train dispatching. The new restarting point is generated from a biased construction procedure, based on overall frequency of elements occupying certain positions. The strategy was originally designed for solving a sequencing problem, which is exactly fitting the context of the proposed train dispatching model. By decreasing the train sequences for the trains with high priorities in previous solutions, a new diversified solution is generated.

For an open track section, the train sequence indexes are used as elements for recording the occupying frequency of the positions in varied train sequence lists. The original train sequence index can be derived from the departing time of the first basic dispatching solution (generated from the simulation). A higher value of departing time represents a later position in the sequence. The notation $Seq_{j,\mathrm{T}}^{O}$ is denoted as the original train sequence index of train $\mathrm{Z}_j$ in open train section T. When setting a new restarting point, the priority index $Seq_{j,\mathrm{T}}{}'$ should be updated as follows:

$$Seq_{j,\mathrm{T}}{}' = Seq_{j,\mathrm{T}}^{O} + d \cdot FrequencyMeasure \tag{6-14}$$

The frequency measure is assigned as percentage of time for a train running in the selected open track section T without unscheduled obstructions, considered from the last restarting point to the current iteration. The parameter $d$ is an empiric number. A simple example (modified from the example shown in [GLOVER und LAGUNA, 1998]) is shown in Table 6-2.

| Train | Initial Sequence Index | Frequency Measure | New Sequence Index |
|---|---|---|---|
| Z1 | 1 | 0.75 | 8.50 |
| Z2 | 2 | 0.40 | 6.00 |
| Z3 | 3 | 0.05 | 3.50 |
| Z4 | 4 | 0.30 | 7.50 |

**Table 6-2 An Example of Constructing a New Restarting Solution**

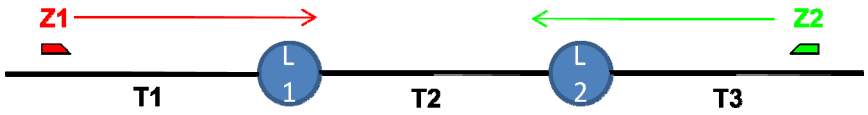
Taking the value $d$ as 10, the new generating sequence indexes will lead to a new sequence (Z3, Z2, Z4 and Z1) in the new restarting solution.

Special arrangement should be designed to select an open track section for generating new restarting solutions. The open track section with most frequent move operations in the last round of STM-based search will be chosen, from which the generated restarting solution will lead to an infrequently exploited searching direction. In addition, the last selected open track section will be recorded to avoid the same track

section being chosen in the next round (in case an open track section is selected again, the track section with second highest number of move operations will be chosen).

Deadlocks and priority consistency should be considered to ensure the validity of the new generated restarting solution. Both a move operation and a process of generating a new restarting solution involve arranging the train sequence at a selected open track section. Unlike the process of a move operation, which should respect to the existing train sequences in other open track sections, the train sequences of a new generated sequence in a selected open track section will dominate the sequences at the rest of the open track sections. After a new train sequence has been reset at a selected open track section T, further adjustments should be made between every two trains (denoted to $\mathrm{Z}_m$ and $\mathrm{Z}_n$, and $\mathrm{Z}_m$ takes precedence over $\mathrm{Z}_n$) at their common partial macro-path:

- For successive movements, train $\mathrm{Z}_n$ should follow $\mathrm{Z}_m$ before and after T, until the two trains reach the start or end of the common partial macro-path, or they arrive at a node where an overtaking is possible (same as the process in a move operation).
- For opposite movements, it is invalid if more than one meeting point at the common partial path appears after a new train sequence in T is set (see Section 6.2.4.2). Thus, the train sequences at the common partial path should be adjusted. One of the nodes connected to T will be taken as the only meeting point, or no meeting point exists at all. Taking the same example in Section 6.2.4.2, when a new train sequence that is invalid in T2 is set. Unlike in a move operation, the invalid sequence is not prevented any more. Instead one of the further adjustments (no meeting point, meeting point at L2, or meeting point at L1) will be arranged as shown in Fig. 6-15.




| Train Sequence | T1 | T2 | T3 |
|---|---|---|---|
| Original Solution (Invalid) | Z1, Z2 | **Z2, Z1** | Z1, Z2 |
| One of the following Adjustments will be applied (marked with bold font) | | | |
| No Meeting Point | **Z2, Z1** | Z2, Z1 | **Z2, Z1** |
| Meeting Point at L2 | **Z2, Z1** | Z2, Z1 | Z1, Z2 |
| Meeting Point at L1 | Z1, Z2 | Z2, Z1 | **Z2, Z1** |

**Fig. 6-15 Train Sequences Adjustments for Opposite Movements**

With the new calculated train sequence and the related adjustment, a biased solution is generated from the previous solutions. Based on the new solution, several new rounds of iterations are carried out with short term memory structures. It is possible that a solution can be searched in good quality without long term memory consideration. However, the implementation of long term memory broadens the possibility to find better solutions, especially if the solution quality is not improved in several iterations with using short term memory structures only.

## 6.3 Microscopic Timetable Elaboration

After the basic dispatching solution has been optimized on a macroscopic level, a detailed dispatching timetable based on the blocking time models can be elaborated on a microscopic level. The involved elements for generating an elaborated dispatching solution are: train routes, train sequences, scheduled operation time, and unscheduled waiting time (see Section 6.2.2).

In the optimized macroscopic solution, if additional overtaking operations are not required in a loop node, a train will take the same route in the loop node generated in the basic dispatching solution. For an overtaking operation that takes place in the optimized macroscopic solution, the overtaking train route can be established from the predefined operational route (in German: *Betriebsfahrweg*) inside a loop node. Therefore, all of the train routes can be determined before a microscopic elaboration starts.

Train sequences in all the open track sections have been set in the optimized macroscopic solution. Inside a node, train sequences can be determined according to the train sequences in adjacent open track sections.

If train $Z_m$ and $Z_n$ are merging at node J (see Fig. 6-16): the train sequences in J should be kept the same as that in the next open track section T2 ($HVOR_{m,n,\mathrm{J}} = HVOR_{m,n,\mathrm{T2}}$).

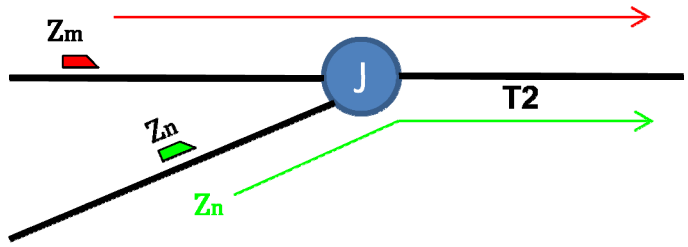



**Fig. 6-16 Train Sequences of a Juction Node for Merging**

If train $Z_m$ and $Z_n$ are filtering out at node J (see Fig. 6-17): the train sequences in J should be kept the same as that in the previous open track section T1 ($HVOR_{m,n,\mathrm{J}} = HVOR_{m,n,\mathrm{T1}}$).

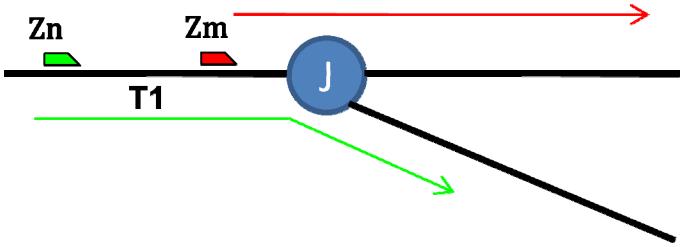



**Fig. 6-17 Train Sequences of Junction Node for Filtering Out**

If train $Z_m$ and $Z_n$ have a common macro-path from T1 via node N to T2 (see Fig. 6-18), and the train sequence variables at T1 and T2 are the same, then the sequences in N are also identical ($HVOR_{m,n,\mathrm{N}} = HVOR_{m,n,\mathrm{T1}} = HVOR_{m,n,\mathrm{T2}}$). This rule is also valid for opposite movements.

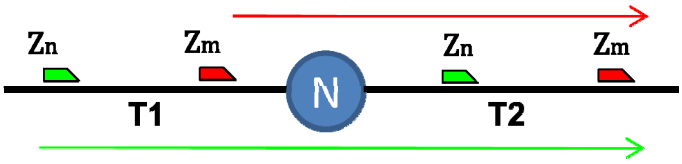



**Fig. 6-18 Without Changing of Train Sequences Before or After a Node**

If train $Z_m$ and $Z_n$ have a common train from T1 via node L to T2, and in node L an overtaking (for successive movements) or passing (for opposite movements) operation takes place, the train sequences of two conflicting routes at the entry or exit of the overtaking (or passing) route should be the same as the one in the connected open track section. For the example shown in Fig. 6-19, denote $HVOR_{\mathrm{A,B}}$ as the bi-

nary decision variable for two conflicting routes A and B, there are: $HVOR_{\mathrm{S1-S2,S1-S3}} = HVOR_{m,n,\mathrm{T1}}$ and $HVOR_{\mathrm{S2-S4,S3-S4}} = HVOR_{m,n,\mathrm{T2}}$. This rule is also valid for opposite movements.

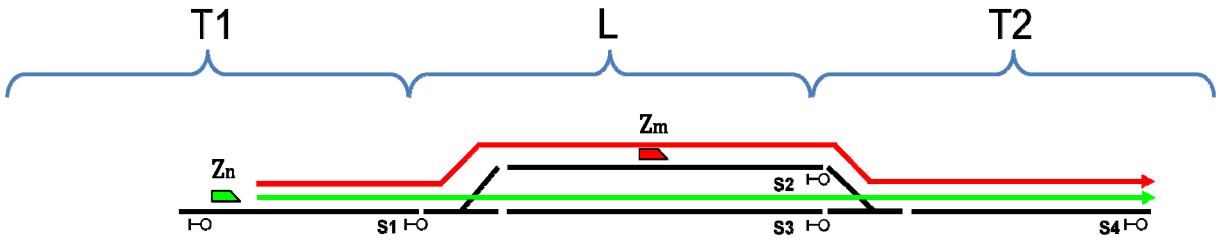



**Fig. 6-19 An Example of Changing Train Sequences**

After train routes, train sequences and the scheduled operation time (calculated by a running time calculation module) are determined, the last element required for generating an elaborated dispatching solution is unscheduled waiting time, which can be derived with analytical approaches or simulative approaches.

With analytical approaches, MARTIN's model (see Section 2.2.2) can be adopted, and the computational complexity problem will not exist anymore. As all the binary decision variables (for train routes and train sequences) have been determined already, the optimization problem has transferred from a mixed integer programming (MIP) problem to a pure linear programming (LP) problem that can be solved in polynomial time (see [HUANG und HAN, 2006]). The unscheduled waiting time and the departing/passing time at each block section are resolved in the linear programming model. Thus, a new dispatching timetable is obtained with one round of simulation (to generate a basic dispatching solution) and two rounds of optimization on macro- and microscopic levels. For more information about the optimization model and implementations refer to [MARTIN, 1995], [SCHLAICH, 2002], [HLAWENKA, 2003] and [CUI, 2005].

Alternatively, simulative approaches can be used for generating the elaborated dispatching solution. Asynchronous simulation is suggested at this phase, since it is faster than synchronous methods when train routes and train sequences are already determined (see the discussion in 2.2.1). About the theory and praxis of asynchronous simulation see [JACOBS, 2003], [DB NETZ AG, 405.0105 III] and [JACOBS, 2008].

A multi-level dispatching and optimization framework provides a feasible and optimized solution with acceptable computational costs. It should be stated again, that

the most sophisticated dispatching activities are still required to be fulfilled by dispatchers. An automatic dispatching system is good at fulfilling several routines and regular tasks involving much computational workloads with optimization capabilities. These tasks are difficult to be done manually in a short time. However, an automatic dispatching system only serves as an assistant tool, and the final decisions are always subjected to dispatchers.

# 7 Summary, Conclusion and Future Development

In this dissertation, a simulation-based hybrid model for a partially-automatic dispatching system was developed to support train dispatching also in a large scale control-territory with a lot of train movements.

The system productivity and performance of railway operation control can be improved with the assistance of a computer supported dispatching system, although only partially-automatic dispatching is possible in current practices. Three different types of automatic dispatching models, simulative, analytical, and heuristic, are prevailing in the field of automatic dispatching. Present practices show that no straightforward solution exists to solve dispatching problems in totality. Hybrid approaches are desirable to balance the system performance and solution quality among the different models. In cooperation with these models, several requirements with regard to system performance, solution optimization, and implementation costs are addressed and satisfied.

To provide a basic dispatching solution in real-time operation, simulative models are employed. Synchronous simulation is the basic form of simulation and is capable of integrating asynchronous features. Deadlocks are the most challenging problem of the synchronous simulation process, and therefore the Banker's algorithm is utilized for deadlock avoidance. Special improvement measures were helpful to improve the system performance and the efficiency of the Banker's algorithm. Train priorities influence dispatching decisions either implicitly or explicitly; in a simulative dispatching mode, the calculated train priority parameters are utilized in requesting infrastructure resources, allocating infrastructure resources or both.

With a basic dispatching solution, further optimization can be carried out on a macroscopic level. On this macroscopic level, the target is to determine optimized train sequences in open track sections. A Tabu search, as a solution-based metaheuristic algorithm, can be used when the computation amount is too large to be handled with a deterministic optimization model. The final elaboration is then executed within a linear programming model or an asynchronous model on a microscopic level. Lastly, within the framework of multi-level dispatching and optimization, a feasible and optimized dispatching solution can be developed by adopting the advantages of varied automatic dispatching models.

The conducted research for this dissertation shows:

- A synchronous simulation model is capable of generating basic dispatching solutions with low computational complexity.
- The Banker's algorithm is suitable for deadlock avoidance in railway synchronous simulation. It is able to deal with very complex network with complicated operations, such as in auxiliary stations for shunting processes.
- Improvement measures are important for the Banker's algorithm to improve system efficiency and reliability. The decision to implement the improvements is application-dependent. If the Banker's algorithm is applied to operations other than shunting operations, additional improvement measures would need to be investigated.
- Within the software PULRAN, the cooperation between synchronous simulation and asynchronous simulation increases the system flexibility and solution quality. In future development, it would also be possible to generate additional solutions based on one or more simulation modes or combinations thereof.

Further optimization can be executed within the framework of hybrid dispatching approaches that are proposed as a concrete solution to balance the system performance and solution quality. For future work on this topic, the following aspects are worth being considered:

- A Tabu search can be employed as a solution-based optimization technique. Further research on the algorithm and development of the implementation will improve the solution quality by optimizing dispatching solutions on a macroscopic level.
- The objective function is the most decisive part of an optimization model. By analyzing evaluation indicators and identifying invariants of the evaluation structure, a configurable dispatching system will be helpful to adapt possible changes of optimization objectives. To pursue system flexibility, an automatic dispatching system is highly dependent on the research in the fields of rule-based dispatching and evaluation methods.
- Optimization technology is evolving year by year. The design of the optimized dispatching system should incorporate a separation of the model definition from its implementation details as much as possible. It is important that the model itself is focused on the optimization objectives and not necessarily bound to one specific type of optimization technique.

An automatic dispatching system can only be developed progressively. System functions and requirements of an automatic system should be analyzed, reviewed, and defined iteratively based on the current theories, technologies, and practices. The dispatching measures can be categorized and planned in many waves of development. Essentially, each wave of development should focus on the individual automation of certain parts of the manual dispatching workloads that can be explicitly described, modeled, and optimized. The understanding of the status quo of railway operation control and software technology will be decisive to build a vision of a development plan. A mature automatic dispatching product should be built gradually from a series of pilot projects and be based on multi-disciplinary knowledge that has been accumulated over a long period of time. The art of railway operation control will continue to evolve with the development and integration of new automatic dispatching techniques.

# Literature References


[ALBRECHT, 2008] ALBRECHT, Thomas: Energy-Efficent Train Operation. In: Ingo Arne HANSEN, Jörn PACHL (eds.): *Railway Timetable & Traffic* Hamburg : Eurailpress, 2008. pp. 83-105

[AMIR, 2000] AMIR, Yair: *The Difference between Deadlock Prevention and Deadlock Avoidance* (10. October 2000). <http://www.cs.jhu.edu/~yairamir/cs418/os4/sld011.htm>, Accessed 20. August 2009.

[BELIK, 1987] BELIK, Ferenc: Deadlock Avoidance with a Modified Banker's Algorithm. In: *BIT Numerical Mathematics* 27 (1987), No. 3, pp. 290-305

[COFFMAN et al., 1971] COFFMAN, Edward G., ELPHICK, M. J. and SHOSHANI, Arie: System Deadlocks. In: *Computing Surveys* 3 (1971), No. 2, pp. 67-78

[CUI, 2005] CUI, Yong: *Implementation of the Optimization Theory for User Oriented Automatic Dispatching Systems in Railway Transport*. Universität Stuttgart, Master Thesis, 2005

[DB NETZ AG, 2007] DB NETZ AG: *Das Trassenpreissystem der DB Netz AG*. Frankfurt : Deutsche Bahn AG Marketingkommunikation (KMK), 2007

[DB NETZ AG, 405.0105 III] DB NETZ AG: Richtlinie 405.0105 III: *Fahrwegkapazität: Leistungsuntersuchungen mit Simulationsmethoden* (2006)

[DB NETZ AG, 420.0105] DB NETZ AG: Richtlinie 420.0105: *Dispositionsregeln* (2005)

[DIJKSTRA, 1959] DIJKSTRA, Edsger W.: A Note on Two Problems in Connexion with Graphs. In: *Numerische Mathematik* 1 (1959), No. 1, pp. 269-271

[DIJKSTRA, 1982] DIJKSTRA, Edsger W.: The Mathematics behind the Banker's Algorithm. New York : Springer-Verlag, 1982. pp. 308-312

[FEDERAL RAILWAY AUTHORITY, 2009] FEDERAL RAILWAY AUTHORITY: *English Website of EBA* (2009). <http://www.eba.bund.de/EN>, Accessed 20. August 2009.

[GLOVER and LAGUNA, 1998] GLOVER, Fred, LAGUNA, Manuel: *Tabu Search*. Norwell, Dordrecht : Kluwer Academic Publishers, 1998

[HLAWENKA, 2003] HLAWENKA, Alexander: *Entwurf und Implementierung eines Dispositionswerkzeugs zur Prozessoptimierung betriebsbedingter Wartezeiten im Schienengebundenen Verkehr*. Universität Stuttgart, Studienarbeit, 2003

[HUANG and HAN, 2006] HUANG, Hongxuan, HAN, Jiye: *Mathematic Programming*. Beijing : Tsinghua Publishing House, 2006

[IEV, 2005] IEV: *Projekt "ZLR Rastatt" Spezifikation der Arbeitspakete und Datenstrukturen*. Universität Stuttgart, 2005

[IEV, 2009] IEV: *PULRAN Spezifikation*. Universität Stuttgart, 2009

[JACOBS, 2003] JACOBS, Jürgen: *Rechnerunterstützte Konfliktermittlung und Entscheidungsunterstützung bei der Disposition des Zuglaufs*. Rheinisch-Westfälischen Technischen Hochschule Aachen, Dissertation, 2003

[JACOBS, 2008] JACOBS, Jürgen: Rescheduling. In: Ingo Arne HANSEN, Jörn PACHL (eds.): *Railway Timetable & Traffic* Hamburg : Eurailpress, 2008. pp. 182-191

[KROHN and LYNCH, 2009] KROHN, Teresa, LYNCH, Maureen B.: *Wissenschaftliches Kolloquium - KosiDispo* : Presentation Stuttgart, 28. May 2009

[KUMAR et al., 2008] KUMAR, Ajith K., HOUPT, Paul K., MATHE, Stephen S. et al.: *Multi-level Railway operation Optimization System and Method*. EP 1 697 196 B1. 2008.

[MARTIN, 1995] MARTIN, Ullrich: *Verfahren zur Bewertung von Zug- und Rangierfahrten bei der Disposition*. Technische Universität Braunschweig, Dissertation, 1995

[MARTIN, 2002] MARTIN, Ullrich: Deviation Management in Rail Operation. In: Wolfgang MÖHLENBRINK, Michael BARGENDE, Ulrich HANGLEITER and Ullrich MARTIN (eds.): *Networks for Mobility 2002, Proceeding of the International Symposium* Stuttgart : Universität Suttgart, 2002. pp. 368-377

[MARTIN, 2008] MARTIN, Ullrich: Performance Evaluation. In: Ingo Arne HANSEN, Jörn PACHL (eds.): *Railway Timetable & Traffic* Hamburg : Eurailpress, 2008. pp. 192-208

[MILLS and PUDNEY, 2003] MILLS, Graham, PUDNEY, Peter: The Effects of Deadlock Avoidance on Rail Network Capacity and Performance., 2003. pp. 49-63

[MÜCKE, 2002] MÜCKE, Wolfgang: *Operation control Systems in Public Transport*. Hamburg : Eurailpress, 2002

[OETTING, 2006] OETTING, Andreas: Train Guidance in Congested Networks. Networks for Mobility. In: Wolfgang MÖHLENBRINK, Ulrich HANGLEITER, Frank C. ENGLMANN et al. (eds.): *Networks for Mobility 2006, Proceeding of the 3rd International Symposium* Stuttgart : Universität Stuttgart, 2006. p. 28

[PACHL, 1993] PACHL, Jörn: *Steuerlogik für Zuglenkanlagen zum Einsatz unter stochastischen Betriebsbedingungen*. Technische Universität Braunschweig, Dissertation, 1993

[PACHL, 2002] PACHL, Jörn: *Railway Operation and Control*. Mountlake Terrace : VTD Rail Publishing, 2002

[PACHL, 2007] PACHL, Jörn: Avoiding Deadlocks in Synchronous Railway Simulations., 2007.

[PETERSEN and TAYLOR, 1983] PETERSEN, E. R., TAYLOR, A. J.: Line Block Prevention in Rail Line Dispatch. In: *INFOR Journal* 21 (1983), No. 1, pp. 46-51

[PT1, 2007] *Public Transportation and Railway Operation*. Universität Sutttgart, Vorlesung, Winter Semester 2007

[RADTKE, 2008] RADTKE, Alfons: Infrastructure Modelling. In: Ingo Arne HANSEN, Jörn PACHL (eds.): *Railway Timetable & Traffic* Hamburg : Eurailpress, 2008. pp. 43-57

[SCHLAICH, 2002] SCHLAICH, Johannes: *Beitrag zur Entwicklung eines Dispositionswerkzeugs zur Optimierung betriebsbedingter Wartezeiten im schienengebundenen Verkehr*. Universität Stuttgart, Studienarbeit, 2002

[SIEFER, 2008] SIEFER, Thomas: Simulation. In: Ingo Arne HANSEN, Jörn PACHL (eds.): *Railway Timetable & Traffic* Hamburg : Eurailpress, 2008. pp. 155-169

[SILBERSCHATZ et al., 2005] SILBERSCHATZ, Abraham, GALVIN, Peter B. and GAGNE, Greg: *Operating System Concepts*. Hoboken : John Wiley & Sons, Inc., 2005

[STANGE, 2007] STANGE, Hendrik: *Optimierte Anordnung der Infrastrukturelemente von Eisenbahnstandardstrecken*. Leibniz Universität Hannover, Dissertation, 2007

[THOMA, 2008] THOMA, Daniel: *Automatische Disposition im Bahnverkehr*. Universität Stuttgart, Studienarbeit, 2008

[TRITSCHLER et al., 2005] TRITSCHLER, Stefan, CUI, Yong and DOBESCHINSKY, Harry: Störfallmanagement im ÖPNV. In: *Der Nahverkehr* (2005), pp. 14-18

[WHITE, 2003] WHITE, Thomas: *Elements of Train Dispatching Vol. 1*. Mountlake Terrace : VTD Rail Publishing, 2003

# Keywords References

## N



## O



## P



## R



## S

## T



## U

# Abbreviations

| | |
|---|---|
| AA | Arrive-arrive headway |
| AD | Arrive-depart headway |
| AR | Currently available resources |
| ASDIS | Asynchronous Dispatching, in German: *Asynchrone Disposition* |
| Bd | Regional Dispatcher, in German: *Bereichsdisponent* |
| BS | Operational site, in German: *Betriebstelle* |
| BZ | Operation control center, in German: *Betriebszentrum* |
| CTC | Centralized traffic control |
| DRR | Dynamic route reservation |
| EBA | the German Federal Railway Authority, in German: *Eisenbahn-Bundesamt* |
| Fdl/özF | Train director/local operator, in German: *Fahrdienstleiter/ örtlich zuständiger Fdl* |
| FIFO | First in first out |
| FT | Feasible track |
| IEV | Institute of Railway and Transportation Engineering of the Universität Stuttgart, in German: *Institut für Eisenbahn- und Verkehrswesen der Universität Stuttgart* |
| JR | Junction type resource |
| KosiDispo | Consistent Dispatching in Planning and Operation, in German: *Konsistente Disposition in Planung und Betrieb* |
| LP | Linear programming |
| LTM | Long term memory |
| LZ | Locomotives or traction units, in German: *Lokzügen* |
| MCA | Movement consequence analysis |
| MIP | Mixed integer programming |
| MR | Maximal required resources |
| NJR | Non-junction type resource |
| NP | Non-deterministic polynomial |
| OCC | Operation control center |

| | |
|---|---|
| OR | Currently occupied resources |
| P | Performer set. $P_1$ includes the performer with a feasible task. $P_0$ includes the performer, of which the task has not yet been proven as a feasible task |
| Rail CRC | Cooperative Research Center for Railway Engineering and Technologies |
| RUDY | Regional Enterprise-spreading Dynamic Sampling of Timetable Information, Reservation and Operation in Public Transport, in German: *Regionale Unternehmensübergreifende Dynamisierung von Fahrplaninformation, Buchung und Betrieb im ÖPNV* |
| PULRAN | Program to Research the Logistic in the Shunting Operation, in German: *Programm zur Untersuchung der Logistik im Rangierdienst* |
| PULZURE | Program to Support a Partially-automatic Train Running Control, in German: *Programm zur Unterstützung einer Teilautomatischen Zuglaufregelung* |
| STM | Short term memory |
| TS | Tabu search |
| VzG | Permitted speed regulation with special given usage conditions, in German: *Verzeichnis örtlich zulässiger Geschwindigkeiten* |
| Zd/Zlr | Traffic controller, in German: *Zugdisponent/Zuglenker* |

# Appendix A Train Priorities Determination

### Calculation of Class-oriented Priority Value for a New Introduced Indicator

$$V_k' = \begin{cases} \dfrac{V_1 \cdot k}{C_i}, & 1 \le k < C_i \\ V_j, & k = j \cdot C_i \\ V_{j-1} + \left(V_j - V_{j-1}\right) \cdot \left[\dfrac{k}{C_i} - (j-1)\right], & (j-1) \cdot C_i < k < j \cdot C_i \text{ and } j > 1 \end{cases} \tag{5-1}$$

Notations used:

| | |
|---|---|
| $V_k'$ | priority value for new sub-classes $k$ |
| $V_j$ | priority value for class $j$ before the new indicator is introduced |
| $C_i$ | number of categories classified by the new introduced indicator |

### Calculation of Train Priority Value

$$P_{\mathrm{z}} = V_k + \alpha \cdot V_{\mathrm{z}} \tag{5-2}$$

Notations used:

| | |
|---|---|
| $P_{\mathrm{z}}$ | priority value for train Z |
| $V_k$ | class-oriented indicator value for the train class $k$ |
| $V_{\mathrm{z}}$ | train-oriented indicator value for train Z |
| $\alpha$ | an empirical constant value |

### The Possibility of Reduction of Delay

$$C_{\mathrm{RED}}[-] = \frac{\bar{V}_{\mathrm{zRest}}[\mathrm{km/h}] \cdot T_{\mathrm{RRest}}[\mathrm{h}]}{L_{\mathrm{Rest}}[\mathrm{km}]}\,, \qquad \text{with } L_{\mathrm{Rest}} \ge 1\mathrm{km} \tag{5-3}$$

Notations used:

| | |
|---|---|
| $C_{\mathrm{RED}}$ | value of the possibility of reduction from delays |
| $\bar{V}_{\mathrm{zRest}}$ | average speed in the rest of the route |
| $T_{\mathrm{RRest}}$ | sum of the scheduled recovery time in the rest of the route |
| $L_{\mathrm{Rest}}$ | length of the rest of the route |

**Modification of Requesting Distance**

$$D_{\mathrm{req}} = D_{\mathrm{min}} \cdot \left[1 + \frac{P_{\mathrm{z}}}{\max\,(V_k)}\right] \tag{5-4}$$

Notations used:

| | |
|---|---|
| $D_{\mathrm{req}}$ | requesting distance in the dynamic resource requesting mode |
| $D_{\mathrm{min}}$ | predicted minimum requesting distance in the time interval |
| $P_{\mathrm{z}}$ | priority value for train Z |
| $V_k$ | class-oriented indicator value for the train class $k$ |

# Appendix B Multi-Level Dispatching and Optimization

### The Adjusted Running Time through Recovery from Delay

$$t_{\text{adjusted}} = \max\left(t_{\text{s}} - t_{\text{d}},\ t_{\text{min}}\right) \tag{6-1}$$

Notations used:

| | |
|---|---|
| $t_{\text{adjusted}}$ | adjusted running time |
| $t_{\text{s}}$ | scheduled running time |
| $t_{\text{d}}$ | delay of the train compared with original timetable, must not be negative |
| $t_{\text{min}}$ | required minimal running time |

### The Objective Function of Macroscopic Optimization

$$\sum_{j=1}^{nges} C_j \left[\sum_{i=1}^{iges} (TW_{i,j} + TI_{i,j})\right] \rightarrow \min \tag{6-2}$$

Notations used:

| | |
|---|---|
| $i$ | index of the knot |
| $j$ | index of the train |
| $C_j$ | train related constant for weighting of trip time for train $\mathrm{Z}_j$ |
| $TI_{i,j}$ | scheduled operation time for train $\mathrm{Z}_j$ in knot $i$ |
| $TW_{i,j}$ | unscheduled waiting time caused by obstructions for train $\mathrm{Z}_j$ in knot $i$ |
| $iges$ | total number of knots for a given macro-path |
| $nges$ | total number of trains |

### Recursive Departing Time Calculation for Macroscopic Optimization

For an open track section

$$TB_{i,j} = TB_{i-1,j} + TI_{i,j} \tag{6-3}$$

For a node:

$$TB_{i,j} = TB_{i-1,j} + TI_{i,j} + TW_{i,j} \tag{6-4}$$

Specially:

$$TW_{i,j} = 0 \mid (\text{knot } i \text{ is an open track section}) \tag{6-5}$$

Notations used:

$TB_{i,j}$ departing/passing time for train $Z_j$ in knot $i$

$TW_{i,j}$ unscheduled waiting time for train $Z_j$ in knot $i$

**Conflict-free Constrains for Successive Movements (6-6), Opposite Movements (6-7), and Train Sequence Constrains (6-8)**

$$TB_{i_n-1,n} \geq TB_{i_m-1,m} + HVOR_{m,n,T} \cdot AA_{m,n,T} - (1 - HVOR_{m,n,T}) \cdot INF \quad (6\text{-}6)$$
$$TB_{i_m-1,m} \geq TB_{i_n-1,n} + (1 - HVOR_{m,n,T}) \cdot AA_{n,m,T} - HVOR_{m,n,T} \cdot INF$$

$$TB_{i_n-1,n} \geq TB_{i_m,m} + HVOR_{m,n,T} \cdot AD_{m,n,T} - (1 - HVOR_{m,n,T}) \cdot INF \quad (6\text{-}7)$$
$$TB_{i_m-1,m} \geq TB_{i_n,n} + (1 - HVOR_{m,n,T}) \cdot AD_{n,m,T} - HVOR_{m,n,T} \cdot INF$$

$$HVOR_{m,n,T}|(m < n) = \begin{cases} 1, & \text{train } Z_m \text{ has a higher priority than train } Z_n \\ 0, & \text{else} \end{cases} \quad (6\text{-}8)$$
$$HVOR_{m,n,T}|(m > n) = 1 - HVOR_{n,m,T}$$

Notations used:

$AA_{m,n,T}$ minimum arrive-arrive headway when train $Z_n$ is after train $Z_m$ passing T with successive movements

$AD_{m,n,T}$ minimum arrive-depart headway when train $Z_n$ is after train $Z_m$ passing T with opposite movements

$i_m$ knot index of the open track section T for train $Z_m$

$i_n$ knot index of the open track section T for train $Z_n$

$HVOR_{m,n,T}$ train sequence decision variable for the open train section T. If the value equals 1, train $Z_m$ takes priority over train $Z_n$

**Deadlock Avoidance with Linear programming**

$$\sum_{i=1}^{iges} TW_{i,j} < 24h \quad (6\text{-}9)$$

**A Move Operation in Tabu Search**

$$Seq'_{m,\mathrm{T}} = Seq_{n,\mathrm{T}} \qquad (6\text{-}10)$$
$$Seq'_{n,\mathrm{T}} = Seq_{m,\mathrm{T}}$$

Notations used:

$Seq_{j,L}$ position of train $\mathrm{Z}_j$ in the train sequence of open track section T

$Seq'_{j,L}$ new position of train $\mathrm{Z}_j$ after a move operation

Here, two train $\mathrm{Z}_m$ and $\mathrm{Z}_n$ passing through T, and $Seq_{n,\mathrm{T}} = Seq_{m,\mathrm{T}} + 1$

**Short Term Memory**

$$TabuDelayTenure = Seq_{j,\mathrm{T}} \qquad (6\text{-}11)$$
$$TabuExpediteTenure = n - Seq_{j,\mathrm{T}} + 1$$

$$TabuDelayStart(j,\mathrm{T}) = Iter_0 \qquad (6\text{-}12)$$
$$TabuExpediteStart(prev,\mathrm{T}) = Iter_0$$

$$Iter \leq TabuExpediteStart(j,\mathrm{T}) + TabuExpediteTenure \qquad (6\text{-}13)$$
$$Iter \leq TabuDelayStart(prev,\mathrm{T}) + TabuDelayTenure$$

Notations used:

$TabuDelayTenure$ number of iterations to prevent a train from being moved to a later position

$TabuExpediteTenure$ number of iterations to prevent a train from being moved to an earlier position

$TabuDelayStart(m,\mathrm{T})$ array to record the iteration number where train $\mathrm{Z}_m$ becomes tabu active, and train $\mathrm{Z}_m$ is not allowed to be swapped to a later position

$TabuExpediteStart(n,\mathrm{T})$ array to record the iteration number where train $\mathrm{Z}_n$ becomes tabu active, and train $\mathrm{Z}_n$ is not allowed to be swapped to an earlier position

**Long Term Memory**

$$Seq_{j,\mathrm{T}}{}' = Seq_{j,\mathrm{T}}^{O} + d \cdot FrequencyMeasure \tag{6-14}$$

Notations used:

| | |
|---|---|
| $FrequencyMeasure$ | percentage of time for a train $\mathrm{Z}_j$ running in the selected open track section T without unscheduled obstructions |
| $d$ | an empiric number |

# Appendix C An Example of Deadlock Test in PULRAN

The example demonstrates the process of deadlock tests in PULRAN within the given infrastructure and trains as shown in Fig. C-1. The tracks are named with the prefix "*Gleis*" (the word "track" in German), and the points are named with the prefix "*Weiche*" (the word "point" in German). All the train movements and track occupancy situation are logged as shown in Fig. C-2.

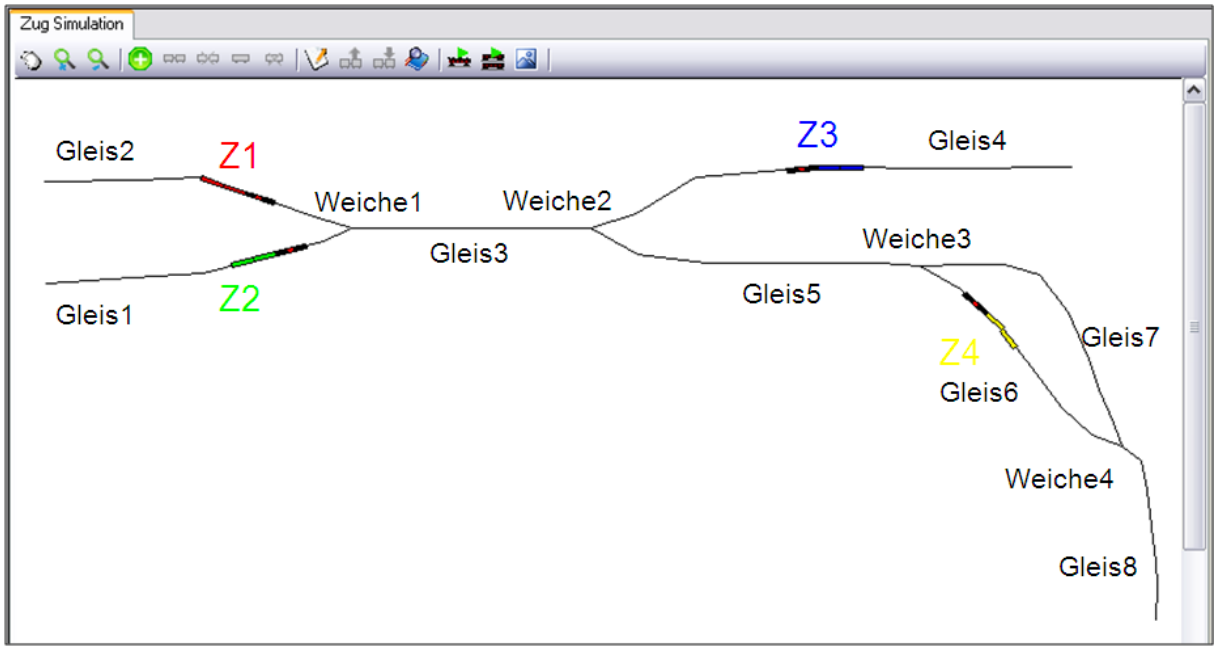



**Fig. C-1 The Initial State of Trains**

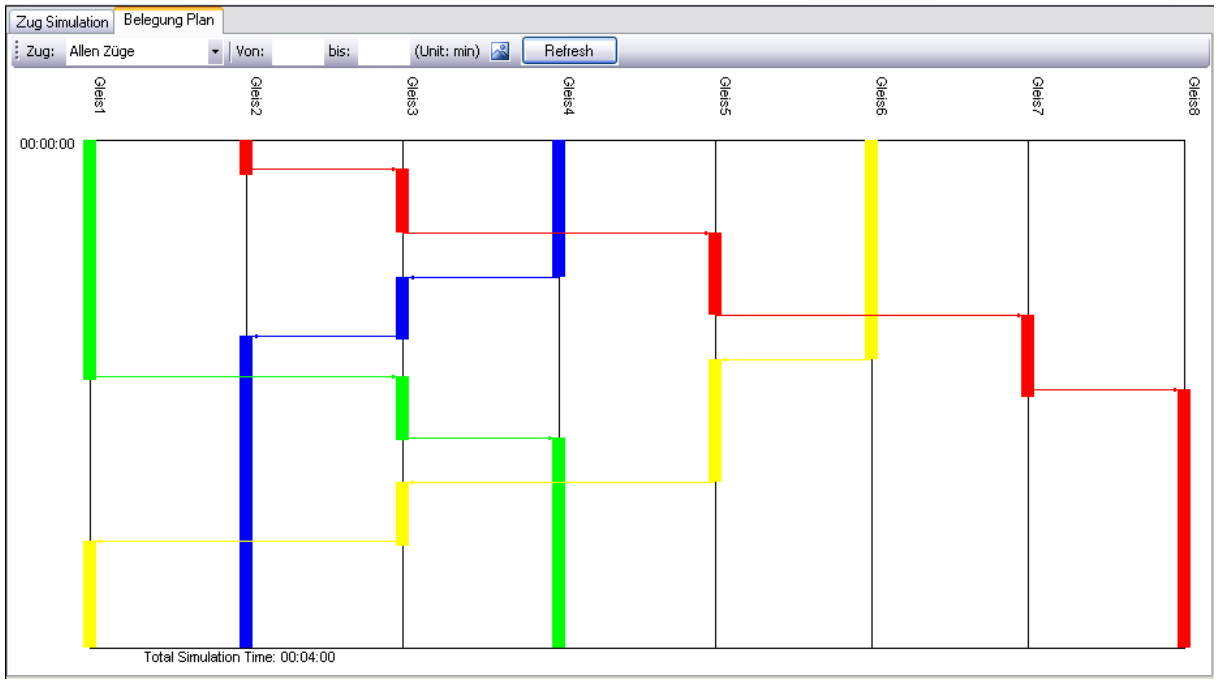



**Fig. C-2 The Track Occupancy Chart**

The process of deadlock tests is listed in Table C-1.

- Start Test Time: first time point that a deadlock test is executed
- Solved Time: time that a deadlock test is passed (without deadlock)
- Requester: requester of the deadlock test
- Resource: requested resource of the deadlock test
- Locked Resource: resource that in MR but not in AR or OR, which will be excluded from the next round of alternative route searching

| Start Test Time (sec.) | Solved Time (sec.) | Requester | Resource | Locked Resource |
|---|---|---|---|---|
| 1 | 94 | Z4 | Weiche3 | Gleis5 |
| 3 | 104 | Z2 | Weiche1 | Gleis3 |
| 7 | 7 | Z1 | Weiche1 | |
| 14 | 14 | Z1 | Gleis3 | |
| 19 | 55 | Z3 | Weiche2 | Gleis3 |
| 34 | 34 | Z1 | Weiche2 | |
| 44 | 44 | Z1 | Gleis5 | |
| 65 | 65 | Z3 | Gleis3 | |
| 73 | 73 | Z1 | Weiche3 | |
| 83 | 83 | Z1 | Gleis7 | |
| 85 | 85 | Z3 | Weiche1 | |
| 93 | 93 | Z3 | Gleis2 | |
| 104 | 104 | Z4 | Gleis5 | |
| 112 | 112 | Z1 | Weiche4 | |
| 112 | 112 | Z2 | Gleis3 | |
| 118 | 118 | Z1 | Gleis8 | |
| 132 | 132 | Z2 | Weiche2 | |
| 141 | 141 | Z2 | Gleis4 | |
| 152 | 152 | Z4 | Weiche2 | |
| 162 | 162 | Z4 | Gleis3 | |
| 182 | 182 | Z4 | Weiche1 | |
| 190 | 190 | Z4 | Gleis1 | |

**Table C-1 The Process of Deadlock Test**

# Appendix D Performance Statistic for PULRAN

In Table D-1, the running time of an execution in PULRAN is listed. Each row of the table records the duration of the running time, the number of resources, and the number of trains for 10 minutes simulation (with 4 seconds time interval for each single processing step). The average duration for 10 minutes simulation is 0.133 seconds. Inside the simulation, the improvements of potential state transition analysis and alternative routes searching are used for deadlock avoidance (see Section 4.5).

| Time (sec.) | Duration (ms.) | Num. of Resources | Num. of Trains |
|---|---|---|---|
| 0 | 31 | 125 | 1 |
| 600 | 109 | 125 | 1 |
| 1200 | 47 | 125 | 1 |
| 1800 | 78 | 125 | 1 |
| 2400 | 63 | 125 | 1 |
| 3000 | 47 | 125 | 1 |
| 3600 | 63 | 125 | 1 |
| 4200 | 406 | 125 | 4 |
| 4800 | 531 | 125 | 9 |
| 5400 | 344 | 125 | 6 |
| 6000 | 250 | 125 | 8 |
| 6600 | 156 | 125 | 7 |
| 7200 | 78 | 125 | 7 |
| 7800 | 78 | 125 | 7 |
| 8400 | 63 | 125 | 7 |
| 9000 | 109 | 125 | 8 |
| 9600 | 78 | 125 | 8 |
| 10200 | 156 | 125 | 7 |
| 10800 | 78 | 125 | 7 |
| 11400 | 78 | 125 | 7 |
| 12000 | 94 | 125 | 7 |
| 12600 | 141 | 125 | 10 |
| 13200 | 78 | 125 | 6 |
| 13800 | 78 | 125 | 6 |
| 14400 | 94 | 125 | 6 |
| 15000 | 78 | 125 | 6 |
| 15600 | 297 | 125 | 8 |
| 16200 | 422 | 125 | 7 |
| 16800 | 172 | 125 | 4 |
| 17400 | 63 | 125 | 4 |
| 18000 | 78 | 125 | 4 |

| Time (sec.) | Duration (ms.) | Num. of Resources | Num. of Trains |
|---|---|---|---|
| 43200 | 125 | 125 | 6 |
| 43800 | 219 | 125 | 4 |
| 44400 | 625 | 125 | 10 |
| 45000 | 703 | 125 | 9 |
| 45600 | 219 | 125 | 4 |
| 46200 | 125 | 125 | 4 |
| 46800 | 109 | 125 | 4 |
| 47400 | 266 | 125 | 7 |
| 48000 | 141 | 125 | 5 |
| 48600 | 109 | 125 | 5 |
| 49200 | 109 | 125 | 5 |
| 49800 | 94 | 125 | 5 |
| 50400 | 109 | 125 | 5 |
| 51000 | 109 | 125 | 5 |
| 51600 | 94 | 125 | 5 |
| 52200 | 109 | 125 | 5 |
| 52800 | 109 | 125 | 5 |
| 53400 | 219 | 125 | 6 |
| 54000 | 313 | 125 | 9 |
| 54600 | 188 | 125 | 3 |
| 55200 | 94 | 125 | 3 |
| 55800 | 94 | 125 | 3 |
| 56400 | 94 | 125 | 3 |
| 57000 | 109 | 125 | 3 |
| 57600 | 94 | 125 | 3 |
| 58200 | 109 | 125 | 3 |
| 58800 | 344 | 125 | 5 |
| 59400 | 219 | 125 | 2 |
| 60000 | 78 | 125 | 2 |
| 60600 | 94 | 125 | 2 |
| 61200 | 109 | 125 | 2 |

| Time (sec.) | Duration (ms.) | Num. of Resources | Num. of Trains |
|---|---|---|---|
| 18600 | 219 | 125 | 7 |
| 19200 | 266 | 125 | 7 |
| 19800 | 125 | 125 | 6 |
| 20400 | 78 | 125 | 6 |
| 21000 | 94 | 125 | 6 |
| 21600 | 78 | 125 | 6 |
| 22200 | 78 | 125 | 6 |
| 22800 | 78 | 125 | 6 |
| 23400 | 94 | 125 | 6 |
| 24000 | 94 | 125 | 6 |
| 24600 | 78 | 125 | 6 |
| 25200 | 94 | 125 | 6 |
| 25800 | 94 | 125 | 6 |
| 26400 | 94 | 125 | 6 |
| 27000 | 78 | 125 | 6 |
| 27600 | 94 | 125 | 6 |
| 28200 | 109 | 125 | 6 |
| 28800 | 94 | 125 | 6 |
| 29400 | 94 | 125 | 6 |
| 30000 | 328 | 125 | 8 |
| 30600 | 547 | 125 | 9 |
| 31200 | 188 | 125 | 2 |
| 31800 | 78 | 125 | 2 |
| 32400 | 63 | 125 | 2 |
| 33000 | 234 | 125 | 5 |
| 33600 | 281 | 125 | 5 |
| 34200 | 125 | 125 | 4 |
| 34800 | 78 | 125 | 4 |
| 35400 | 94 | 125 | 4 |
| 36000 | 94 | 125 | 4 |
| 36600 | 78 | 125 | 4 |
| 37200 | 94 | 125 | 4 |
| 37800 | 172 | 125 | 8 |
| 38400 | 125 | 125 | 6 |
| 39000 | 94 | 125 | 6 |
| 39600 | 94 | 125 | 6 |
| 40200 | 94 | 125 | 6 |
| 40800 | 125 | 125 | 6 |
| 41400 | 141 | 125 | 6 |
| 42000 | 125 | 125 | 6 |
| 42600 | 141 | 125 | 6 |
| 61800 | 94 | 125 | 2 |
| 62400 | 94 | 125 | 2 |
| 63000 | 94 | 125 | 2 |
| 63600 | 94 | 125 | 2 |
| 64200 | 94 | 125 | 2 |
| 64800 | 78 | 125 | 2 |
| 65400 | 94 | 125 | 2 |
| 66000 | 94 | 125 | 2 |
| 66600 | 94 | 125 | 2 |
| 67200 | 94 | 125 | 2 |
| 67800 | 94 | 125 | 2 |
| 68400 | 94 | 125 | 2 |
| 69000 | 78 | 125 | 2 |
| 69600 | 94 | 125 | 2 |
| 70200 | 94 | 125 | 2 |
| 70800 | 266 | 125 | 0 |
| 71400 | 78 | 125 | 0 |
| 72000 | 78 | 125 | 0 |
| 72600 | 94 | 125 | 0 |
| 73200 | 109 | 125 | 0 |
| 73800 | 78 | 125 | 0 |
| 74400 | 94 | 125 | 0 |
| 75000 | 78 | 125 | 0 |
| 75600 | 78 | 125 | 0 |
| 76200 | 94 | 125 | 0 |
| 76800 | 78 | 125 | 0 |
| 77400 | 94 | 125 | 0 |
| 78000 | 78 | 125 | 0 |
| 78600 | 109 | 125 | 0 |
| 79200 | 78 | 125 | 0 |
| 79800 | 78 | 125 | 0 |
| 80400 | 78 | 125 | 0 |
| 81000 | 94 | 125 | 0 |
| 81600 | 94 | 125 | 0 |
| 82200 | 94 | 125 | 0 |
| 82800 | 78 | 125 | 0 |
| 83400 | 78 | 125 | 0 |
| 84000 | 94 | 125 | 0 |
| 84600 | 94 | 125 | 0 |
| 85200 | 78 | 125 | 0 |
| 85800 | 78 | 125 | 0 |

**Table D-1 Performance Statistic of PULRAN**