Report No. K-TRAN:KU-95-7
Final Report

DEVELOPMENT OF PROJECT ACTIVITY DURATION AND RESOURCE REQUIREMENT ALGORITHMS BASED ON HISTORICAL DATA

W.M. Kim Roddis
Liye Zhang
University of Kansas
Lawrence, Kansas

April 1997

K-TRAN

A COOPERATIVE TRANSPORTATION RESEARCH PROGRAM BETWEEN:
KANSAS DEPARTMENT OF TRANSPORTATION
THE KANSAS STATE UNIVERSITY
THE UNIVERSITY OF KANSAS

1. Report No.	2. Government Accession No.	3. Recipient Catalog No.
K-TRAN: KU 95-7		

4. Title and Subtitle	5. Report Date
Development of Project Activity Duration and Resource Requirement Algorithms Based on Historical Data	Apr 1997
	6. Performing Org. Code
7. Author(s) W. M. Kim Roddis and Liye Zhang	8. Performing Org. Report No.
9. Performing Organization Name and Address University of Kansas Department of Civil & Environmental Engineering Lawrence, Kansas 66045	10. Work Unit No. (TRAIS) 11. Contract or Grant No. C-843
12. Sponsoring Agency Name and Address Kansas Department of Transportation Docking State Office Bldg. Topeka, Kansas 66612	13. Type of Report and Period Covered Final Report May 1995 to Apr 1997 14. Sponsoring Agency Code 106 RE-0065-01

15. Supplementary Notes

16. Abstract

Effective planning and scheduling has become increasingly important for the state departments of transportation to efficiently use resources and avoid project delays. Accurate estimates are needed for task durations and resource requirements. The predictive models for durations and resource requirements need to accurately reflect the agency's current business practices and requirements. Updating predictive models is thus important to the improvement of planning and scheduling.

Updating predictive models for KDOT's management system has two goals: prediction and description. Prediction involves using some variables in the data base to predict unknown values of interest, in this case, activity duration. Description involves finding regularities underlying the data, which are interpretable to the domain engineers. These two goals can be accomplished when the appropriate numerical relationships are induced from the data base.

This report presents a method, combining machine learning technique and regression analysis, to automaticallly and intelligently update predictive models used in KDOT's internal project management system. Different predictive models are used in different projects. The predictive models used by KDOT consist of Planning factors (predictive equations) and base quantities (Scaling factors), which are applied to predict durations and resources of Functional Units (defined as subsubactivities). The method developed may be used for either task durations or resource requirements, but this report focuses on duration examples due to KDOT's priority in first updating these models.

The method proposed is a three stage learning process. The first stage is concerned with data analysis to obtain new knowledge of each Functional Unit. The second stage is concerned with searching for Functional Units that behave similarly. The third stage is concerned with generating new planning factors and the base quantities used to scale a single planning factor for use to predict several Functional Unit activity durations.

17. Key Words	18. Distribution Statement
Task duration, predicitve model, data base, machine learning, gram, pier, abutment.	No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22161

19. Security Classification (of this Report)	20. Security Classification (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	100	

Form DOT F 1700.7 (8-72)

DEVELOPMENT OF PROJECT ACTIVITY DURATION AND RESOURCE REQUIREMENT ALGORITHMS BASED ON HISTORICAL DATA

KTRAN Project No. KU-95-7

Final Report

April 1997

By
W. M. Kim Roddis
Liye Zhang
University of Kansas
Lawrence, KS 66045

PREFACE

This research project was funded by the Kansas Department of Transportation K-TRAN research program. The Kansas Transportation Research and New-Developments (K-TRAN) Research Program is an ongoing, cooperative and comprehensive research program addressing transportation needs of the State of Kansas utilizing academic and research resources from the Kansas Department of Transportation, Kansas State University and the University of Kansas. The projects included in the research program are jointly developed by transportation professionals in KDOT and the universities.

NOTICE

The authors and the State of Kansas do not endorse products or manufacturers. Trade and manufacturers names appear herein solely because they are considered essential to the object of this report.

This information is available in alternative accessible formats. To obtain an alternative format, contact the Kansas Department of Transportation, Office of Public Information, 7th Floor, Docking State Office Building, Topeka, Kansas, 66612-1568 or phone (913)296-3585 (Voice) (TDD).

DISCLAIMER

The contents of this report reflect the views of the authors who are responsible for the facts and accuracy of the data presented herein. The contents do not necessarily reflect the views or the policies of the State of Kansas. This report does not constitute a standard, specification or regulation.

Abstract

Effective planning and scheduling has become increasingly important for the state departments of transportation to efficiently use resources and avoid project delays. Accurate estimates are needed for task durations and resource requirements. The predictive models for durations and resource requirements need to accurately reflect the agency's current business practices and requirements. Updating predictive models is thus important to the improvement of planning and scheduling.

Updating predictive models for KDOT's (Kansas Department of Transportation) management system has two goals: prediction and description. Prediction involves using some variables in the data base to predict unknown values of interest, in this case activity duration. Description involves finding regularities underlying the data, which are interpretable to the domain engineers. These two goals can be accomplished when the appropriate numerical relationships are induced from the data base.

This report presents a method, combining machine learning technique and regression analysis, to automatically and intelligently update predictive models used in KDOT's internal project management system. Different predictive models are used in different projects. The

predictive models used by KDOT consist of planning factors (predictive equations) and base quantities (scaling factors), which are applied to predict durations and resources of Functional Units (defined as subactivities). The method developed may be used for either task durations or resource requirements, but this report focuses on duration examples due to KDOT's priority in first updating these models.

The method proposed is a three stage learning process. The first stage is concerned with data analysis to obtain new knowledge of each Functional Unit. This stage finds the numerical relationship between the duration of the Functional Unit activity and the project attribute values recorded in the data base, the most difficult task in updating the predictive models. The second stage is concerned with searching for Functional Units that behave similarly. This stage identifies which Functional Units can be described by the same planning factor, that is to say, the same independent variables and same form of predictive equation. The third stage is concerned with generating new planning factors and the base quantities used to scale a single planning factor for use to predict several Functional Unit activity durations.

A system called PFactor, written in C and executing both on UNIX and Windows systems, is built based on the proposed method. The tests on artificial data sets show good performance. The performance on real data sets is highly depended on the quality of the training date provided. Due to the strong noise of KDOT's real data sets, the average quality of the updated predictive models is not high. To improve the results obtained from the system, data sets have to be cleaned and preprocessed.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1. General

Transportation engineering is concerned with various transportation systems, such as highways, railways, aviation and other public modes. The main aim of the departments of transportation is to ensure the safe and efficient movement of people and commodities throughout the transportation networks. To meet this goal, many transportation projects have to be done. Generally, transportation projects are complex and costly, requiring great attention to the management of both time and resources. This makes planning and scheduling transportation projects an interesting and important subarea of transportation engineering.

Construction project planning and scheduling can be viewed as falling into two stages. The first stage involves planning and scheduling of project development and engineering, a task that is typically the responsibility of agencies like KDOT that manage many construction projects. This stage deals with planning and scheduling project preparation going on in the agency itself before release of the project for bid. It stresses the allocation of resources within the agency. The second stage involves planning and scheduling of project execution and construction, a task that is typically the responsibility of the contractors. Instead of covering the whole process of planning and scheduling a construction project, this report deals with the improvement of planning and scheduling transportation projects in the first stage. The particular focus of this report is to improve the prediction of project activity duration.

1.2. Problem Statement

KDOT is in the process of developing construction project activity networks that more accurately reflect the agency's current business practices and requirements. In order to successfully implement the new construction project activity networks, updated duration and resource requirement prediction models are needed. Activity duration and resource requirements are currently based on predictive models which have not been systematically updated in a number of years. This project analyzes historical activity duration data that have been maintained by KDOT. Construction project predictive models based on actual data are derived for use in the new construction project activity networks. These models, in conjunction with project variables, proposed work type, skeletal activity network, and letting data, may be used to determine activity durations for planned projects.

1.3. Background

Two main objectives of managing transportation projects in KDOT are time management and resource management. Failure to properly manage time may result in schedule slippage and cost overruns. Resource management concerns the usage of labor resources within KDOT itself. Accurate duration and resource predictions of activities required for completing a transportation project are crucial to the effectiveness of planning and scheduling transportation projects.

The predictive models currently in use were developed by experienced KDOT personnel in 1983-85. These models were based on a small set of projects (approximately six projects for each fundamental work type, probably restricted to four to five fundamental work types) selected by the experts to be typical cases. The induced predictive models in fact are mathematical functions, using project attributes to predict the project activity durations. The details on predictive models are discussed in Section 2.2, Chapter 2. Applying those predictive models, for example, an activity duration is predicted as 107 days but actual duration is 46 days; another activity duration is predicted as 193 days but actual durations is 43 days. The percentage deviation are 133 and 78 for these two cases respectively, where percentage deviation is defined as

$$\text{Percentage deviation} \times 100 \;=\; \left| \frac{\text{Predicted value} - \text{Actual value}}{\text{Actual value}} \right|. \qquad (1.1)$$

Such inaccuracy of duration prediction obviously results in poor planning and scheduling. It is necessary to update the predictive models in order to improve the effectiveness of planning and scheduling transportation projects for KDOT.

Predictive models are mathematical relations. Mathematically speaking, they use a set of independent variables x_1, x_2, ..., x_m to determine the targeted dependent variable y. In this problem, the targeted variable is the activity duration and the independent variables are the project attributes or a subset of the project attributes. Updating the predictive models consists of finding the numerical relationship between the targeted dependent variable y and the independent variables x_1, x_2, ..., x_m as

$$y = f\{x_1, x_2, \ldots, x_m\} \tag{1.2}$$

based on actual project records. The actual projects are recorded by KDOT in its project management system. In the management system, a database contains all project information, including project attributes and project activity durations. More details on the data records are provided in Section 2.4, Chapter 2.

The data analysis is performed on data extracted from KDOT's management system. The intended result is that the new numerical relationship thus induced from historical data will give numeric prediction for the variable y of new cases more accurately than the current models. In addition, it is intended that the new models can be systematically updated as more data is collected.

Building new predictive models is difficult and time consuming, especially when it is desired to update predictive models as more data is collected. Traditional regression based methods for finding numerical relationships succeed when the relationships to be discovered are homogeneous, that is to say, the same relationships between variables hold over the entire problem domain. These methods also require the variables to be of one type, that is to say, the variables are all numeric. In addition, traditional regression analysis must assume a model a priori. That is to say, the sets of independent and dependent variables are predefined. However, the real world engineering problem under study does not satisfy these requirements of the traditional methods. The characteristics of the problem thus prevent the direct application of traditional regression analysis. A different method is needed to automatically, intelligently, and efficiently update the desired predictive models.

Machine learning, a rapidly developing subarea of artificial intelligence, provides very powerful tools for updating predictive models. A method combining machine learning techniques and statistical analysis is used in this study to analyze historical data to derive new predictive models.

1.4. Objective and Scope

Updating predictive models for KDOT's management system has two goals: prediction and description. Prediction involves using some variables in the data base to predict unknown values of interest, in this case activity duration. Description involves finding regularities underlying the data, which are interpretable to the domain engineers. These two goals can be accomplished when the appropriate numerical relationships are induced from the data base.

Development of the predictive models begins with review of the current planning value table structures (i.e., predictive model structures). Historical data base records from KDOT's project management systems, RMS (Resource Management System) and CPMS (Comprehensive Program Management System), are used to construct the new models. The methodology proposed to develop the new predictive models consists of a combination of computer induction and statistical approaches, primarily linear regression. The project assumes that KDOT provides available necessary data sets in an electronic form. The quality of the resulting predictive models is highly dependent on the quality of the input historical data provided by KDOT.

A method that can efficiently and intelligently update predictive models is proposed, which combines machine learning and statistic analysis. A system, called PFactor, is built based on the proposed method. The testing of the system shows improved performance compared to current models. The establishment of such a system can keep the predictive models updated over time by modifying the models when additional data is collected.

This report presents the results of building the system. The report is arranged as follows: Chapter 2 outlines the planning and scheduling procedure and the predictive models used in KDOT's existing planning and scheduling management system. Chapter 3 briefly reviews the machine learning techniques, presents the method used in updating predictive models and the algorithmic basis of the PFactor system. Chapter 4 discusses the system PFactor. Chapter 5 shows the performance of the PFactor system in comparison to the performance of the current KDOT models.

Chapter 2.

Predictive Models in Transportation Planning and Scheduling

There are many ways to do planning and scheduling [Stella and Glavinich, 1994] including various predictive models. In this section, we first briefly review the planning and scheduling method and the predictive models [planning value tables] used by KDOT in its internal management system. Then, we discuss the domain knowledge underlying the current predictive models. Finally, we discuss the process of updating predictive models.

2.1. Planning and scheduling

KDOT classifies the various types of transportation projects by templates. Generic planning templates are available for typical project types such as: bridge replacement, new road construction, pavement overlay, etc. All template types used by KDOT are listed in Table 2.1. To plan and schedule a transportation project in its management network, KDOT follows the steps shown in Fig. 2.1.

Table 2.1 List of templates in use

1. LOCATION STUDY (LSTD2)
2. GRADE, BRIDGE & SURFACE (GBS03)
3. GRADE, BRIDGE & SURFACE (GBS04)
4. BRIDGE REPLACEMENT (BRRL3)
5. 3R GRADING & SURFACING (3R002)
6. OVERLAY
7. BRIDGE OVERLAY (BROV2)
8. BRIDGE PAINT (PNT01)
9. BRIDGE REPAIR (BRPR2)
10. BRIDGE (BR002)
11. 3R WITH BRIDGE REPLACEMENT (3RRPL)
12. SURFACING (SU002)
13. SIGNING (SG001)
14. LIGHTING (LT001)
15. SPECIAL (SPEC1)
16. SEALS (SEAL1)
17. SAFETY REST AREA (SRA01)
18. MISCELLANEOUS

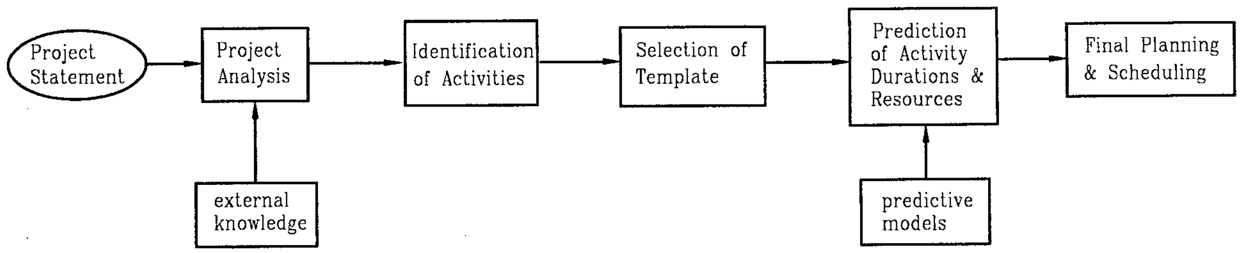

Fig. 2.1. Procedure of managing a project.

When a new transportation project is planned, the project statement is given to an analyst. The first step is for the analyst to break down and review the transportation project according to its project statement. This requires the review of the project scope and objectives. The results of this step is the identification of all activities that must be performed in order to complete the project. Then the analyst uses his/her knowledge to choose a generic template that most closely matches the project type. That is to say, an activity network is selected for the project. The following step is to predict how long each activity of the project will take according to the predictive models stored in the management system. The final step is to generate a complete plan and schedule either by forward or backward pass calculation [Stella and Glavinich, 1994]. To finish the project, the activities in the activity network have to be done according to the activity arrangements of the activity network [Activity networks of templates]. In the following, we look at the project activity networks in order to understand the predictive models used in KDOT's management system.

For a typical template, an activity network flow chart is used to plan and schedule a construction project. The network adopts the Critical Path Method [Stella and Glavinich, 1994] in planning and scheduling. Fig. 2.2 shows the activity network flow chart of a generic template suitable for a bridge replacement.

The basic components of the management network for a project are Work Phases, Events, and Activities. Work Phases are comprised of Events and Activities. Events are either Milestones and/or Border Check Points (lesser significant Milestones). The components of the management network are shown in Fig. 2.2. For instance, the Utility Work Phase is comprised of the events of UTILP (Utility Plans), UTAGR (Utility Agreement Complete) and UTCOM (Utility Adjustment Complete), and the activities of UTENG (Utility Engineering) and UTADJ (Utility Adjustments). The Milestones and Border Check Points are the beginning or ending of an Activity, and mark a particular point in time for reference or measurement. They do not take any elapsed time in planning and scheduling.

Activities are associated with time and time is their important factor. An Activity can start when all predecessors to that Activity are complete. For instance, the activity of PS&E (Plans, Specification & Estimates) in Fig. 2.2 can start only when its predecessors of UTADJ (Utility Adjustments), FIDES (Final Design) and RWCDM (Right of Way Condemnation) are all finished.

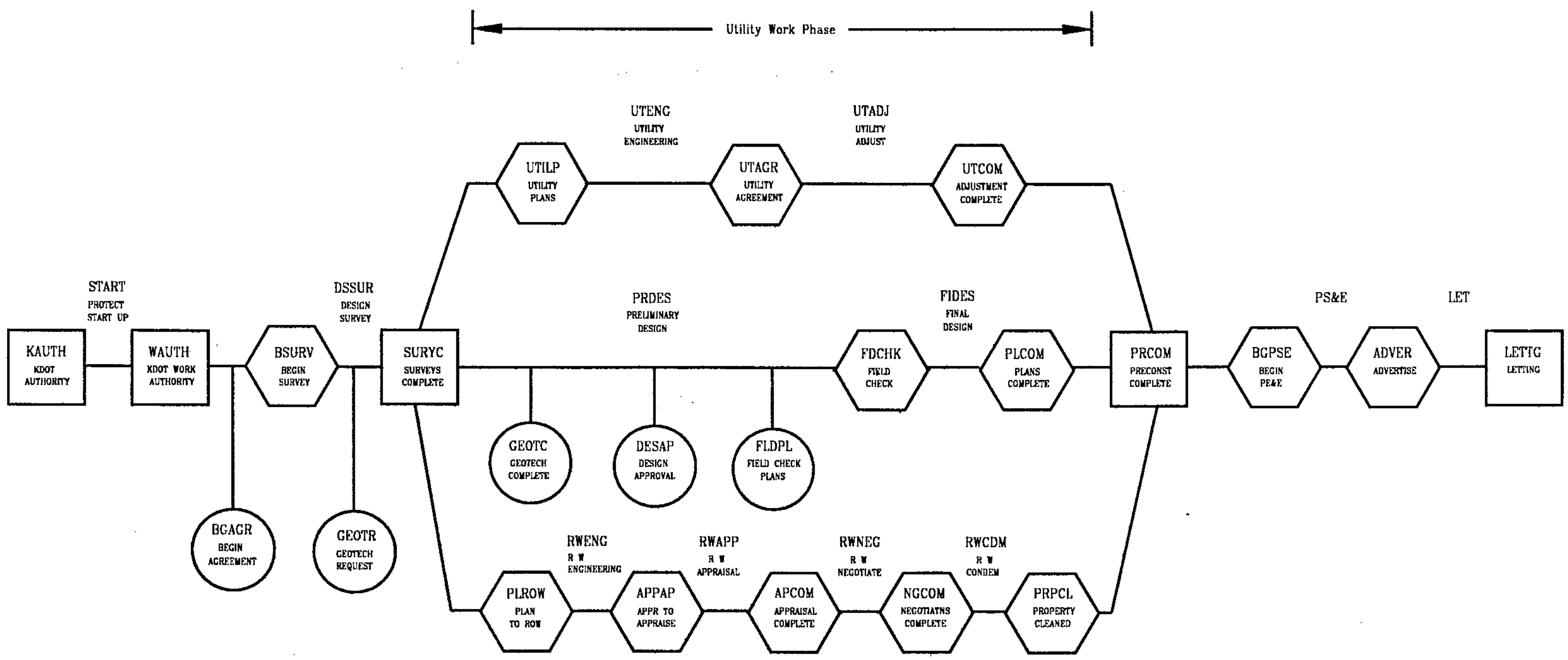

Fig.2.2. Activity network flow chart of a generic template

Table 2.2 Activities contain their Functional Units in the template of 3R/Bridge replacement.

Activities	Functional Units
START	START
DSSUR (DESIGN SURVEY)	FSURV (FIELD SURVEY) DESUP (DESIGN SUPPORT) DMATL (DISTRICT MATERIALS) MATLS (MATERIALS)
PREDES (PRELIMINARY DESIGN)	BRIDG (BRIDGES) ENVIR (ENVIRONMENTAL) GEOL (GEOLOGY) PVMNT (PAVEMENT) SOILS (SOILS) ROAD (ROAD DESIGN)
FIDES (FINAL DESIGN)	BRIDG (BRIDGES) TRCO (TRAFFIC CONTROL) ENVIR (ENVIRONMENTAL) GEOL (GEOLOGY) LANDS (LANDSCAPE) PVMT (PAVEMENT) ROAD (ROAD DESIGN) SIGNS (SIGNING) SOILS (SOILS)
RWENG (RIGHT OF WAY ENGINEERING)	RWENG (RIGHT OF WAY ENGINEERING)
RWAPP (RIGHT OF WAY APPRAISAL)	RWAPP (RIGHT OF WAY APPRAISAL) CONOF (CONSTRUCTION OFFICE)
RWNEG (RIGHT OF WAY NEGOTIATIONS)	CONOF (CONSTRUCTION OFFICE) RWACQ (RIGHT OF WAY ACQUISITIONS) RQMGT (RIGHT OF WAY MANAGEMENT) RWREL (RIGHT OF WAY RELOCATIONS)
RWCDM (RIGHT OF WAY CONDEMNATION)	LEGAL (LEGAL)
UTENG (UTILITY ENGINEERING)	UTIL (UTILITIES) CONOF (CONSTRUCTION OFFICE)
UTADJ (UTILITY ADJUSTMENT)	CONOF (CONSTRUCTION OFFICE)

Further, an Activity consists of subactivities called Functional Units. As shown in Table 2.2, the activity of DSSUR (Design Surveys) includes the functional units of FSURV (Field Survey), DESUP (Design Support) , DMATL (District Materials) and MATLS (Materials). The Functional Units of an Activity can be performed at the same time, and their duration may be different. Therefore, the duration of an Activity is determined by the one of its Functional Units whose duration is the longest. The ideal relationship between an Activity and its Functional Units is shown in Fig. 2.3. In Fig. 2.3, indicates the duration of Functional Units. Actual cases involving fragmented and nonoverlapping scheduling of functional unit times are discussed in Chapter 5 under data quality issues.

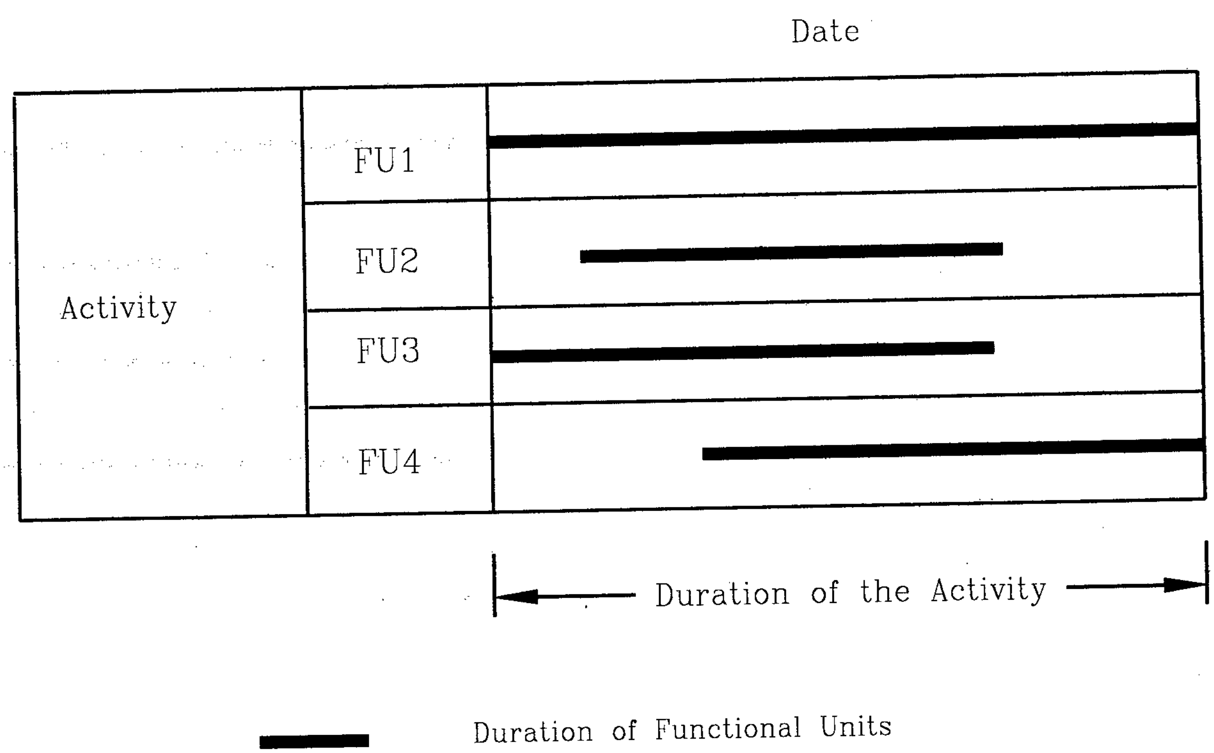

Fig. 2.3 Relationship of durations between an Activity and its Functional Units.

After the determination of Functional Unit durations, Activity durations are easily evaluated. Finally, the total duration of a project is determined by the summation of the time taken by the Activities on the critical path. The critical path is defined as the longest continuous chain of activities through the network schedule that establishes the minimum overall project duration [Stella and Glavinich, 1994].

According to the above discussions, it is clear that the more accurate the duration prediction of Functional Units, the more accurate the duration prediction of Activities, resulting in more effective planning and scheduling.

2.2. Predictive models

The above discussion displays the importance of the duration prediction of Functional Units. The duration of Functional Units of a project strongly depends on the attributes of the project. The attributes of a project include road length, the number of lanes and bridges, the location of a project, etc. In other words, the attributes of a project define the project. Based on the information of attributes, KDOT personnel analyze the project, identify the activities, select an appropriate project template, determine the durations of Functional Units, and complete the planning and scheduling of the project.

The attributes used in defining a project managed in KDOT are summarized from predictive models [Planning value tables] and listed in Table 2.3. Table 2.3 shows that a project has many attributes and the attributes are of mixed types, i.e., symbolic and numeric. An attribute is symbolic when its values are unordered and the number of its values is finite [Breiman, Friedman, Olshen and Stone, 1984]. For instance, the attributes <US81_ind> and

<Urban_ind> in Table 2.3 are symbolic. An attribute is numeric when its values are ordered and the number of its values is infinite [Breiman, Friedman, Olshen and Stone, 1984]. For example the attributes <Bridges> and <Length> in Table 2.3 are numeric.

In terms of project attributes, the duration of a Functional Unit can be described as

$$d = f\{\text{ attributes}\} \tag{2.1}$$

where d denotes the duration of the Functional Unit. Eq. (2.1) gives the general forms of duration predictive models for Functional Units.

Compared with Eq. (1.2), the duration, d, is the targeted dependent variable y while the project attributes a_1, a, ..., a_m are the independent variables x_1, x_2, ..., x_m. The construction of the predictive models involves finding the numerical relations between the targeted dependent duration d and the independent attributes a_1, a, ..., a_m.

Eq. (2.1) indicates that in the most general case the duration is assumed to be a function of all attributes listed in Table 2.3. However, several of those attributes usually dominate the influence on the duration of a particular Functional Unit. For example, the duration of some Functional Units depends only on the attributes <Urban_ind> and <Length>. To establish the current predictive model for the duration of a particular Functional Unit, the experts in planning and scheduling based on their experiences determined:

Table 2.3 Attributes related with planning factors.

ATTRIBUTE NAME	ATTRIBUTE VALUES	TYPE
Access ind	Controlled or Uncontrolled	Symbolic
Borrow	Yes or No	Symbolic
Bridges	The number of bridges	Numeric
Bridge replacement	The number of bridge replacement	Numeric
Bridge width	Numeric	Numeric
Bridge length	Numeric	Numeric
Construction under traffic	Yes or No	Symbolic
Crossing	Small, Medium, Large, etc.	Symbolic
Design	In-House or Consultant	Symbolic
Distance	Travel miles from Topeka	Numeric
FHWA improvement type	Integer indicating different types	Symbolic
Lanes	Two, Four or Six	Symbolic
Length	Numeric	Numeric
Light tower	The number of light tower required	Numeric
Location study	Major, No-major, etc.	Symbolic
Location construct	New or Existing	Symbolic
Metro	Normal or High	Symbolic
Places	Kansas City, Wichita, Topeka or Others	Symbolic
Relocation	Yes or No	Symbolic
Sign footing	The number of sign footing required	Numeric
Sign project	New or Modified	Symbolic
Sign truss	Yes or No	Symbolic
Surface work type	Grading & surfacing , Grading or Surfacing	Symbolic
Surface material	Bituminous or Concrete	Symbolic
Time	Time of letting: Jan., Feb.,..., Dec.	Symbolic
Tracts	The number of tracts to be purchased	Numeric
Tracts relocated	The number of relocated tracts in negotiation	Numeric
Tracts condemned	The number of relocated tracts in condemnation	Numeric
Urban ind	Urban or Rural	Symbolic
US81 ind	East or West	Symbolic
US283 ind	East or West	Symbolic
Utilities	The number of utilities	Numeric
Utilities required	Yes or No	Symbolic

- The subset of attributes that dominate the influence on the duration of the particular Functional Unit. That is to say, the experts selected the attributes judged to be the significant independent attributes on which the duration of the Functional Unit depends. Different Functional Units may have different significant attributes.
- The way the significant attributes determine the duration of the particular planning factor. That is to say, the experts specified the numerical relationship between the significant attributes and the dependent duration of the Functional Unit.

KDOT manages many transportation projects of a variety of types. More than 18 templates as shown in Table 2.1 are stored in the management system to classify the various types of projects. Each template includes many Activities which further consist of many Functional Units as shown in Table 2.2. Therefore, there are hundreds of Functional Units in the management system when projects are viewed at the level of Functional Units. If the duration of each Functional Unit is predicted by its own predictive model like Eq. (2.1), this would lead to an excessive number of models (mathematical functions) for Functional Units. However, analysts at KDOT observed that some Functional Units behave similarly, with durations differing only by a constant. That is to say, those Functional Units have the same significant attributes and those significant attributes influence the duration in the same way except for the influence magnitude. For example, there are three Functional Units FU1, FU2 and FU3. Their durations are simply expressed as

$$\begin{aligned} &\text{FU1: } d = 20 * \langle \text{Length} \rangle; \\ &\text{FU2: } d = 40 * \langle \text{Length} \rangle; \\ &\text{FU3: } d = 40 * \langle \text{Bridges} \rangle. \end{aligned} \tag{2.2}$$

It is said that the Functional Units FU1 and FU2 behave similarly, but Functional Unit FU3 behaves differently from Functional Units FU1 and FU2.

This can be accounted for in the predictive model by splitting the duration of a particular Functional Unit into two parts B and p. Consequently, the duration of a Functional Unit is measured by the product of B and p

$$d = B \times p \tag{2.3}$$

where B is a constant related to the Functional Unit and independent of the attributes, and p is a function associated with the attributes. B and p are called a **base quantity** and a **planning factor** [Project templates] respectively in KDOT's project management system. In other words, predictive models consist of planning factors and base quantities. The durations of Functional Units are proportional to their corresponding planning factors.

The introduction of planning factors in KDOT's planning and scheduling system is very significant, allowing the system to predict the duration of many Functional Units in terms of a small number of planning factors. Some Functional Units even in different templates may share the same planning factor by having their own base quantities. Currently, there are 65 planning factors in addition to a base quantities for each Functional Unit to handle the duration and resource prediction of all transportation projects in KDOT [Planning value tables, Project templates].

2.3. Domain knowledge in predictive models

The process of updating the predictive models used in KDOT's management system may gain valuable direction from review of the current predictive models. These current models are based on the experience of skilled analysts, encoding expert knowledge of the particular problem area. In the field of knowledge based systems, a particular problem area is called a problem domain and knowledge required for solving problems in that particular area is called domain knowledge. In this section, we discuss the domain knowledge underlying the current predictive models.

Section 2.2 shows that the predictive models used in KDOT's management system consist of planning factors and base quantities. According to Eq. (2.3), Activity durations are proportional to planning factors. Therefore the discussion of predictive models is confined to the planning factors.

KDOT uses in-house project management system software to manage its transportation projects. Before KDOT adopted the current management system, CPMS (Comprehensive Program Management System), in 1992, KDOT had used RMS (Resource Management System) as its management system. Although the systems have dissimilarities, the characteristics of the predictive models do not change except for the format of the planning factors. The planning factors used in RMS have two formats [Planning value tables]: table and chart. For example, planning factor P0039 uses the table format as shown in Table 2.4 and planning factor P0021 uses the chart format as shown in Fig. 2.4. These two formats can be reexpressed in a common conditional equation format as shown below.

For planning factor P0039

$$
\begin{array}{lll}
\text{if } <\text{Location\_construct}> = \text{New } \& <\text{Urban\_ind}> = \text{Rural}, & \text{p.f.} = 1 + c_1 * <\text{Length}> & \\
\text{if } <\text{Location\_construct}> = \text{New } \& <\text{Urban\_ind}> = \text{City}, & \text{p.f.} = 1 + c_2 * <\text{Length}> & (2.4) \\
\text{if } <\text{Location\_construct}> = \text{Existing}, & \text{p.f.} = 0 &
\end{array}
$$

For planning factor P0021

$$
\begin{array}{lll}
\text{if} \quad < \text{Urban\_ind} > = \text{Rural}, & \text{p.f.} = \dfrac{< \text{Tracts} > * 3}{150} & \\
\text{if} \quad < \text{Urban\_ind} > = \text{Urban}, & \text{p.f.} = \dfrac{< \text{Tracts} > * 3}{100} & (2.5)
\end{array}
$$

where p.f. denotes planning factor; c_1 and c_2 are constants; <Location_construct>, <Urban_ind>, <Length>, and <Tracts> are the significant attributes. This format is adopted by KDOT's new management system, CPMS, to express planning factors using a single format.

Table 2.4 Planning factor P0039

Length (Miles)	New Location Construct		Existing Location Construct
	Rural	Urban	
1	1.00	1.00	0
2	1.04	1.06	0
3	1.08	1.13	0
4	1.12	1.19	0
5	1.16	1.26	0
6	1.20	1.32	0
7	1.24	1.39	0
8	1.28	1.45	0
9	1.32	1.52	0
10	1.36	1.58	0
11	1.40	1.65	0
12	1.44	1.71	0
13	1.48	1.77	0
14	1.52	1.84	0
15	1.56	1.90	0
16	1.60	1.97	0
17	1.65	2.03	0
18	1.69	2.10	0
19	1.73	2.16	0
20	1.77	2.23	0

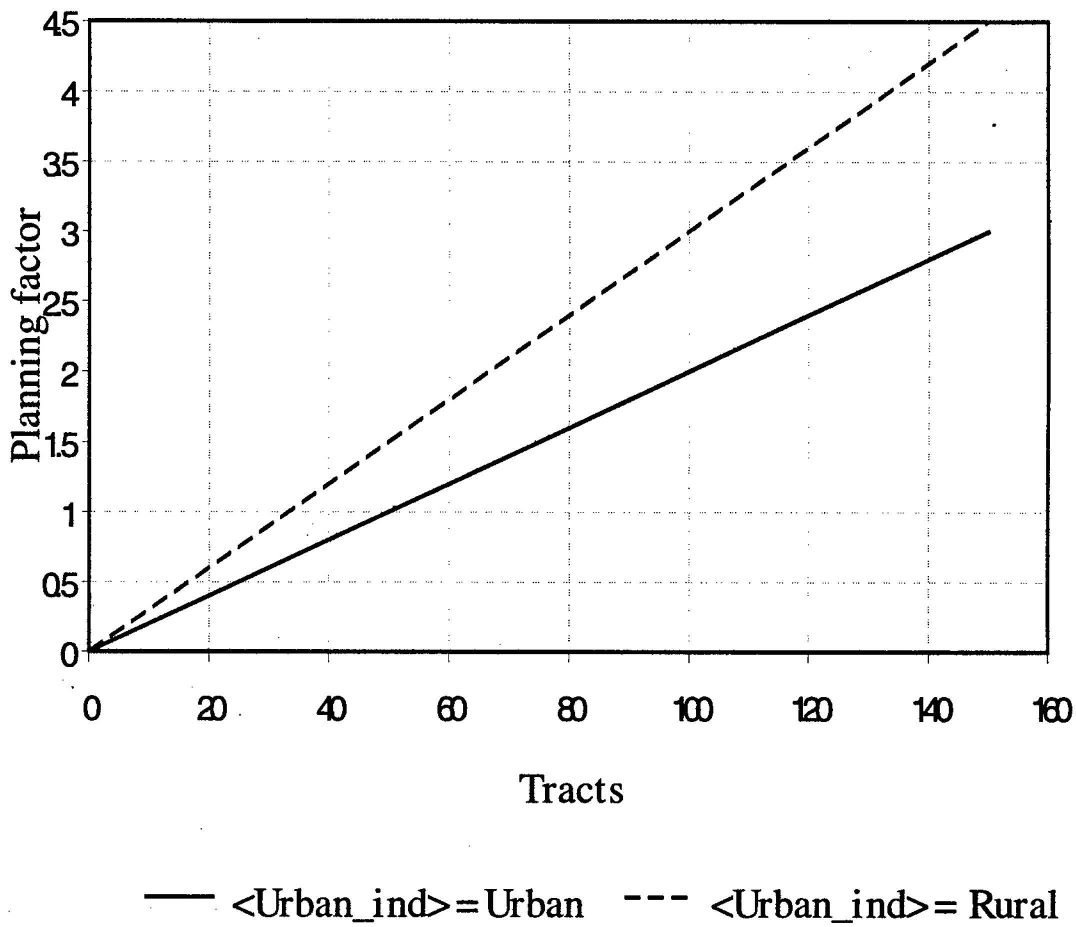

Fig. 2.4 Planning factor P0021.

The current planning factors implicitly represent domain knowledge. An examination of these planning factors [Planning value tables] and the attributes used in the planning factors [Appendix A] show the following useful information about the given problem area:

1) Different planning factors have different significant attributes, i.e., a Functional Unit has its own significant attributes. Those significant attributes can be symbolic or numeric. For example, P0021 has two significant attributes (<Urban_ind> and <Tracts>) while P0039 has three significant attributes (<Urban_ind>, <Location_construct>, and <Length>). The attributes <Tracts> and <Length> are numeric; <Urban_ind>, <Tracts>, and <Location_construct> are symbolic.
2) The space of a planning factor is nonhomogeneous and multidimensional. That is to say, the same numeric relationship does not hold over the entire planning factor space. Rather than express the planning factor as a single function, it is expressed as region:equation pairs. Region descriptions are the "if" parts of the planning factors. For example, planning factor P0021 has two regions (<Urban_ind> = Rural and <Urban_ind> = Urban). These two regions have their own numerical functions. $\frac{<\text{Tracts}> * 3}{150}$ is used in the region (<Urban_ind> = Rural) and $\frac{<\text{Tracts}> * 3}{100}$ is used in the region (<Urban_ind> = Urban).
3) Attributes are used either in region descriptions or region equations. That is to say, the attributes are divided into two groups. One group is used only in region descriptions and the attributes in this group are called region description attributes. The other group of attributes is used only in region equations and the attributes in this group are called

equation attributes. Most description attributes are symbolic and very few description attributes are numeric. The numerical attributes used in descriptions have to be discretized and treated as symbolic. All equation attributes are numeric.

4) The region equations are linear, that is to say, the region equations are linear functions of their significant attributes.

2.4. Process of Updating Predictive Models

Historical data on transportation projects has been collected by KDOT in both the older system, RMS (1980-1991), and the current system, CPMS (1992-now). RMS records planned, estimated, and actual durations while CPMS records only planned and actual durations. The planned durations are derived from the predictive models (planning factors and base quantities). The estimated durations are generated by work group managers. The actual durations match with actual charges on time sheets. Actual for newer projects would have to be inferred from data available in cost center feedback CCFB data bases used for KDOT accounting. The estimated durations contain the prediction by the squad leader early in the process or may reflect a much later revision, usually closer to the actuals. CPMS does not make this distinction between estimated and actual durations.

The goal of updating planning factors is to improve the duration prediction for *actual* duration. Actual duration should be used in data analysis rather than estimated duration. The planned duration should be used in the comparison of the performance of the current and updated planning factors. When a data set is extracted from the master data base, that data set contains examples appropriate for updating a planning factor in the form of a matrix. Each row is a sample record including an actual duration, a planned duration, and project attributes. The detailed information on the format of the data set is given in Chapter 4.

The planned durations are determined by the current predictive models, i.e., planning factors and base quantities that were built by experienced personnel years ago and have rarely been updated. The comparison between the actual and planned durations indicates that the predictive models do not reflect current practice and requirements as shown in Table 2.5. The current predictive models may either overestimate or underestimated durations. The data records in Table 2.5 come from data file "cprms16.txt" discussed later in Section 5.2, Chapter 5.

Table 2.5. Comparison between actual and planned durations of some data records.

Actual Duration (Days)	Planned Duration (Days)	Difference (Actual - Planned) (Days)
34	167	-133
11	120	-109
46	107	-61
189	90	99
174	60	114
193	43	150
244	85	159
336	133	203

Updating predictive models is necessary for KDOT to improve the effectiveness of planning and scheduling. To allow the updated predictive models to be easily integrated with KDOT's existing planning and scheduling system, it is preferred that the format of predictive models remains unchanged, which means keeping predictive models in the same form of planning factors and base quantities. Updating the predictive models thus consists of updating the planning factors and base quantities based on analyzing the available database.

Predictive models predict the durations of Functional Units. Therefore the updating process has to start at the level of Functional Units in order to generate new planning factors and base quantities from the database. From the discussion on predictive models in the last section, the updating process can be viewed as composed of the following three stages

- Analyzing the data set of each Functional Unit
- Searching for Functional Units that behave similarly
- Generating new planning factors and base quantities

1. The first stage: analyzing the data set of each Functional Unit

The first stage of the learning process is to obtain new knowledge of each Functional Unit. In other words, the first stage of the learning process is to find the numerical relationship between the duration of the Functional Unit and project attributes. Therefore, the most general expression of Functional Unit duration is

$$d = f\,\{\text{attributes}\} \tag{2.6}$$

This is a hybrid domain (attributes of symbolic and numeric types) and the duration space is multidimensional and nonhomogeneous. By introducing the domain knowledge discussed in

Section 2.3 into Eq. (2.6), the duration can be expressed as region:equation pairs, where the region description ranges over the description attributes and the region equations are linear functions of the equation attributes. The most general expression of the duration in range R_i is

$$R_i: \quad d = f_i \; \{\text{attributes}\} \tag{2.7}$$

$$= c_{0i} + c_{1i}\, a_1 + c_{2i}\, a_2 + \dots + c_{mi}\, a_m$$

where i depends on the number of attribute values used in region description; R_i represents the ith region description; c_{0i} , c_{1i} , c_{2i} , c_{mi} are region related constants, and a_1, a_2, ..., a_m are project attributes, and m is the number of the attributes used in each region, and m is equal or less than the total number of equation attributes. This equation representation is very important. It represents domain knowledge, i.e., it represents the user's expectations about possible forms of the numerical relationship.

The new knowledge region:equation pairs of a Functional Unit comes from analyzing a data set containing all data related with that Functional Unit extracted from the available RMS and CPMS data bases. The data set includes the project attributes, actual duration and planned duration of the Functional Unit. The output from the first learning stage answers the following:

a) What attributes influence its duration? i.e., what are the significant attributes used in region descriptions and numerical equations?

b) How do the significant attributes determine its duration? i.e., how are the significant attributes used in region descriptions and numerical equations?

As discussed before, not all project attributes influence duration of a typical Functional Unit. Only several significant attributes determine the duration of a typical Functional Unit. However, it is unknown before data analysis what attributes of the projects should be used to determine the duration of the Functional Unit. It can be answered only after analyzing the subdata base of the Functional Unit. Due to the large number of project attributes, the combination of possible relations grows rapidly, making it difficult to select the significant attributes that determine the duration of the Functional Unit. Each of the hundreds of Functional Units in the construction project management system need to follow the same procedure. The method using only regression analysis is thus theoretically possible but practically infeasible. Fortunately, the rapid development of machine learning techniques provides a powerful tool to solve this kind of problem. The method proposed to update the predictive models is a combination of machine learning and statistical regression analysis, discussed in detail in Chapter 3.

2. The second stage: searching for Functional Units that behave similarly

The second stage of updating predictive models needs to identify which Functional Units can be described by the same planning factor. This stage is a process of comparing region-equation pairs. Two Functional Units can be predicted by the same planning factor only when

- The region:equation pairs of two Functional Units have a one-to-one correspondence. The region descriptions are the same in corresponding region:equation pairs.
- The numeric equations in the corresponding pairs are proportional.

Assume two Functional Units have 3 region:equation pairs and their region:equation pairs satisfy the first requirement. Their region-equation pairs are shown in the table below.

Table 2.6 Region:equation pairs of two Functional Units

	Functional Unit 1	Functional Unit 2
Region 1	Eq_11	Eq_21
Region 2	Eq_12	Eq_22
Region 3	Eq_13	Eq_23

The following ratios may be determined for these equations.

$$\frac{\text{Eq\_11}}{\text{Eq\_21}} = e_1, \quad \frac{\text{Eq\_12}}{\text{Eq\_22}} = e_2, \quad \frac{\text{Eq\_13}}{\text{Eq\_23}} = e_3 \tag{2.8}$$

If the numeric equations in the corresponding pairs are proportional, then e_1 , e_2 and e_3 are constants.

- The ratios of the proportionality of corresponding equations are the same. That means

$$e_1 = e_2 = e_3 = \text{constant} \tag{2.9}$$

The new knowledge acquired in this stage is used as the input for the third stage of updating predictive models. The comparison of the region equations is time consuming because of the number of Functional Units, thus this learning process is automated.

3. The third stage: Generating new planning factors and base quantities

The third stage of updating predictive models generates the new planning factors and base quantities. For those Functional Units that can be predicted by the same planning factor, it averages the corresponding parameters of the equations of these Functional Units and obtains the parameters of the planning factor.

The complete process of updating predictive model of base quantities and planning factors is shown in Table 2.7.

Table 2.7 The learning process for updating predictive models.

```
Input       : A set of training examples
Output      : Planning factors and base quantities of Functional Units
Procedure   :
Begin
        decompose the database into subdata bases according to Functional Units
        for each subdata base
          begin
                find d=f {attributes }
          end
        for each two Functional Units
          begin
                check the one to one correspondence;
                check the proportionality of equations in corresponding equations;
                Check the ratio of the proportionality;
                if those three conditions are satisfied
                then use the same planning factor for the Functional Units;
                else use different planning factors.
          end
        for each planning factor
          begin
                for all Functional Units related with the planning factor
                  begin
                        generate planning factor.
                  end
                for each Functional Units
                  begin
                        generate its base quantity based on the new planning factor.
                  end
          end
        Output.
end
```

Chapter 3

Methodology of Updating Predictive Models

The previous chapter introduced the three stage process of updating predictive models. The most difficult task is in the first stage of finding the numerical relationships between durations and attributes, expressed as region:equation pairs. The difficulties come from the following characteristics of the problem: (1) There are many attributes and they are of mixed types, symbolic and numeric. (2) The numerical relations between duration and attributes are multidimensional and nonhomogeneous. (3) Significant attributes are unknown before data analysis.

These problem characteristics preclude a straight forward application of traditional statistical regression analysis. Traditional statistical regression analysis must assume a model a priori (unlike 3 above). It requires variables of one type (unlike 1 above), and also requires that the numerical relations be homogeneous, that is to say, the same relationship is true over the entire domain (unlike 2 above). A significant need exists for a new generation of techniques, combining machine learning and regression analysis, with the ability to intelligently and automatically assist humans in analyzing data bases for useful knowledge. This chapter reviews machine learning techniques, then discusses the proposed method to update the predictive models.

3.1. Machine learning techniques

Machine learning is the subfield of artificial intelligence that is concerned with the design of automatic procedures able to learn from training cases. Since the early 1950s when Turing (1950) proposed this application for computers, machine learning from examples has been an active research area in computer science. The early development of machine learning is briefly reviewed in [Cohen and Feigenbaum, 1982]. Since the 1980s, machine learning has made substantial progress and various machine learning methods have been proposed. In this section, we briefly review the machine learning techniques.

The first paradigm of machine learning techniques uses decision rules, decision trees, or similar knowledge representations. One of the successful algorithms in this paradigm is a tree based algorithm ID3 [Quinlan, 1983], which uses the statistical theory of information. ID4 [Schlimmer and Fisher, 1986] and ID5 [Utgoff, 1988] are incremental induction algorithms for constructing decision trees. Using the ID3 algorithm, C4 and C4.5 were developed by Quinlan in 1987 and 1992 respectively. C4.5 [Quinlan, 1994] is the most popular and successful in this group and its software is commercially available. A limitation of these methods is their requirement of discrete value for attributes. Therefore, continuous attributes have to be discretized for use by these learning algorithms. Binarizing continuous attributes was proposed to deal with continuous attributes in the early 1980s. This method resulted in other algorithms including ACLD and

ASSISTANT [Bratko and Konomenko, 1987; Niblett and Bratko, 1987]. When the predicted decision is ordered continuous numeric value instead of finite classes, CART, constructing *regression trees* [Breiman, Friedman, Olshen and Stone, 1984] and M5, generating *model trees* [Quinlan, 1992], can be applied. The algorithms build the decision trees, generally by greedy search, from root down to the leaves. Information about classes or prediction is stored in the action sides of the rules or leaves of the tree.

The second paradigm of machine learning techniques is case-based or instance based learning. Rather than forming some abstract models such as trees and storing this structure in memory, these methods store instances or cases in memory, and classify unseen cases by referring to similar remembered cases. In other words, they represent knowledge in terms of specific cases or instances and rely on flexible matching methods to retrieve those cases and apply them to new cases. The group contains methods such as nearest neighbor algorithms [Cover and Hart, 1967, Dasarathy 1991], k-nearest neighbor algorithms [Stanfill and Waltz, 1986] and average-case analysis [Langley and Iba, 1993].

The third paradigm of machine learning techniques is neural networks. They represent knowledge as a multilayer network of threshold units that spreads activation from input nodes through internal units to output nodes. Therefore, the knowledge hidden in the data is not explicitly represented. Weights on the links determine how much activation is passed on in each case. Neural networks can be used to predict both real values and classes. The neural network typically attempts to improve the accuracy of classification or prediction by modifying the weights on the links. A comprehensive presentation of various neural networks is given in [Freeman and Skapura, 1991; Skapura, 1996].

The fourth paradigm of machine learning techniques is genetic algorithms, which was derived from the evolutionary model of learning [Forsyth, 1989]. Genetic algorithms use the Darwinian principle of 'survival of the fittest'. A genetic classifier is comprised of a set of classification elements that replicate and mutate to form new generations. But only more successful elements produce variants of themselves and proliferate while elements performing poorly are discarded. The examples of the inductive systems in this group are BEAGLEs, outlined in [Forsyth, 1989].

The fifth paradigm of machine learning techniques concerns numeric law discovery. In recent years there has been growing interest in this field with the rapid growth of the amount of accessible data. The first group was developed from the BACON algorithm [Langley, Simon, Bradshaw and Zytkow, 1987] which was designed to discover scientific laws on the basis of empirical data evidence. BACON systems attempt to find an invariant based on the variable given as input in order to iteratively build the model. But the BACONs seem better able to explain historical laws with artificial data rather than to discover new ones. ABACUS was developed on the basis of BACON [Falkenhainer and Michalski, 1986]. It discards the requirements of the strict monotonic relation of two variables as in BACON. Its controlling search strategy also differs from BACON. IDS was developed by Nordhausen and Langley in 1990. It keeps the basic operators of BACON to define new terms from existing terms. But it applies correlation analysis to direct the search from simple terms to more complex ones instead of trying to find an invariant. A critical review of these methods can be found in [Schaffer, 1990]. KEPLER was suggested by Wu and Wang [1991]. It uses a reduction algorithm to decompose a multivariate formula into binary formulas, then finds the appropriate binary formula by varying two variables at a time and matching data to a set of prototype functions [Wu and Wang, 1991]. These systems are domain independent but have restrictive data requirements including small size, noise free, and one function in the whole domain space.

3.2. Method of Updating Predictive Models

A system that can automatically update the predictive models from the available database would be very helpful to the engineers at KDOT. Such data analysis stresses equally (1) better prediction for new cases and (2) better description of knowledge underlying the predictions. In other words, it is desired that (1) the updated predictive models improve the prediction accuracy and (2) the knowledge hidden in data is expressed as mathematical functions comprehensible to the domain engineers. To update the predictive models in a way that they can be easily installed in the current construction project management system CPMS, the three stage learning process discussed in the Chapter 2 is used. The method used in each stage is presented in turn.

1. The first stage learning process

In the first stage learning process, the numerical relationships between Functional Unit durations and attributes need to be derived. Instead of analyzing the data base from scratch and giving incomprehensible numerical functions, the domain engineers introduce comprehensibility requirements based on the domain knowledge for use in directing data analysis. That is to say, domain engineers give their expectations about possible forms of the numerical relationships.

Therefore, the introduction of domain knowledge restricts the relationships that can be found. Considering the domain knowledge of predictive models discussed in Section 2.3, the expected forms of numerical relationships between the Functional Unit duration and the attributes can be expressed by region:equation pairs shown in Eq. (3.1)

$$R_i: \ d = c_{0i} + c_{1i}\ \mathrm{a}_1 + c_{2i}\ \mathrm{a}_2 + \ldots + c_{mi}\ \mathrm{a}_m \tag{3.1}$$

where R_i is the region description, that is to say, R_i is the region condition setting region boundaries; a_1, a_2,..., and a_m are equation attributes; c_{0i} , c_{1i} , c_{2i} , ... , c_{mi} are region related constants; m is a region related integer (i.e., the number of significant equation attributes used in the region i). Only equation attributes can be used in equations and only description attributes can be used in region descriptions.

To induce unknown region:equation pairs, the tree based models are chosen as the knowledge representation. This knowledge representation fits the application domain and is able to describe the knowledge underlying the data. For example, the region:equation pairs are

$$\begin{array}{ll}
\text{if A} = \text{A}_1 \text{ and B} = \text{B}_1 \text{ and D} = \text{D}_1, & d = f_1 \\
\text{if A} = \text{A}_1 \text{ and B} = \text{B}_1 \text{ and D} = \text{D}_2, & d = f_2 \\
\text{if A} = \text{A}_1 \text{ and B} = \text{B}_2, & d = f_3 \\
\text{if A} = \text{A}_2 \text{ and C} = \text{C}_1, & d = f_4 \\
\text{if A} = \text{A}_2 \text{ and C} = \text{C}_2, & d = f_5 \\
\text{if A} = \text{A}_2 \text{ and C} = \text{C}_3, & d = f_6
\end{array} \tag{3.2}$$

where A, B, C and D are description attributes; A_1 and A_2 are the values of the attribute A; B_1, and B_2 are the values of the attribute B; C_1, C_2 and C_3 are the values of the attribute C; D_1 and D_2 are the values of the attribute C; f_1, f_2, f_3, f_4, f_5 and f_6 are the linear functions in regions. The region:equation pairs of (3.2) can be expressed as the tree shown in Fig. 3.1. Region descriptions are expressed by the nodes and the arcs of the tree. Regional numerical relationships are expressed by the linear equations in the tree leaves.

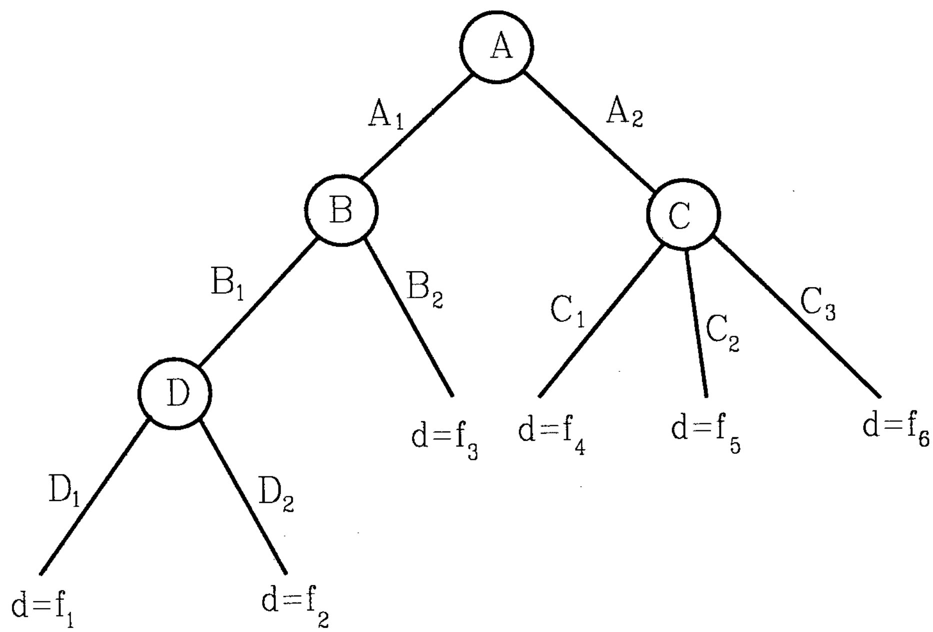

Fig. 3.1 Tree representation of region:equation pairs of Eq.(3.2).

As mentioned before, it is unknown before data analysis what attributes should be used in region descriptions and what attributes should be used in linear equations. Since many attributes are present, choosing the most significant attributes is a formidable task, even if the domain knowledge restricts the choice to a linear function of the significant attributes in each region. A method combining machine learning and regression analysis is proposed to overcome these difficulties so that the numerical relationship between the Functional Unit duration and the attributes can be obtained.

The algorithm used to accomplish the first stage learning process is a tree based algorithm. The tree generated by this algorithm is called an M-model tree. As the M-model tree grows, the algorithm determines what attributes should be put in the nodes of the M-model tree and what attributes should be put in the leaves of the M-model tree. The algorithm starts by randomly dividing the data set into training and testing sets (The percentage of the data set used as testing data is controlled by the parameter TestPer in the program). The training set T is used for building an M-model tree and the testing set is used for pruning the M-model tree.

a. *Building an M-model tree*

An M-model tree is built up by analyzing training cases. The first step of building an M-model tree is to compute the standard deviation [Flannery, Teukolsky and Vetterling, 1988] of the target values of the cases in T that is treated as a measure of error in this process. Unless T contains very few cases or its measure of error is less than a threshold (defined as SdTOL in the program), T is split into two or more subsets T_i on the basis of one of the symbolic attributes in order to make the training cases in the subsets more homogeneous. The default minimum number of cases is 2*(the number of equation attributes in the node) and default threshold (SdTOL) of error measure is 7. However, these can be changed via options described in Chapter 4. The criterion to select an attribute as a node of the M-model tree is evaluated by the expected error reduction [Breiman, Friedman, Olshen and Stone, 1984; Quinlan, 1992]

$$\Delta\, error \;=\; sd\,(\mathrm{T}) \;-\; \sum \frac{\mathrm{T}_i}{\mathrm{T}}\; sd\,(\mathrm{T}_i) \tag{3.3}$$

where $sd\,(T)$ denotes the standard deviation of the set of training case T, and $sd\,(T_i)$ denotes the standard deviation of the subset of training cases T_i. The algorithm uses a greedy search to choose the description attribute that maximizes the expected error reduction. That is to say, the algorithm evaluates all possible splittings based on description attributes, then selects the attribute that gives the maximum error reduction. This process is repeated on the subsets until every subset either contains few cases or the error measure is less than the threshold. Only description attributes not used in ancestor nodes can be selected for the current node.

Multivariate linear models are constructed for the cases at each non-leaf and leaf node of the M-model tree, using standard regression analysis [Flannery, Teukolsky and Vetterling, 1988]. However, instead of using all equation attributes in the standard regression analysis of each node, the equation attributes used in the equation of a node are restricted to the equation attributes inherited from its parent node.

After each linear model is obtained as above, it is simplified by eliminating equation attributes to minimize its weighted standard deviation. Weighted standard deviation of a node is defined as $\Sigma\,(\mathrm{T}_i\,/\,\mathrm{T})\; sd\,(\mathrm{T}_i)$ after an equation attribute is selected for the node. This algorithm uses a greedy search to remove attributes whose elimination decreases the weighted standard deviation. In some cases, the algorithm may remove all attributes, leaving only a constant at the leaf. That is to say, the algorithm evaluates all possible eliminations and then eliminates the one that decreases the weighted standard deviations most. The process is repeated until no attribute decreases the weighted standard deviation or no numerical attribute is left in the linear equation. The reason for eliminating numeric attributes after the selection of symbolic attributes for a non-leaf node is to avoid eliminating significant attributes in the early stage of building up an M-model tree due to strong noise of other insignificant attributes. In leaves, no weighted standard deviation can be found. Therefore, the standard deviation *sd* is used in eliminating numerical attributes rather than the weighted standard deviation. To simplify the equations in leaves, the attributes that do not improve the standard deviation within a threshold are also eliminated.

b. *Pruning an M-model tree*

The recursive partitioning method of constructing the M-model tree continues to subdivide the set of training cases until each subset in the partition contains few cases or the error measure is less than a threshold. The result may overfit the data. After the construction of an M-model tree, testing cases are used to prune the M-model tree in order to simplify the M-model tree so that the simplified M-model tree gives satisfactory prediction without overfitting. Each non-leaf node of the model tree is examined, starting near the bottom after the M-model tree is built up. The algorithm chooses as the final model for this node either the simplified linear model above or the model subtree, depending on which has the lower error estimate (percentage deviation defined in Eq. (1.1)) on the testing data. If the linear model is chosen, the subtree at this node is pruned to a leaf.

c. *Multiple M-model trees*

Due to the noise present in data sets for this engineering problem, a number of M-model trees are generated using different partitioning of training and testing data. The tree with minimum error estimate is selected as the best M-model tree to describe the underlying regularities of the data set. In other words, the process of building and pruning M-model tree is repeated until the error estimate on testing data is less than the preset threshold or the preset repeat times is reached. When the noise is strong, this process becomes more important.

The algorithm for finding the numerical relationship between the Functional Unit duration and the attributes is shown in Table 3.1. A best M-model tree is selected among several M-model trees to describe each subdata set. Therefore, what the first stage of learning obtains is a forest consisting of M-model trees. The outline of this stage is shown in Fig. 3.2.

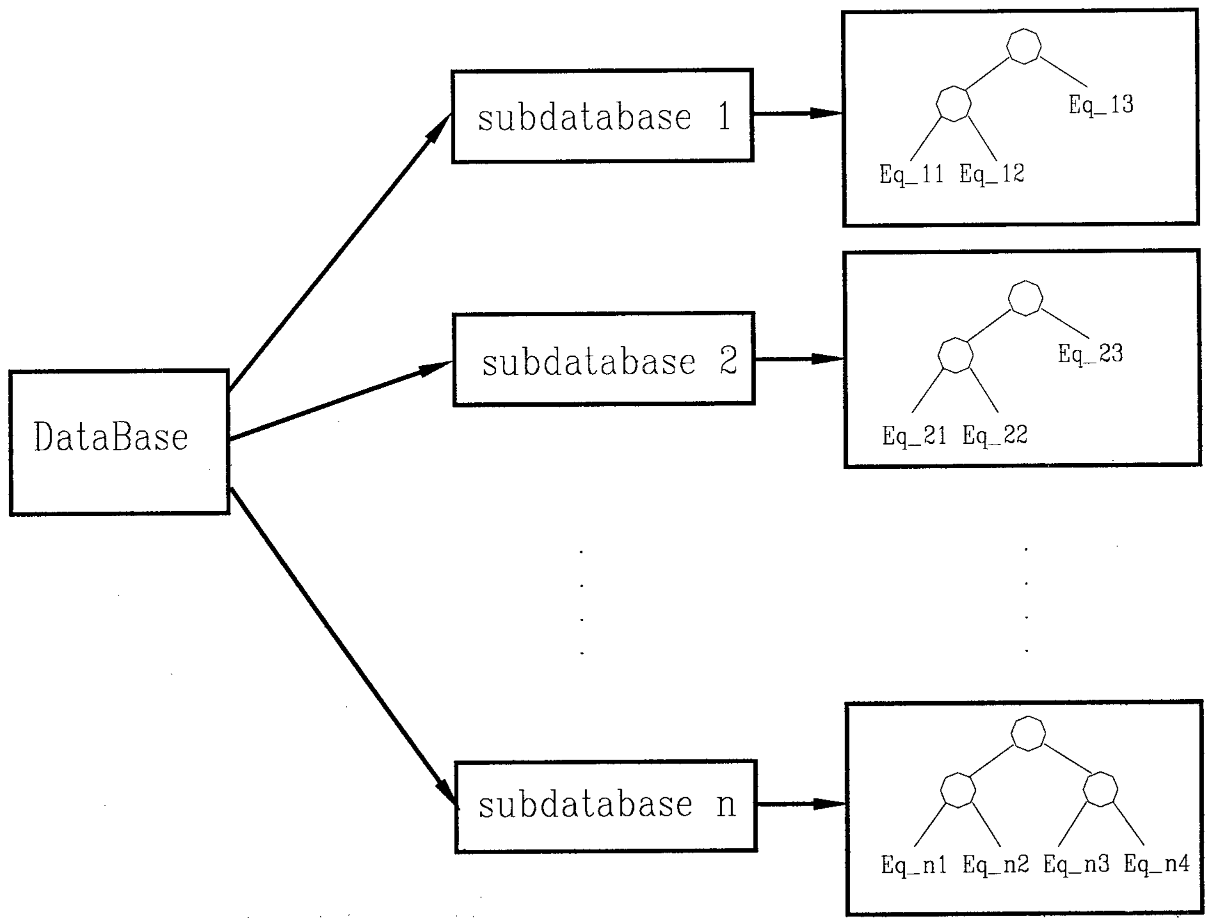

Fig 3.2 The results of the first stage learning process

Table 3.1. The algorithm for finding numerical relations between duration and attributes.

```
Input       : A set of examples
Output      : Numerical relationship between duration and attributes
Procedure   :
  begin
     for the repeat times
       begin
          divide the examples into training and testing sets
          calculate the standard deviation sd of the training set
          for each non-leaf node
            begin
               find the attribute maximizing error reduction
               use the attribute in the node
               simplify linear equation by weighted standard deviation
            end
          for each leaf
            begin
               simplify the numerical relation by standard deviation
            end
          prune the tree by testing set
          if the current tree is better, current tree becomes the best tree;
          else keep the best tree.
          if the threshold of error estimate is satisfied, break;
       end
     Output.
  end
```

2. The second stage of learning process

The second stage of learning process is to compare the numerical relationships of all Functional Units in order to find out what Functional Units behave similarly, in other words, it compares the M-model trees of the forest built up in the first stage learning process. As mentioned in Section 2.4, Chapter 2, if two Functional Units behave similarly, they have to satisfy three conditions. Here, we look at these three conditions when they are applied to M-model trees.

- The first condition requires that the region:equation pairs of two Functional Units are in one-to-one correspondence, and the region descriptions are the same in corresponding region:equation pairs. This requirement means that the M-model tree structures are the same when the requirement is applied to M-model trees. The attributes used in the corresponding nodes should be the same and the attribute values in the corresponding arcs are the same.
- The second condition requires that the numerical equations in the corresponding pairs are proportional. That is to say, the equations in the corresponding leaves of the M-model trees are proportional when this requirement is applied to M-model trees. Here we look at how two equations are proportional.

  If two equations are proportional, the same numerical attributes are used in the equations, which means if only a_1 and a_2 are used in one of two equations, the other equation also has only a_1 and a_2. In addition, the corresponding coefficients of the same attributes are proportional. For example, there are two equations Eq1 and Eq2.

$$\text{Eq1}: \quad d = c_{01} + c_{11}\,\mathrm{a}_1 + c_{21}\,\mathrm{a}_2 \tag{3.4}$$

$$\text{Eq2}: \quad d = c_{02} + c_{12}\,\mathrm{a}_1 + c_{22}\,\mathrm{a}_2$$

where a_1 and a_2 are numeric attributes. If they are proportional,

$$\frac{c_{02} + c_{12}\,\mathrm{a} \;\; + c_{22}\mathrm{a}_2}{c_{01} \;\; + c_{11}\mathrm{a}_1 + c_{21}\,\mathrm{a}_2} = \frac{e(c_{01} + c_{11}\,\mathrm{a}_1 + c_{21}\,\mathrm{a}_2)}{c_{01} + c_{11}\,\mathrm{a}_1 + c_{21}\,\mathrm{a}_2} = e = \text{constant} \tag{3.5}$$

That is to say

$$\begin{aligned}
e\,c_{01} &= c_{02} & &\rightarrow & \frac{c_{02}}{c_{01}} &= e = c_0 \\
e\,c_{11} &= c_{12} & &\rightarrow & \frac{c_{12}}{c_{11}} &= e = c_1 \\
e\,c_{21} &= c_{22} & &\rightarrow & \frac{c_{22}}{c_{21}} &= e = c_2
\end{aligned} \tag{3.6}$$

Further

$$c_0 = c_1 = c_2 = e = \text{constant} \tag{3.7}$$

Obviously, it is impossible to make the condition Eq. (3.7) be exactly satisfied in practice. Instead of requiring $c_0 = c_1 = c_2 = e =$ constant, the algorithm requires the relative difference among c_0 , c_1 and c_2 to be less than a threshold, ConstTOL (Constant TOLerance in the program). That is to say

$$\left|\frac{c_0 - c_1}{c_0}\right| < \text{ConstTOL} \qquad \text{and} \qquad \left|\frac{c_0 - c_2}{c_0}\right| < \text{ConstTOL} \tag{3.8}$$

if $c_0 \neq 0$. The default value of this threshold ConstTOL is 15%, but users can change it via an option discussed later. The lower the threshold, the lower the possibility that two equations are proportional. When two equations are proportional, the average of the ratios of coefficients c_0 , c_1 and c_2 will be the ratio e of two equations.

$$e = \frac{1}{3}(c_0 + c_1 + c_2), \tag{3.9}$$

- The third condition is that the ratios of all corresponding equations are constant. That is to say, the ratios of the equations in all corresponding leaves are constant when this requirement is applied to M-model trees. For example, two trees have 4 corresponding leaves and the equations in corresponding leaves are proportional. Their ratios of corresponding equations are e_1 , e_2 , e_3 and e_4 . When

$$e_1 = e_2 = e_3 = e_4 = \text{constant} \tag{3.10}$$

it is said that the requirement is satisfied. As with the previous requirement, it is also impossible to get the condition $e_1 = e_2 = e_3 = e_4 =$ constant to be exactly satisfied in practice. As previously, this condition is assumed to be satisfied when the relative differences among the ratios e_1 , e_2 , e_3 and e_4 are less than a threshold. This threshold is again taken as ConstTOL. Therefore, the condition of Eq. (3.10) becomes

$$\left|\frac{e_1 - e_2}{e_1}\right| < \text{ConstTOL} \quad \text{and}$$

$$\left|\frac{e_1 - e_3}{e_1}\right| < \text{ConstTOL} \quad \text{and} \tag{3.11}$$

$$\left|\frac{e_1 - e_4}{e_1}\right| < \text{ConstTOL}$$

This threshold is the same as the one used in the second requirement. The ratio of two M-model trees B will be the average of the ratios of all corresponding equations.

$$B = (e_1 + e_2 + e_3 + e_4)/4 \tag{3.12}$$

The algorithm will check the requirements from the first to the third to determine the similarity of M-model trees. The algorithm for comparing two M-model trees is shown in Table 3.2.

Table 3.2. The algorithm for comparing two trees.

Input	: two M-model trees
Output	: if the trees are proportional and similar
Procedure	:
begin	
	check if the model tree structures are the same.
	check the proportionality of equations in the corresponding leaves.
	check if the ratios of equation proportionality are constant
	if those three conditions are satisfied
	then the trees are proportional and similar;
	else the trees are not proportional.
	Output.
end	

3. The third stage of learning process

Based on the comparison of the M-model trees in the forest of the second stage learning process, the trees are divided into groups. In each group, the M-model trees are proportional to each other. Base quantities for each M-model tree and one planning factor will be obtained for every group of the M-model trees. Assume a group has two M-model trees. If the first is taken as the primary tree, 1 and B are the base quantities for the first and second tree respectively. The common tree is

$$\frac{1}{2}\left(\text{tree1} + \frac{\text{tree2}}{\text{B}}\right) \tag{3.13}$$

This common tree represents the planning factor for this group of trees. This tree can be changed to the expression of region:equation pairs. The output form of the planning factor is in the region-equation pairs rather than the tree representation.

Chapter 4

System PFactor

Applying the method discussed in the last chapter, a system called PFactor is developed to accomplish the task of updating the predictive models used at KDOT. We discuss how PFactor is used in this chapter before showing the performance of the system PFactor in the next chapter.

4.1. System Structure

The source code of the system PFactor is written in C language on a Unix System. It consists of 20 files. The brief descriptions of these files are given in Appendix B. Because the source code of the files is very long (more than 3000 lines), the source code is not printed out and put as appendix. The structure of the system is arranged as shown in Table 4.1.

Table 4.1 System structure.

GetData	Read data set from input data file
GroupData	Group data into subdata sets
BuildForest	Build an M-model tree for each subdata set
PFactors	Combine the second and third stage learning process
CompErr	Compare the error estimate by current and updated predictive models (planning factors and based quantities).
Output	Output the updated predictive models (planning factors and based quantities).

Table 4.2. Input data file format.

<	p	b	d	e	a	a	a	x	....	a	>
[	s	n	n	n	s	n	s	s	....	n	]
{	pfactor2	base_Q	Duration	Duration	City	Length	Lane	Material	....	Attr. *n*	}
	P0002	10	35	67	Y	2.3	2	C	....	x	
	P0002	5	54	40	N	10	4	A	....	x	
	:	:	:	:	:	:	:	:	....	:	
	:	:	:	:	:	:	:	:	....	:	

4.2. Input file

Before the system begins, the user needs to give the system an input file name, which stores the data for learning. The input file has format requirements.

The format of an input file is shown in Table 4.2. The data file has to start with attribute information, which includes three lists. The list information is followed by learning examples. The first list gives information on attribute status, the second list gives information on attribute types, and the third list gives information on attribute names.

List 1 must start with "<", followed by a sequence of the following symbols: "p", "b", "d", "e", "a", "x" and "?". The last symbol of the list must be ">". "b" and "d" have to be in the list, and they can appear in the list only once. "p" and "e" can be in the list only once if they appear in the list. But "a", "x" and "?" can be in the list more than once depending on the attributes in the learning data. The order of these symbols is not a mandatory. The elements of the lists must be separated by space(s) or tab(s) or comma(s) ",". Delimiter choice is up to the users' preference.

p : Indicates that the column stores current planning factor identification.

b : Indicates that the column stores current base quantities.

d : Indicates that the column stores *actual* durations.

e : Indicates that the column stores *estimated* durations by current predictive models.

a : Indicates that a column stores an attribute that will be used in deriving new predictive models.

x : Indicates that a column stores an attribute that will not be used in deriving new predictive models. Introduction of both "a" and "x" to denote attribute status is due to the consideration that some attributes obviously do not influence the duration. This information given by users will greatly improve the efficiency of the machine learning process.

? : Indicates that a column stores an attribute that can not be described by any of the above characters.

<u>List 2</u> must start with "[", followed by a sequence of the following symbols: "s" and "n". The last symbol of the list must be "]". This list gives information on the type of attributes in the column, description attributes or equation attributes. Description attributes are used only in region descriptions and equation attributes are used only in region equations. Delimiters are the same as list 1.

s : description.

n : equation.

<u>List 3</u> must start with "{", followed by a sequence of the names of each column. The last symbol of the list must be "}". Delimiters are the same as list 1. The elements of this list store the names of the columns. Each name can be only one string. For instance, "US81 indicator" is not a correct attribute name, but "US81_indicator" is a legal attribute name. It is suggested

that the number of characters of attribute names be no more than 15 characters although no limitation is put on it.

Learning examples follow these three lists. The elements of the examples must be separated by delimiters as for list 1.

When reading data from the input data file, the system checks the data format. If the data format does not satisfy the format requirements, the system exits with an error message. For example, if no first list is given in the data file, the system outputs the message "No first list of attribute information in data file". If no element in the first list is "d", the system outputs the message "No actual duration in data file".

4.3. Running Options

The system PFactor provides many options for users to control how PFactor behaves. These options can be used in any order.

-c ConstTOL (default = 0.3)

ConstTOL is the threshold that controls the proportionality of equations and trees, see Eqs (3.7) and (3.9). If this option is invoked, it will change the threshold of comparison of equations and trees. The smaller the ConstTOL, the lower the possibility that two equations/trees are proportional. It is suggested that ConstTOL be less than 0.5, otherwise strongly non-proportional equations and trees may be taken as proportional.

-i infile (default = no input file name)

This option **must** be used in order to tell the system where to get data. If no input file name is given or the given input file does not exists, the system will terminate the execution with a message. Generally, a filename can be any string of characters that is acceptable as a file name to the operating system. In addition, the input file has to follow a certain format described in section 4.2.

-p SdPER (default = 15 percent)

If this option is invoked, the threshold used to simplify linear equations will change. The smaller the SdPER, the less the possibility of eliminating attributes.

-r Repeat (default = 10)

This option is used to tell the system the repeat times of construction of M-model trees for each subdata set. Appropriate numbers of repeats depend on data quality. The stronger the noise, the larger the Repeat. However, too large a Repeat does not help. Therefore, it is suggested that Repeat be less than 30.

-s SdTOL (default = 7)

This option is used to set the threshold of error measure. It tells the system when to stop data splitting. The lower the SdTOL, the less possibility of stopping data splitting. Consequently, the M-model tree will become large.

-v VERBOSITY (default = 0)

This option is used to control the output. The lower the VERBOSITY, the less detailed are the results that are output.

0 Output only the updated predictive models (planning factors and base quantities).

1 Output the best M-model trees (regions:equation pairs) of all subdata sets.

2 Output the M-model trees (regions:equation pairs) of each repeat after pruning.

3 Output the M-model trees (regions:equation pairs) of each repeat before pruning.

4 Output all the details of constructing M-model trees.

-t TestPER (default = 15)

This option controls what percentage of data is used for testing.

4.4. Running PFactor

Before running PFactor, the executable file "PFactor" and the input data files need to be copied to the same directory. The input data files have to have the format described in Section 4.2. The typed execution command is

PFactor -i trydata -v 1

After hitting "return", PFactor is invoked. It reads data from the file "trydata". The results are output to the current output channel, for instance, screen. Because VERBOSITY is 1, the best tree (actual region:equation pairs) of each subdata set will be output to the screen. The order of the options is not mandatory, therefore the command

PFactor -v 1 -i trydata

is the same as the last command. To direct the results to an output file, type

PFactor -i trydata -v 1 > tryout

Then, the system writes the results to the file "tryout".

Chapter 5

Performance of the system PFactor

Based on the algorithm discussed above, a system called PFactor was developed and implemented. This system was tested on both artificial data and real data. The performance of the system PFactor is discussed in this section.

5.1. Performance on Artificial Data Sets

1. First group of artificial data sets

The purpose of testing PFactor on the first group of data sets is to show how PFactor builds up M-model trees that represent the regularities underlying the data sets.

The first data set includes 100 examples. Each example has 10 independent variables x_1, ..., x_{10} and one dependent variable y. The goal is to find how the dependent variable y is determined by the independent variables x_1, ..., x_{10}. The data were generated from the following model by Matlab [Matlab, 1992]

Take x_1, ..., x_5 symbolic independent variables used in region descriptions. The discrete values of these variables are evenly distributed, i.e.

$$
\begin{aligned}
P(x_1 = Y) &= P(x_1 = N) = 1/2 \\
P(x_2 = T) &= P(x_2 = F) = 1/2 \\
P(x_3 = E) &= P(x_3 = W) = 1/2 \\
P(x_4 = U) &= P(x_4 = V) = P(x_4 = W) = 1/3 \\
P(x_5 = A) &= P(x_5 = B) = P(x_5 = C) = 1/3
\end{aligned}
\tag{5.1}
$$

Take x_6, ..., x_{10} numeric independent variables used in region equations. Let Z, introduced noise, be independent of x_1, x_2, ..., x_{10} and normally distributed with mean zero and variance 3. Then

$$
\begin{array}{ll}
\text{if } x_1 = Y \text{ and } x_2 = T & \text{set } y = 8 + 6\,x_6 + 4\,x_7 + Z \\
\text{if } x_1 = Y \text{ and } x_2 = F & \text{set } y = 12 + 10\,x_6 + 10\,x_7 + Z \\
\text{if } x_1 = N & \text{set } y = 7 + 5\,x_8 + Z
\end{array}
\tag{5.2}
$$

This model consists of three distinct regression equations with the choice of equations ranged by two symbolic variables x_1 and x_2.

As discussed in Chapter 4, PFactor provides users with some options to control how the system behaves in the learning process. Via the options, the following related parameters and thresholds are preset.

$$
\begin{aligned}
&\text{TestPER} = 15 \\
&\text{SdPER} = 15 \\
&\text{Repeat} = 1 \\
&\text{SdTOL} = 4
\end{aligned}
\tag{5.3}
$$

Now we discuss how the system PFactor builds up an M-model tree for the data set using these parameters and thresholds. The M-model tree generated by PFactor is shown in Fig. 5.1.

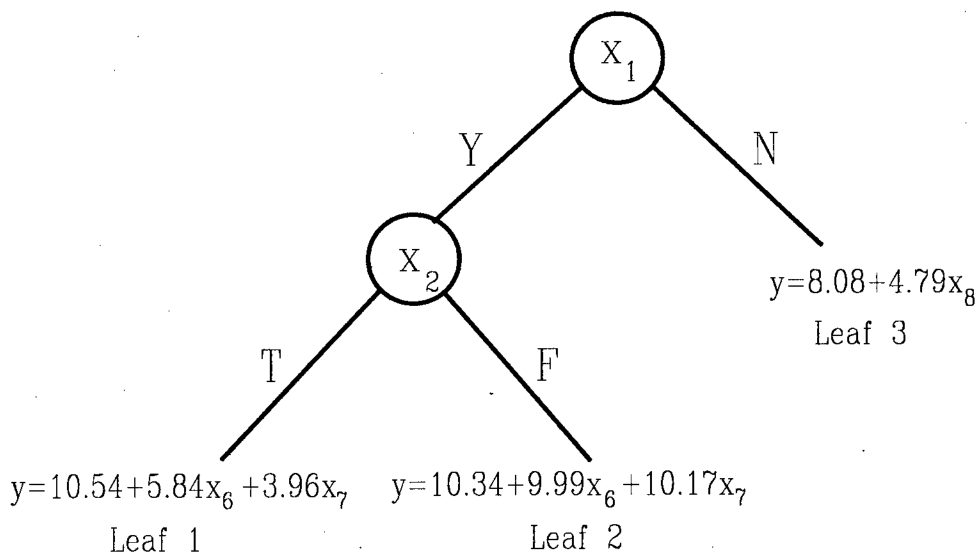

Fig. 5.1 The M-model tree of the first artificial data set when SdTOL = 4.

Before the beginning of building up the tree, the system randomly chooses 15 percent of the data (TestPER = 15) as the testing data, leaving 85 percent of the data as the training data. The system uses the 85 percent data to construct the M-model tree.

PFactor begins the construction of the tree with calculating the standard deviation. The standard deviation (76.19) is greater than the standard deviation tolerance SdTOL (5.00). Therefore, PFactor calculates the error reduction of all symbolic variables to find which symbolic variable should be used to split the data, in other words, which symbolic variable should be used in the root node of the M-model tree. The system finds x_1 gives maximum error reduction, so that x_1 is put in the root node.

Then, the algorithm calculates the weighted standard deviation. The weighted standard deviation without the elimination of any numeric attributes is 27.28. If x_6 is eliminated, the weighted standard deviation is 34.67. If x_7 is eliminated, the weighted standard deviation is 37.87. If x_8 is eliminated, the weighted standard deviation is 33.11. If x_9 is eliminated, the weighted standard deviation is 27.07. If x_{10} is eliminated, the weighted standard deviation is 27.05. Therefore, the variable x_{10} is the most effective elimination. Because the weighted standard deviation without x_{10} is less than that of all numerical attributes, x_{10} is eliminated. The weighted standard deviation of the node becomes 27.05. This process is repeated, and the system finds out the elimination of x_9 also reduce the weighted standard deviation so x_9 is eliminated. The elimination of the remaining numerical attributes does not reduce the weighted standard deviation. Therefore, only x_6 , x_7 and x_8 survive and only x_6 , x_7 and x_8 are inherited by the child nodes of the root node.

For the branch $< x_1 = Y >$ of the root node $< x_1 >$, PFactor again calculates the error reduction of all remaining symbolic variables and x_2 is selected. Calculating the weighted standard deviation, no numeric variable is eliminated therefore all the numeric variables x_6 , x_7 and x_8 are inherited by its child nodes. For the branch $< x_1 = Y$ and $x_2 = T >$ of the node $< x_2 >$, the standard deviation is less than the threshold of standard deviation SdTOL, therefore the branch < x_1 = Y and x_2 = T > becomes leaf 1. Leaves 2 and 3 become leaves also because their standard deviation is less than the standard deviation threshold SdTOL.

At the leaf level, no weighted standard deviation can be obtained, so numeric variables are further eliminated only when their elimination does not significantly influence the standard deviation. The significance of the influence is defined by a threshold (SdPER). In other words, numeric variables are further eliminated only when their elimination does not influence the standard deviation within the threshold (SdPER). For example, in the leaf 1, the standard deviation of x_6 , x_7 and x_8 is 3.67. If x_6 is eliminated, the standard deviation of x_7 and x_8 is 39.41; if x_7 is eliminated, the standard deviation of x_6 and x_8 is 17.08; if x_8 is eliminated, the standard deviation of x_6 and x_7 is 3.82. Among those three possible eliminations, the elimination of x_8 gives the least standard deviation so the x_8 is considered as the most effective elimination. Then, the algorithm finds that the elimination of the variable x_8 influences the standard deviation within the preset threshold SdPER of 15 percent, so that the variable x_8 is eliminated and only the variables x_6 , x_7 remain. Now, the standard deviation of the left becomes 3.82. Next, the system checks the possible eliminations again. If x_6 is eliminated, the standard deviation of x_7 is 38.76; if x_7 is eliminated, the standard deviation of x_6 is 16.95. Among the two possible eliminations, the elimination of x_7 gives the least standard deviation so x_7 is considered as the most effective elimination. Comparing the standard deviation of x_6 with the standard deviation of x_7, the elimination will increase the standard deviation out of the 15 percent range. Therefore, no variables can be eliminated. In this leaf, x_6 and x_7 are left to build the linear model.

Looking at the standard deviations with x_8 (3.68) and without x_8 (3.82) respectively, x_8 does not give much contribution to the standard deviation even though the standard deviation with x_8 is smaller. Therefore, a threshold SdPER is set up. If the elimination of numeric variables does not influence the standard deviation within the threshold range, the numeric variable is eliminated. PFactor provides users with an option to set SdPER. The value of SdPER is based on users' understanding of domain knowledge and data quality. The default value of SdPER is 15 percent.

After building the tree, the algorithm tries to prune the tree and shows it is unnecessary to prune the tree. Therefore, the tree in Fig. 5.1 is taken as the final tree and the responding region - equation pairs are obtained.

$$\begin{array}{lll} \text{if } x_1 = \mathrm{Y} \text{ and } x_2 = \mathrm{T} & y = 8.82 + 5.98\,x_6 + 3.81\,x_7 & \\ \text{if } x_1 = \mathrm{Y} \text{ and } x_2 = \mathrm{F} & y = 11.27 + 10.12\,x_6 + 10.03\,x_7 & (5.4) \\ \text{if } x_1 = \mathrm{N} & y = 8.27 + 4.71\,x_8 & \end{array}$$

Comparing Eq. (5.4) with the known function Eq. (5.2), it is noticed that Eq. (5.4) is close to the known function. It is understandable that the system does not get the exact same function as Eq. (5.2) due to the introduction of noise in the data.

Next, we discuss the parameter "Repeat". Via the option **-r** , PFactor is run on the same data set again when "Repeat" is preset as 5. The learning results are quite close. For example, another set of region-equation pairs is

$$\begin{array}{lll} \text{if } x_1 = \mathrm{Y} \text{ and } x_2 = \mathrm{T} & y = 8.62 + 5.94\,x_6 + 3.92\,x_7 & \\ \text{if } x_1 = \mathrm{Y} \text{ and } x_2 = \mathrm{F} & y = 10.92 + 10.10\,x_6 + 10.07\,x_7 & (5.5) \\ \text{if } x_1 = \mathrm{N} & y = 8.42 + 4.69\,x_8 & \end{array}$$

Compared with the region-equation pairs in Eq. (5.4), no obvious difference is observed. The reason is that this artificial data has high quality and low noise (variance of 3). The repeat times of the learning process does not strongly influence the results. High quality data sets do not need the value of "Repeat" to be set very high. For the sake of obtaining satisfactory results, the default value of "Repeat" is set as 5.

Now, we discuss the influence of the threshold SdTOL on the learning process. The variance of noise introduced in the data set is 3, and standard deviation tolerance SdTOL is preset as 4, little greater than the noise variance. We get the right tree (Fig. 5.1) without pruning the tree. Assume the variance of noise is unknown and SdTOL is preset as 2. The tree obtained when SdTOL=2 is larger than that when SdTOL is set as 4. The tree is shown in Fig. 5.2.

After the growth of the tree, the system starts the process of pruning the tree. The process of tree pruning means eliminating the symbolic variables that do not significantly contribute to the reduction of error estimate on testing data. Rather than compare the values of error estimates of a node and its subtree, the system compares the improvement of error estimate. If the improvement of error estimate of the subtree is lower than a certain threshold, the subtree should be pruned. This threshold is set the same as the SdPER. Now we see how the tree in Fig. 5.2 is pruned back to the tree in Fig. 5.1.

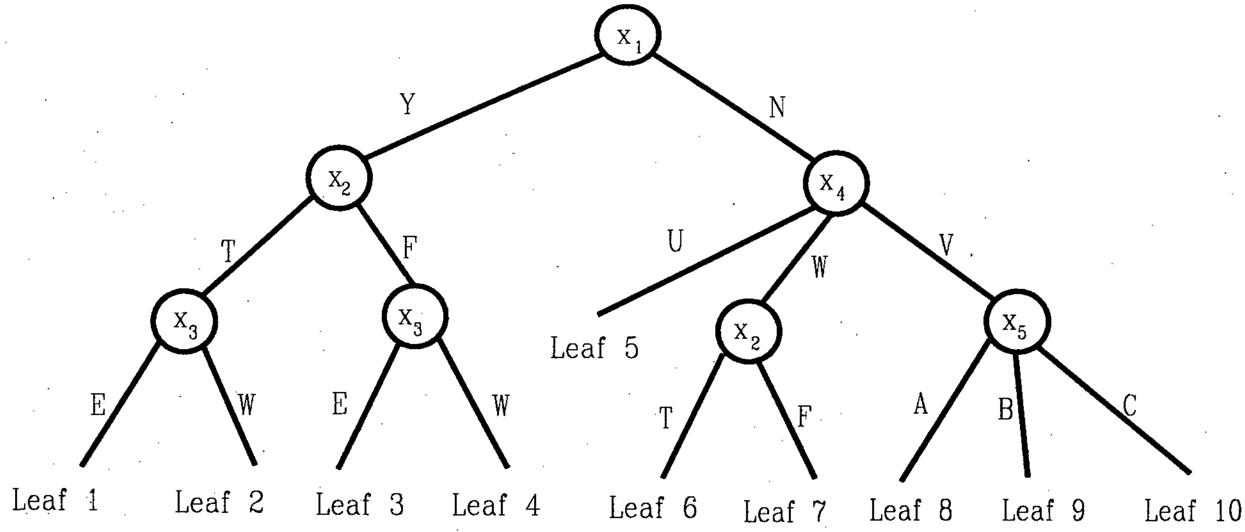

Fig. 5.2 The M-model tree of the first artificial data set when SdTOL = 2.

The pruning of the tree begins near the leaves of the tree. First, it checks whether the node $< x_3 >$ of $< x_1 = \text{Y}$ and $x_2 = \text{T} >$ should be pruned. The error estimate of the node (the average of percentage deviation on the testing data) is 2.97 and the error estimate of its subtree is 4.81. Its subtree (leaves 1 and 2) should be substituted by the $< x_3 >$ of $< x_1 = \text{Y}$ and $x_2 = \text{T} >$. Therefore the node $< x_3 >$ of $< x_1 = \text{Y}$ and $x_2 = \text{T} >$ becomes a leaf. The system checks all non-leaf nodes of the tree. The node $< x_3 >$ of $< x_1 = \text{Y}$ and $x_2 = \text{F} >$, the node $< x_5 >$ of $< x_1 = \text{N}$ and $x_4 = \text{V} >$ and the node $< x_2 >$ of $< x_1 = \text{N} >$ should be pruned, and the nodes become leaves. The tree after pruning is the same as the tree in Fig. 5.1, therefore the obtained region:equation pairs are the same. In some cases, the error estimate of the node is little larger than that of its subtree. In such cases, the subtree is also pruned as long as the error estimate of its subtree is not improved within the threshold. This threshold is set the same as the threshold SdPER.

To focus on the influence of data quantities on the learning results, a second artificial data set is generated using the same model of Eq.(5.1) and Eq. (5.2). But the second artificial data includes 400 examples. The learning results are

$$
\begin{array}{ll}
\text{if } x_1 = \text{Y and } x_2 = \text{T} & y = 8.07 + 6.00\,x_6 + 3.97\,x_7 \\
\text{if } x_1 = \text{Y and } x_2 = \text{F} & y = 12.07 + 10.04\,x_6 + 9.98\,x_7 \\
\text{if } x_1 = \text{N} & y = 6.89 + 5.03\,x_8
\end{array}
\qquad (5.6)
$$

when the parameters and thresholds have their default values. The region:equation pairs in Eq. (5.6) are closer to the known model than those in Eqs (5.4) and (5.5), showing that the larger the data set, the better the results.

To focus on the influence of noise on the learning results, a third artificial data set of 100 examples is generated by the same model as Eq. (5.1) and Eq. (5.2). But the third artificial data set includes the introduced noise Z of variance 10. The PFactor generates the following results when the running parameters and thresholds are the default values:

$$
\begin{array}{ll}
\text{if } x_1 = \text{Y and } x_2 = \text{T} & y = 12.47 + 5.87\,x_6 + 3.57\,x_7 \\
\text{if } x_1 = \text{Y and } x_2 = \text{F} & y = 8.91 + 9.15\,x_6 + 9.88\,x_7 + 1.93\,x_8 \\
\text{if } x_1 = \text{N} & y = 8.05 + 4.63\,x_8
\end{array}
\qquad (5.7)
$$

The generated functions are not close to the known function. Strong noise can result in the generated function not matching the known function. However, more data examples can compensate for the existence of data noise. The fourth artificial data set is generated, the same as the third artificial data set except for more examples (400). The generated functions are

$$\begin{array}{ll} \text{if } x_1 = \text{Y and } x_2 = \text{T} & y = 7.35 + 6.00\,x_6 + 4.06\,x_7 \\ \text{if } x_1 = \text{Y and } x_2 = \text{F} & y = 13.64 + 9.94\,x_6 + 10.04\,x_7 \\ \text{if } x_1 = \text{N} & y = 7.79 + 4.90\,x_8 \end{array} \tag{5.8}$$

The generated function is close to the known function, showing that more data points compensate for the noise to enable generation of correct results.

2. Second group of artificial data sets

The second group of artificial data sets are generated to test PFactor's whole learning process applied to finding planning factors and base quantities.

An artificial data set is generated to simulate the real data base. The influence of the parameters and threshold on building an M-model tree is discussed in the proceeding section on the first group of artificial data sets. Those parameters and thresholds are not discussed in this example. The discussion is focused on the threshold "ConstTOL" that controls the comparison of equations Eq. (3.8) and trees Eq. (3.11). This controls how many different planning factors are generated.

The data set is divided into 3 subdata sets, each of which consists of 400 examples. Every example has 10 independent variables x_1, ..., x_{10} and one dependent variable y. The data were generated from the following models by Matlab [Matlab, 1992]

Take x_1, ..., x_5 symbolic independent variables used in region descriptions. The discrete values of these variable are evenly distributed in the same way as shown in Eq. (5.1)

Take x_6, ..., x_{10} numeric independent variables used in region equations. Let Z, the introduced noise, be independent of x_1, x_2, ..., x_{10} and normally distributed with mean zero and variance 3. Then

The first subdata set:

$$\begin{array}{ll} \text{if } x_1 = \text{Y and } x_2 = \text{T} & \text{set } y = 5 + 8\,x_6 + 20x_7 + Z \\ \text{if } x_1 = \text{Y and } x_2 = \text{F} & \text{set } y = 6 + 4\,x_6 + 10\,x_7 + Z \\ \text{if } x_1 = \text{N} & \text{set } y = 5 + 8\,x_8 + Z \end{array} \tag{5.9}$$

The second subdata set:

$$\begin{array}{lll} \text{if } x_1 = \text{Y and } x_2 = \text{Tset } y = 10 + 16\,x_6 + 40\,x_7 + Z & & = 2\,(5 + 8\,x_6 + 20x_7\,) + Z \\ \text{if } x_1 = \text{Y and } x_2 = \text{F} & \text{set } y = 12 + 8\,x_6 + 20\,x_7 + Z & = 2\,(6 + 4\,x_6 + 10\,x_7) + Z \\ \text{if } x_1 = \text{N} & \text{set } y = 10 + 16\,x_8 + Z & = 2\,(5 + 8\,x_8\,) + Z \end{array} \tag{5.10}$$

The third subdata set is the same as shown in Eq. (5.2).

According to the known models in Eqs (5.1), (5.9), (5,10), and (5.12), the subdata sets 1 and 2 behave similarly. In other words, they can be described by the same planning factor in terms of their own base quantities. If the equations Eq. (5.9) of the first subdata set is selected, the second subdata set should be described by the equations Eq. (5.9) multiplied by the constant 2. That is to say, if Eq. (5.9) is taken as their common planning factor, subset 1 should have base quantity of 1 and subset 2 should have base quantity of 2.

The parameters and thresholds in the learning process is the same as Eq. (5.3) plus the threshold ConstTOL. Using these parameters and thresholds, PFactor builds up an M-model tree for each of subdata sets. The region:equation pairs corresponding to the M-model trees are listed below:

The first subdata set:

$$\begin{array}{ll} \text{if } x_1 = \text{Y and } x_2 = \text{T} & y = 4.88 + 8.01\,x_6 + 20.01\,x_7 \\ \text{if } x_1 = \text{Y and } x_2 = \text{F} & y = 5.68 + 4.02\,x_6 + 10.02\,x_7 \\ \text{if } x_1 = \text{N} & y = 4.58 + 8.11\,x_8 \end{array} \tag{5.11}$$

The second subdata set:

$$\begin{array}{ll} \text{if } x_1 = \text{Y and } x_2 = \text{T} & y = 10.25 + 15.92\,x_6 + 40.03\,x_7 \\ \text{if } x_1 = \text{Y and } x_2 = \text{F} & y = 13.02 + 7.93\,x_6 + 19.96\,x_7 \\ \text{if } x_1 = \text{N} & y = 10.13 + 15.98\,x_8 \end{array} \tag{5.12}$$

The third subdata set:

$$
\begin{array}{lll}
\text{if } x_1 = \text{Y and } x_2 = \text{T} & y = \ \ 7.98 + 5.94\,x_6 + \ 4.02\,x_7 & \\
\text{if } x_1 = \text{Y and } x_2 = \text{F} & y = 11.96 + 9.96\,x_6 + 10.02\,x_7 & (5.13) \\
\text{if } x_1 = \text{N} & y = \ \ 6.63 + 5.01\,x_8 &
\end{array}
$$

When ConstTOL is set as 0.10, the results of planning factors and base quantities are shown as following

There are three planning factors

The 1st planning factor is

$$
\begin{array}{lll}
\text{if } x_1 = \text{Y and } x_2 = \text{T} & y = 4.88 + 8.01\,x_6 + 20.01\,x_7 & \\
\text{if } x_1 = \text{Y and } x_2 = \text{F} & y = 5.68 + 4.02\,x_6 + 10.02\,x_7 & (5.14) \\
\text{if } x_1 = \text{N} & y = 4.58 + 8.11\,x_8 &
\end{array}
$$

The 2nd planning factor is

$$
\begin{array}{lll}
\text{if } x_1 = \text{Y and } x_2 = \text{T} & y = 10.25 + 15.92\,x_6 + 40.03\,x_7 & \\
\text{if } x_1 = \text{Y and } x_2 = \text{F} & y = 13.02 + \ \ 7.93\,x_6 + 19.96\,x_7 & (5.15) \\
\text{if } x_1 = \text{N} & y = 10.13 + 15.98\,x_8 &
\end{array}
$$

The 3rd planning factor is

$$\begin{array}{lll} \text{if } x_1 = \text{Y and } x_2 = \text{T} & y = 7.98 + 5.94\,x_6 + 4.02\,x_7 & \\ \text{if } x_1 = \text{Y and } x_2 = \text{F} & y = 11.96 + 9.96\,x_6 + 10.02\,x_7 & (5.16) \\ \text{if } x_1 = \text{N} & y = 6.63 + 5.01\,x_8 & \end{array}$$

The planning factors and base quantities of subdata sets are

The 1st subdata set :	The 1st planning factor;	Base quantity = 1.00
The 2nd subdata set:	The 2nd planning factor,	Base quantity = 1.00
The 3rd subdata set:	The 3rd planning factor,	Base quantity = 1.00

For the subdata sets 1 and 2, the coefficients in the equations of the region $< x_1 = \text{N} >$ do not satisfy Eq. (3.8) when ConstTOL is 0.10. Thus the equations in the region are not proportional. Consequently, they can not be described by the same planning factor. That is why they have their own planning factors. But when ConstTOL is 0.15, the results of planning factors and base quantities are different. They are

There are two planning factors :

The 1st planning factor is

$$\begin{array}{lll} \text{if } x_1 = \text{Y and } x_2 = \text{T} & y = 5.02 + 8.01\,x_6 + 20.08\,x_7 & \\ \text{if } x_1 = \text{Y and } x_2 = \text{F} & y = 6.11 + 4.00\,x_6 + 10.03\,x_7 & (5.17) \\ \text{if } x_1 = \text{N} & y = 4.84 + 8.07\,x_8 & \end{array}$$

The 2nd planning factor is

$$
\begin{array}{lll}
\text{if } x_1 = \text{Y and } x_2 = \text{T} & y = & 7.98 + 5.94\,x_6 + 4.02\,x_7 \\
\text{if } x_1 = \text{Y and } x_2 = \text{F} & y = & 11.96 + 9.96\,x_6 + 10.02\,x_7 \\
\text{if } x_1 = \text{N} & y = & 6.63 + 5.01\,x_8
\end{array} \tag{5.18}
$$

The planning factors and base quantities of subdata sets are

The 1st subdata set :	The 1st planning factor;	Base quantity = 1.00
The 2nd subdata set:	The 1st planning factor,	Base quantity = 1.99
The 3rd subdata set:	The 2nd planning factor,	Base quantity = 1.00

It shows that the threshold ConstTOL is critical in the comparison of the regularities in two subdata sets. The larger the threshold ConstTOL, the greater the possibility that the two subdata sets can described by the same planning factor by introduction of their own base quantities. Therefore, it is not a surprise that the same data set can generate different results.

5.2. Performance on Real Data Sets: Planning Factors P0016, P0018, and P0020

The purpose of building the system PFactor is to update the predictive models of planning factors and base quantities used by KDOT. The system has to run on real data sets. From the discussion in Section 5.1, it is known that the system works well for artificial data sets. In this section, we discuss the performance of PFactor on real data sets.

There are 65 planning factors and they are updated one-by-one. Therefore there are 65 files, each of which is used to update one planning factor. Each file contains the data records related with the same current planning factor. The records of Functional Units that may come from different Activities of different templates may be collected in the same data file as long as their durations are predicted by the same current planning factor. The data are drawn from both management systems CPMS and RMS.

A file is used as input for the system PFactor to update the corresponding planning factor. As discussed in Chapter 2, the learning starts at the level of Functional Units, therefore the subsets should be divided by each particular Functional Unit, that is to say, each Functional Unit should have an M-model tree to describe the regularity between the Functional Unit duration and the attributes. However, there may not be enough data for a particular Functional Unit. For instance, in data file "cprms16.txt" (used in updating planning factor P0016), the Functional Unit CONRD has only 7 examples, and the Functional Unit ENVIR has only 4 examples. Instead of giving up, the learning on real data sets starts based on the assumption that Functional Units originally described by the same planning factor and the same base quantity behave the same. This assumption means that the subdata sets should be grouped by the current base quantity of Functional Units rather than by each Functional Unit.

Here, we discuss the performance of PFactor when it runs on data file “cprms16.txt” to modify the planning factor P0016. The choice of the planning factor to test the system PFactor is random. The data file "cprms16.txt" includes 189 examples from the management system CPMS and 127 examples from the management system RMS. When the data set is generated from the master project data base, domain engineers exclude the attributes irrelevant to duration using the domain knowledge. In this case, 9 independent attributes are left. Five of the attributes are used in region descriptions and four of them are used in region equations:

Description: <US_81>, <Lanes>, <Urban_ind> and <Util_reloc>

Equation : <Length>, <Bridges>, <Tracts>, <Tracts_condem>, and <Tracts_reloc>

All description attributes are symbolic and all equation attributes are numeric. When the system runs on the data set, all parameters and thresholds use their default values except Repeat = 30. The updated planning factor generated from "cprms16.txt" is

$$p.f. = 75.98 \quad \text{(standard deviation = 77.65)} \tag{5.19}$$

The average of percentage deviation by the current planning factor and updated planning factor are 1.52 and 1.52 respectively. Therefore, there is no improvement given by the updated planning factor. The standard deviation has the same unit as duration. Here, the duration had unit "day", therefore the standard deviation has unit "day" too. In fact, the standard deviation (77.65) of the regression equation in Eq. (5.19) is very large. It represents the extreme scatter of the data points as shown in Fig. 5.3.

The generated planning factor is not satisfactory. The system PFactor runs on the data file "cpms16.txt", which excludes the data from the management system RMS. The updated planning factor generated from "cpms16.txt" is

if <US81_ind>=E and <Util_reloc>=N, p.f. = 88.25

if <US81_ind>=E and <Util_reloc>=Y, p.f. = 50.23 + 2.01*<Tract> (5.20)

if <US81_ind>=W, p.f = 113.29

where the standard deviations of regions (<US81_ind>=E and <Util_reloc>=N), (<US81_ind>=E and <Util_reloc>=Y), and (<US81_ind>=W) in Eq. (5.20) are 74.94, 54.10 and 115.39 respectively; the average of percentage deviation by the current planning factor and updated planning factor are 1.12 and 0.96 respectively. The improvement rate is 14.29%.

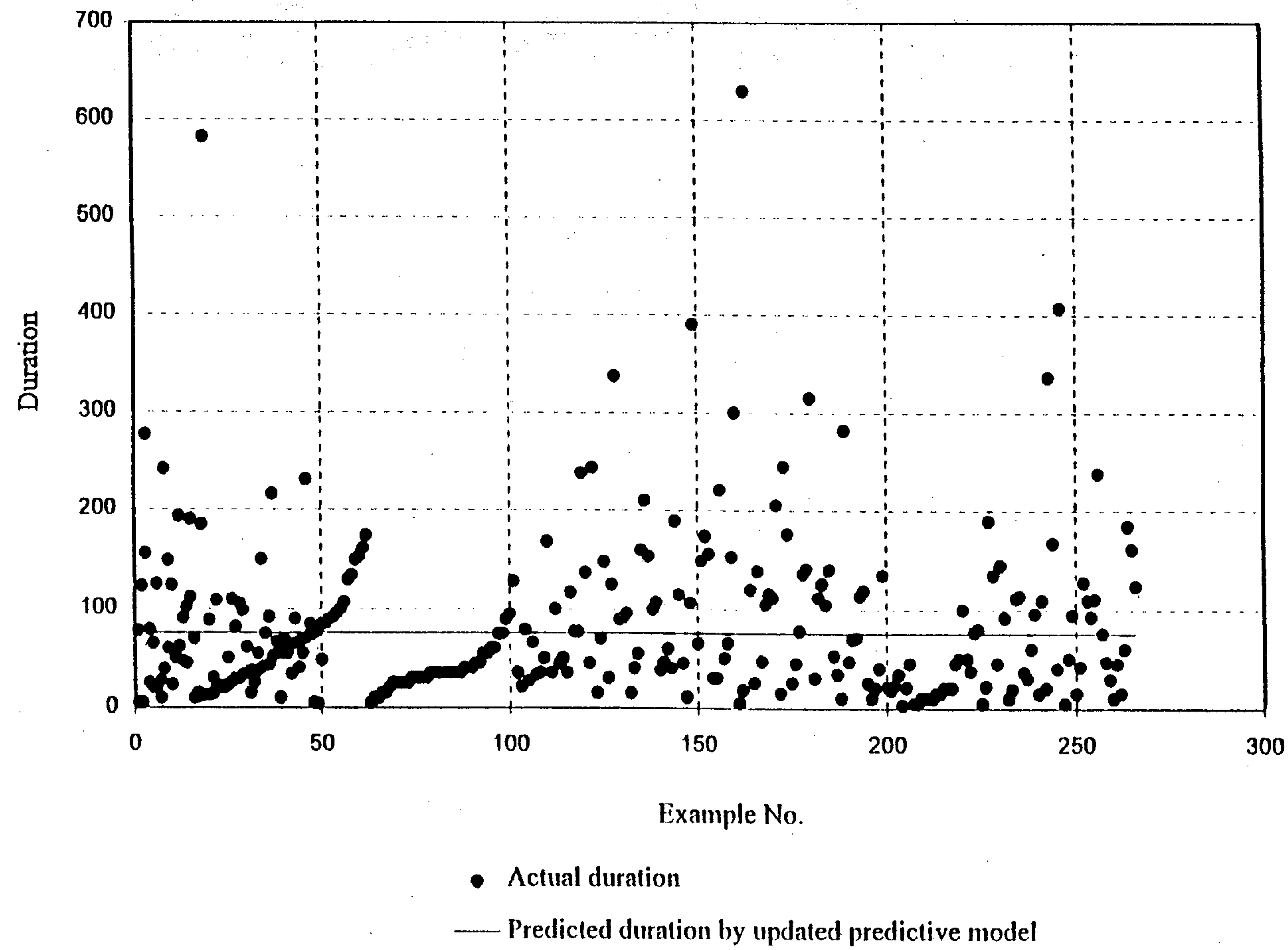

Fig. 5.3 Example data and Eq. (5.19).

According to the average percentage deviations of the testing data, data file "cpms16.txt" can generate an updated planning factor which predicts durations on testing data better than the current planning factor **based on the current data quality**. The current data quality requires that users be careful when using the updated planning factor for the following reasons.

- First, we look at the percentage deviation given by both current and updated planning factors. It shows that the percentage deviation by the updated planning factor improves by 14.29%. However, the percentage deviation by the updated planning factor itself is 96%, quite high. The 14% improvement of percentage deviation of updated planning factor is obtained **comparing with the current planning factor**.
- Second, we look at the standard deviation of regions in Eq. (5.20), which are 74.94, 54.10 and 115.39 respectively. The standard deviations in Eq. (5.20) are very large. The example data and Eq. (5.20) are shown in Figs. (5.4) to (5.6). Using regression equations with such high standard deviations must be done cautiously.
- Third, we notice that there are 4 description attributes but only 2 are used, even though the standard deviations are high. Here, we check the process of building the M-model tree corresponding to region:equation pairs. The M-model tree built up by the system before pruning is shown in Fig. 5.7.

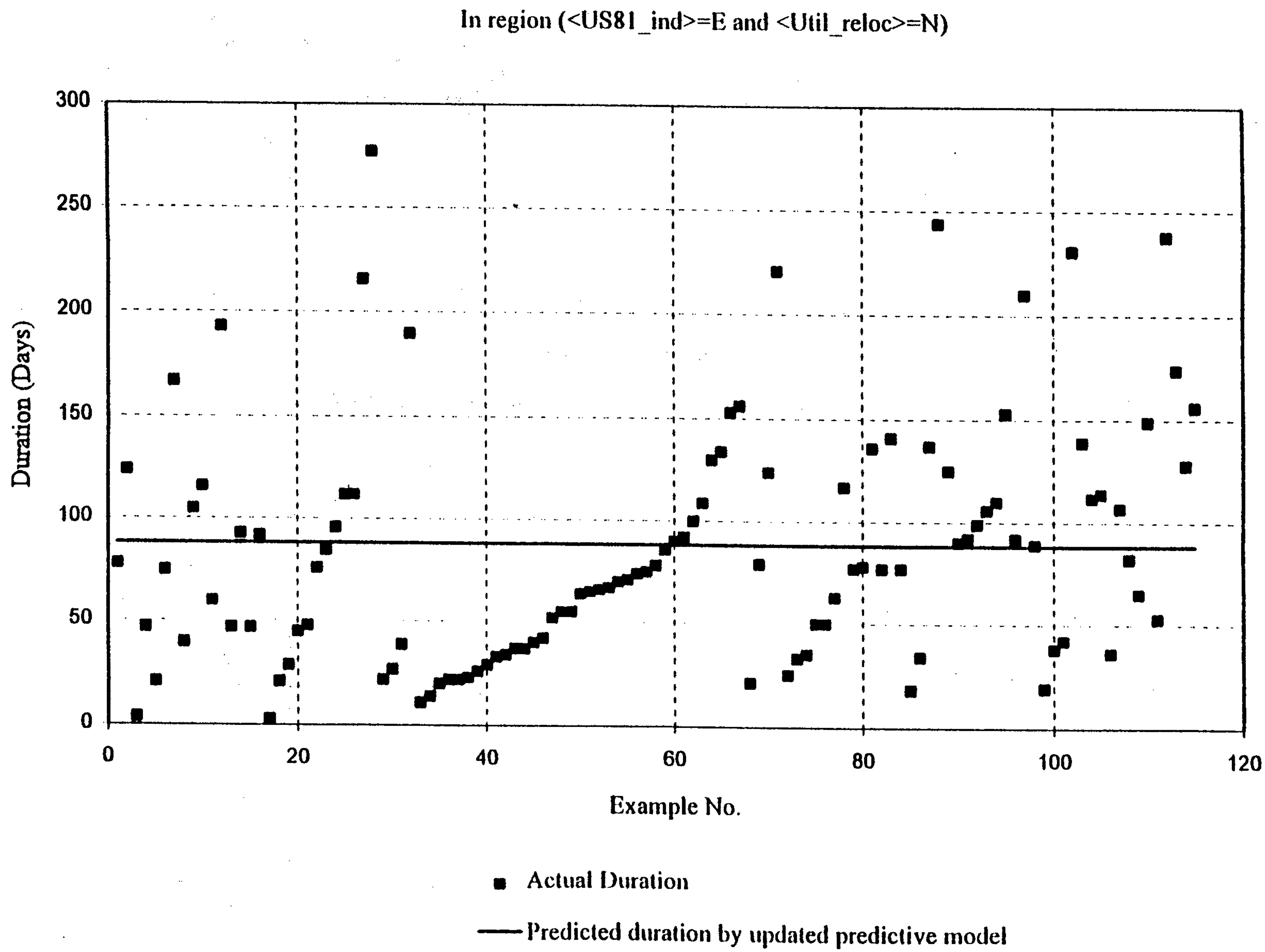

Fig. 5.4 Example data in region (<US81_ind>=E and <Util_reloc>=N) of Eq. (5.20)

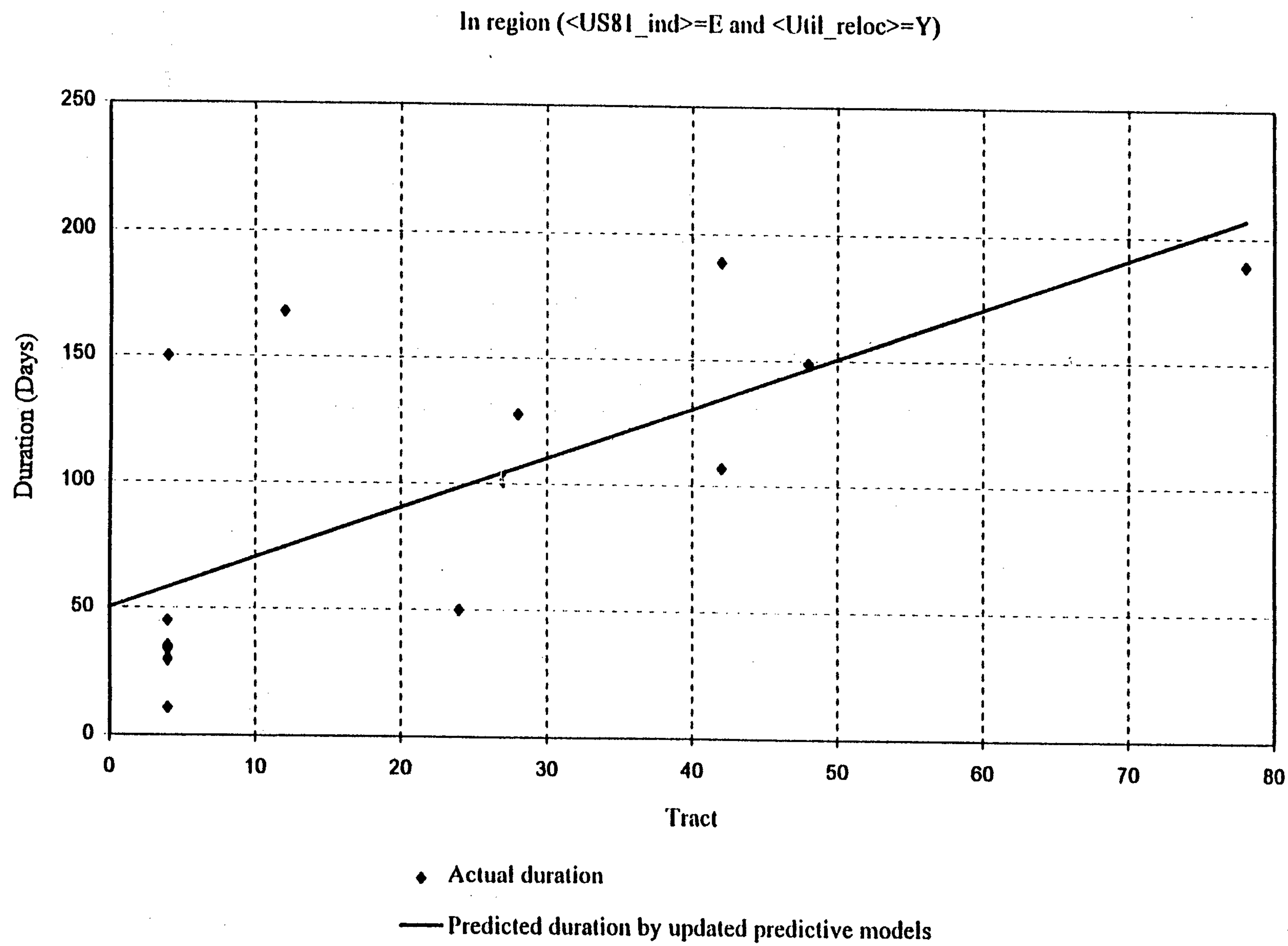

Fig. 5.5 Example data in region (<US81_ind>=E and <Util_reloc>=Y) of Eq. (5.20)

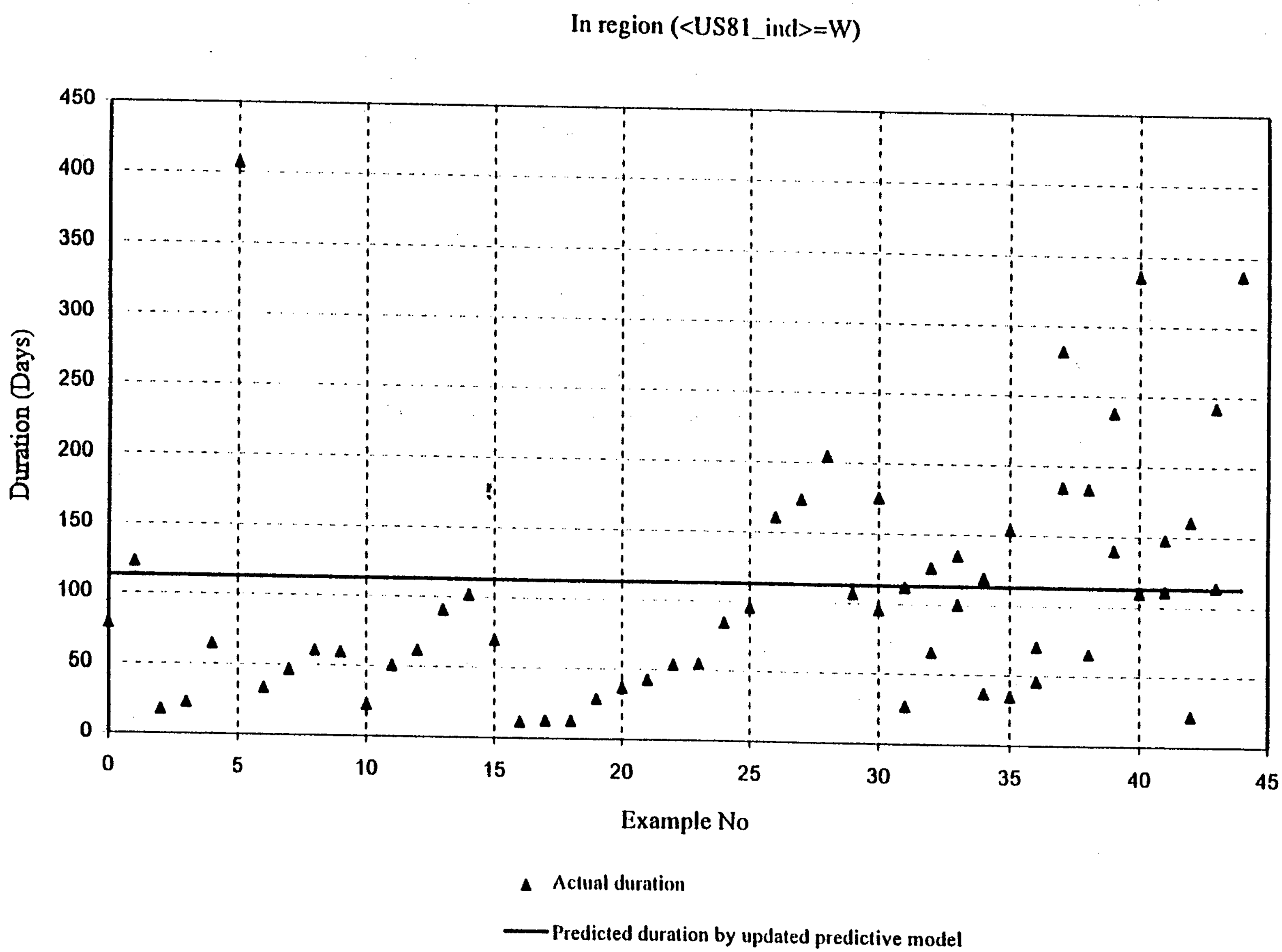

Fig. 5.6 Example data in region (<US81_ind>=W) of Eq. (5.20).

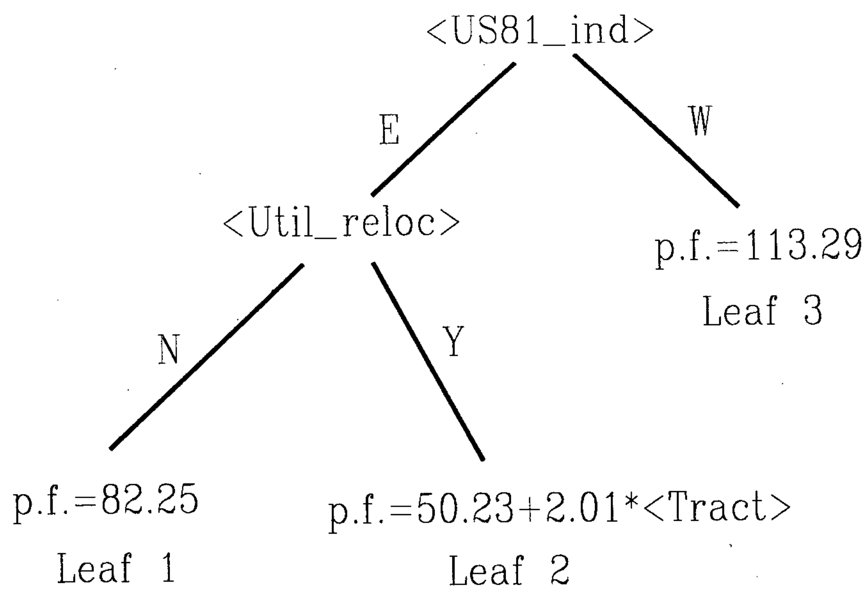

Fig. 5.7 M-model tree built for real data set.

The nodes become leaves not because the standard deviations are smaller than the threshold 7 but because no remaining description attributes can be used to split the data. For instance, before the node becomes leaf 1, it has one equation attribute it inherited from its mother node. The remaining description attributes are <Lane> and <Urban_ind>. Its minimum number of examples is 6. If the attribute <Lane> is used, one of its branches contains only 3 examples. Therefore, it can not be used. If the attribute <Urban_ind> is used, one of its branches contains only 4 examples. Therefore, it can not be used. The node stops growing and becomes leaf 1. The same situation happens at the other leaves. Lack of data prevents all attributes from being used in building up the M-model tree.

Two other planning factors are selected to test the system: P0018 and P0020. The information on the examples is listed in Table 5.1.

Table 5.1 Information on the files.

Planning factor	Examples from CPMS	Examples from RMS	Total examples	No of base quantities
P0018	179	124	303	1
P0020	211	114	325	1

The tests on the system show that using the data from both CPMS and RMS could not obtain updated planning factors that give better prediction on durations. They also show that using the data only from CPMS can obtain updated planning factor that gives better prediction on duration based on the current data quality. But, the system generates updated planning factors with high standard deviations because noise exists in the data sets as shown in Figs. (5.3) to (5.6). Therefore, users must carefully consider the application of the updated planning factors using domain knowledge.

5.3 Data quality issues

The quality of updated planning factors is determined by data quantity and data quality. That is to say, on the one hand, high quality of updated planning factors requires large data sets for learning; on the other hand, high quality of updated planning factors requires high quality of data sets, which means the noise in data sets is low.

The updated planning factors generated by the system PFactor from real data sets have very large standard deviations although the updated planning factors give better predictions when compared with the current planning factors. The percentage deviation of the updated planning factors are still high. They indicate the quality of updated planning factors is not high. The problems come from the following reasons:

1) Although the master data base is large, examples are not enough for a particular Functional Unit. This prevents some attributes from being used in building the M-model tree for Functional Units. The test on data file "cpms16.txt" displays such a shortage of examples. For instance, if an attribute indeed has significant influence on duration for a data set, a shortage of examples does not allow the attribute to be used in building up the M-model tree. The given information may not be fully exploited.

2) The way of drawing data from the master data base may bring noise into the data sets. The "actual" duration is obtained from the data base by the following calculation

$$duration = \text{end\_date} - \text{start\_date} \tag{5.21}$$

If a Functional Unit was suspended for some time, the calculation does not exclude the time when no work was done on the Functional Unit. This results in the duration of the Functional Unit used to update the models being greater than the ACTUAL duration.

3) The duration's unit may not be consistent. The more people working on a project, the shorter the duration. Duration obtained from the calculation Eq. (5.2) does not indicate how many people work on Functional Units of a project. When different numbers of people

work on a Functional Unit, the Functional Unit will have different durations based on the method of drawing data Eq. (5.21). Inconsistent duration units certainly bring some noise to data sets.

4) To overcome the shortage of examples for each Functional Units, it is assumed that Functional Units originally described by the same planning factor and the same base quantities behave the same. If the assumption is not appropriate, the assumption brings noise into the data set.

To obtain high quality of updated planning factors, shortage of data and low quality of data sets have to be overcome. At present, a large number of data could not be collected immediately because of the long period of transportation projects. Improvement of data quality should be focused at the current stage.

Noise of data sets, to a great extent, do not come from the outliers as discussed in statistics. In fact, no outliers can be identified due to extreme scatter of the data sets as shown in Figs. (5.3) to (5.6). Improving data quality should be focused on those aspects discussed in the first part of this section. The methods of reducing noise mainly include the following:

1) Duration of Functional Units should be continuous and nonfragmented. If duration of Functional Units includes time when no work is done, that part of time should be subtracted from duration of the Functional Units.

2) The units of Functional Units' duration are kept consistent. Duration unit should use "days/person" instead of "days".

3) A third issue, raised in Section 2.4 is the use of estimated durations in place of actual

durations in CPMS examples. This approach should be reevaluated and the possibility of extracting true actual durations from cost center feedback CCFB data should be explored.

Chapter 6

Conclusions

1. Combining machine learning techniques and regression analysis has a very promising application in engineering problems. Complicated engineering problems can be solved by means of machine learning.
2. PFactor is a knowledge based machine learning system. It can induce nonhomogeneous mathematical functions between a targeted variable/attribute and many other variables/attributes from data sets, whose variables/attributes are of mixed types (symbolic and numeric) and significant variables/attributes are unknown before data analysis. It can update the predictive models of planning factors and base quantities used in KDOT project management system.
3. The performance of system PFactor is very good on artificial data sets including a degree of noise. Experiments show that increasing the number of examples can compensate for the noise existing in data sets.
4. The performance of system PFactor is not very good on actual KDOT provided data sets due to extreme noisiness of these data sets and the small quantity of data examples. The low quality of input data results in low quality of updated predictive models. To obtain a higher quality of updated predictive models, collecting more data and improving data quality is required.

5. To improve the quality of data sets, the data must be preprocessed and cleaned. On the one hand, durations of Functional Units should be continuous and nonfragmented; on the other hand, the units of duration should be consistent. In addition, actual durations are needed for CPMS examples.

References

Activity networks of templates, KDOT internal document.

Bratko, I. and Konomenko, I. (1987), Learning Diagnostic Rules from Incomplete and Noisy Data, *Interaction in Artificial Intelligence and Statistical Methods*, ed. B. Phelps. Aldershot, England: Technical.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984), *Classification and Regress Tree*, Wadsworth, Belmont CA.

Cohen, P.R. and Feigenbaum, E.A. (1982), *The Handbook of Artificial Intelligence*, vol 3, William Kaufmann, Inc..

Cover, T.M. and Hart P.E. (1967). Nearest Neighbor Pattern Classification. *IEEE Transaction of Information Theory*, 13, 21-27.

Dasarathy, B.V. (ed) (1991). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.* Los Alamitos, CA: IEEE Computer Society Press.

Falkenhainer, B.C. and Michalski, R.S. (1986), Integrating Quantitative and Qualitative Discovery: The ABACUS System. *Machine Learning,* 1(4).

Forsooth, R. (1989), *Machine Learning, Principles and Techniques*, Chapter 4, Chapman and Hall

Company.

Freeman, J.A. and Skapura, D.M. (1991). *Neural Networks: Algorithms, Applications, and Programming Techniques.* Addison-Wesley, Reading, MA.

Langley, P. and Iba, W. (1993). Average-Case Analysis of a Nearest Neighbor Algorithm, *Proc. of 13th Int. Joint Conf. on Artificial Intelligence*, vol 2.

Langley, P., Simon, H.A., Bradshaw, G.L. and Zytlow, J.M (1987), *Scientific Discovery, Computational Explorations of the Creative Processes*, The MIT Press.

Niblett, T. and Bratko, I. (1987), Learning Decision Rules in Noisy Domains, *Research and Development in Expert Systems III*, ed M.A. Bramer, Cambridge: Cambridge University Press.

Planning value tables. KDOT internal document.

Press, W.H., S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, *Numerical Recipes in C*, Cambridge University Press, 1992.

Project templates. KDOT internal document.

Quinlan, R.J. (1983), Learning Efficient Classification Procedures and Their Application to Chess End Game, *Machine Learning: The Artificial Intelligence Approach*, eds R.S. Michalski, J.G. Carbonell and T. Mitchell, Tioga Press, Palo Alto, pp463-82.

Quinlan, R.J. (1992), Learning with Continuous Classes, *Proceedings AI'92*, eds Adams and Sterling.

Quinlan, R.J. (1994), *C4.5 : Programs for Machine Learning*, Morgan Kaufmann Publisher Inc.

Schaffer, C. (1990), *Domain-Independent Scientific Function Finding*, Ph.D. Dissertation, Rutgers University.

Schlimmer, J.C. and Fisher, D. (1986), A Case Study of Incremental Concept Induction Science, *Proc. of the 5th National Conference on AI*, San Mateo, CA, Morgan Kaufmann.

Skapura, D.M., *Building Neural Networks,* ACM Press, 1996.

Stanfill, C. and Waltz, D. (1986), Toward Memory-Based Reasoning. *Communications of the ACM, 29*, 1213-1228.

Stella, P.J. and Glavinich, T.E. (1994), *Construction Planning and Scheduling*, AGC of America.

Turing, A. (1950), Computing machinery and intelligence, *Mind*, (59).

Utgoff, P.E. (1988), ID5: An Incremental ID3, *Proc. of the 5th Inter. Conf. on Machine Learning*, Morgan Kaufmann Publisher Inc.

Wu, Y.H. and Wang, S. (1991), Discovering Functional Relationships from Observations Data, *Knowledge Discovery in Databases*, ed. G. Piatetsky-Shapiro and W.J. Frawley, The MIT Press.

Appendix A

Significant Attributes Used in Current Planning Factors

In general, a small numbers of significant attributes dominantly influence a Functional Unit's duration, are shown in the current planning factors using only a few attributes in determining each factor. As known in Section 2.3, planning factors are expressed in region:equation pairs. Attributes used in region descriptions and region equations are listed respectively in the following table. In Table A.1, "S" indicates that attributes are symbolic and "N" indicates that attributes are numeric. It is observed from Table A.1 that

1) Different planning factors have different significant attributes.
2) In all planning factors, attributes used in region equations are numeric. Except for planning factors P0016, P0018 and P0026, attributes used in region descriptions are symbolic. When numeric attributes are used in region descriptions, they are discretized and treated the same as symbolic attributes. For example, numeric attribute <Bridge_width> is used in region descriptions of P0026. This results in <Bridge_width> = (24'~30') and <Bridge_width> = (31'~56').

3) In all planning factors except planning factors P0016, P0018 and P0020, attributes are used either in region descriptions or region equations but attributes are not used both in region descriptions and region equations. That is to say, the attributes are divided into two groups, one group is used only in region descriptions and the attributes in this group are called region description attributes. The other group of attributes is used only in region equations and the attributes in this group are called equation attributes. In planning factors P0016, P0018 and P0020, the attribute <Tract> is used in region descriptions and equations.

Table A.1 Significant attributes in current planning factors.

Planning	Attributes in region description		Attributes in numerical equations	
factors	Names	Types	Names	Types
P0002	<FHWA_improvement_type>	S		
P0003	<Location_study>	S		
P0004			<Tracts_relocated>	N
P0005			<Tracts_relocated >	N
P0006			<Tracts_condemned>	N
P0007			<Tracts_condemned>	N
P0008			<Length>	N
P0009			<Tracts>	N
P0010	<US81_ind>	S	<Length> <Bridges>	N N
P0011	<Location_study> <US81_ind>	S S	<Length> <Bridges>	N N

P0012	<US81_ind>	S	<Length> <Bridges>	N N
P0013	<US81_ind>	S		
P0014	<US81_ind>	S	<Bridge_replacement> <Length>	N N
P0015			<Bridges>	N
P0016	<Urban_ind> <Tracts>	S N	<Tracts>	N
P0017	<Urban_ind>	S	<Tracts>	N
P0018	<Urban_ind> <Tracts>	S N	<Tracts>	N
P0019	<Urban_ind>	S	<Tracts>	N
P0020	<Urban_ind>	S	<Tracts>	N
P0021	<Urban_ind>	S	<Tracts>	N
P0022	<Relocation>	S	<Tracts>	N
P0023	<US283_ind> <Crossing>	S S	<Length>	N
P0024	<Crossing>	S		
P0025			<Length> <Bridges>	N N
P0026	<Bridge_width> <Metro>	N S	<Bridge_length>	N
P0027	<Borrow> <US81_ind>	S S	<Length>	N
P0028			<Length> <Distance>	N N
P0029			<Sign_footing>	N
P0030			<Light_tower>	N
P0031			<Utilities>	N

P0032			<Length>	N
P0033	<Design>	S		
P0034	<Location_construct> <Design>	S S		
P0035	< Location_construct > <Design>	S S		
P0036	< Location_construct > <Design>	S S		
P0037	<Urban_ind>	S	<Length>	N
P0038	<Urban_ind>	S	<Length>	N
P0039	< Location_construct > <Urban_ind>	S S	<Length>	N
P0040	<Urban_ind> <Location_construct > <Places>	S S S	<Length>	N
P0041	<Urban_ind> <Access_ind>	S S	<Length>	N
P0042	<Design>	S	<Bridges>	N
P0043	<Surface_material>	S	<Length>	N
P0044			<Length> <Bridges>	N N
P0045	<Surface_material>	S	<Length> <Bridges>	N N
P0046	<Design>	S		
P0047	<Design>	S		
P0048			<Bridges>	N
P0049			<Tracts>	N
P0050	<Lane>	S	<Length>	N

P0051	<Construction_under_traffic> <Urban_ind> <Lanes> <Surface_work_type>	S S S S	<Length> <Bridges>	N N
P0052	<Construction_under_traffic> <Urban_ind> <Lanes> <Surface_work_type>	S S S S	<Length> <Bridges>	N N
P0053	<Design> <Urban_ind> <Lanes>	S S S	<Length>	N
P0054	<Design> <Urban_ind> <Lanes>	S S S	<Length>	N
P0055	<Design> <Surface_work_type> <Urban_ind> <Lanes>	S S S S	<Length>	N
P0056	<Design> <Surface_work_type > <Urban_ind> <Lanes>	S S S S	<Length>	N
P0057	<Design>	S	<Bridges>	N
P0058	<Design>	S	<Bridges>	N
P0059	<Sign_project > <Sign_truss>	S S		
P0060	<Sign_project >	S		
P0061	<Sign_truss>	S		
P0062	<Util_required>	S		
P0063	<Urban_ind> <Lanes>	S S	<Length>	N

P0064	<Urban_ind> <Lanes>	S S	<Length>	N
P0065			<Length>	N
P0066			<Length>	N
P0067	<Time>	S		

Appendix B

File Description

This appendix gives file descriptions of source code of the system PFactor. The source code file names (in bold), the number of lines of the files, and the simple descriptions of the files are given, followed by the functions in that file with simple descriptions of the functions.

bforest.c 92 Build up forest consisting of M-model trees for subdata sets

BuildForest() Build up M-model forest

bsfunc.c 120 Basic funcitons

readkey() Get correct reading from data file

getstr(ifp) Get string from input file

FindChar(fc, ifp, n) Find certain character.

IsChar(ifp, c) Confirm a reading from file is a char.

btree.c 283 Building one M-model tree

BuildTree(Tree) Build an M-model tree

NodeAttId(np) Greedy search the attribute used in a node

UsedAtt(i,Node) Check if an attribute is used in ancestor nodes

NodeSprout(Node,sp) Sprout next generation of a node.

AddChild(np,Sibs) Add a child node

errmeasure.c 107 Calculate error measure

CompErr(ofname) Calculation of error by old and new pfactors.

CalcErr(np,Np,base,ofp) Calculation of error in a node

estimate.c 57 Calculate error

Estimate(curTSet, Node, di,simp) Calculate error

exfunc.c 99 Functions of handling example set

GroupEx(ExSet,i,UniqVal,sub) Group tested example sets into subsets based on the discrete values of the attribute

CountSetNo(ExSet) Count the number of example sets

freefunc.c 215 Functions of freeing unnecessory memory

FreeSet (ExSet) Free memory of example set

FreeRoot(Node) Free memory of root node

FreeNode(Node)   Free memory of non-leaf node

FreeSibs(Sibs)   Free memory of sibling nodes

FreeLayer(Layer)   Free memory of a layer

FreeRules(Rules)   Free memory of rules

FreeSubTree(np)   Free memory of subtree

FreeTree(Tree)   Free memory of a tree

FreeGFU(GrpFU)   Free memory of a group of trees

genfactor.c   269   Generating planning factors

PFactors()   Group similar trees and generate pfactors and base quantities

StructCmp(T1,T2)   Compare tree structures

idcmp(np1,np2)   Check the numerical variables in eq

coefcmp(np1,np2)   Compare proportionality of tree eqs

ConstRatio(T1,T2)   Compare Ratio of tree similarities

GetPF(GFuncU)   Find planning factor

groupdata.c   148   Group data set into subdata sets

GroupData()   Group data according to former base quantities

GroupExSet()   Group examples according to base quantities

GetSymAtt()   Finding out the description attributes

GetNumAtt()   Finding out the description attributes

itree.c 67 Initialize a model tree

InitTree(Tree,Exm) Initialize a model tree

listfiles.h 178 List functions

listnum.h 25 List functions

myutil.c 106 Utility functions from "Numerical Recipes"

vector(nl,nh) Allocate a float vector with subscript range v[nl..nh]

ivector(nl,nh) Allocate an int vector with subscript range v[nl..nh

matrix(nrl,nrh,ncl,nch) Allocate a float matrix with subscript range

free_vector(v,nl,nh) Allocate a float vector with subscript range v[nl..nh]

free_ivector(v,nl,nh) Free an int vector allocated with ivector()

free_matrix(m,nrl,nrh,ncl,nch) Free a float matrix allocated by matrix()

myutil.h 52 Type definition

numfunc.c 415 Functions for regression analysis from "Numerical Recipes"

sort(n,ra) Sort data in ascending order

Regress(ExSet,NPT,aid,NPOL,a,chisq) Regression Analysis

svdfit(ExSet,aid,y,sig,ndata,a,ma,u,v,w,chisq,lyy,funcs) Singular value decomposition fitting svdvar(v,ma,w,cvm) Singular value decomposition variance

fdata(ep,aid,p,ano) Get data for calculating standard deviation

svbksb(u,w,v,m,n,b,x) <Numerical recipe in c> p64

svdcmp(a,m,n,w,v) <Numerical recipe in c> p67

pythag(a,b) Compute (sq(a)+sq(b))^(1/2) without destructive underflow or overflow

otfunc.c 53 Functions to do basic checking

CoeSign(Rules) Check the signs of coefficients

FindMaxPar(np) Find Max parameter

pfactor.c 98 Main program

main(Argc, Argv) Main program

getopt(Argc, Argv, Str) Get options

ptree.c 96 Prune a model tree

runeTree(Tree) Prune a model tree

PruneNode(Node) Prune a node

NodeORsub(np,subErr) Compare node with its sub tree

AllLeaf(Node) Check whether a node is of all leaf children

readfile.c 258 Read data from input data file

GetData(ifname) Get data

GetAttNo(ifp) Get the number of columns in data file

GetAttInfo(ifp) Get the information of all attributes, mainly including the first three lines of data file.

GetExample(ifp) Obtain all examples in data file.

PreCheck() Check the format of data file

res_lst.c 251 List intermediate results

lst_units() List examples

lst_A(ap) List attribute information

lst_E(ExSet) List examples in a node

lst_row(ep) List one example

ShowUnique() List discrete values of symbolic attributes

lst_rules_bylayer(tp) List rules by layer

PrintConds(np) List region description

PrintEq(np) List region equation

lst_rules(Rules) List rules by rule chain

lst_ivector(n,ParId) List numeric attributes in region description

lst_node(tp,Node) List information on a node

lst_layer(tp,lp) List information on a layer

lst_pf() List planning factors

lst_bq() List base quantities

lst_sib(sib) List sibling nodes

rules.c 99 Find region:equation pairs

FindRuleChain(tp) Find region:equation pairs

Chain(rp,np) Find a chain region:equation pairs

Recover(Node) Prepare data set for next partitioning

simplify.c 236 Simplify linear model

WtSimplify(Node) Eliminate numeric attribute by weighted standard deviation

SdSimplify(Node) Eliminate numeric attribute by standard deviation

NewParId(m,np) Numeric attribute inherited by child nodes

types.h 167 Type definitions