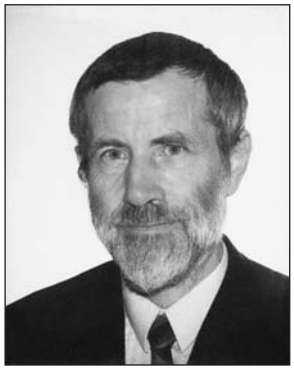
COMMUNICATIONS



Dear reader,

You have got in your hands this volume of the university scientific letters, which is a third time devoted mostly to informatics and its applications to a broad spectrum of scientific and professional branches.

Nowadays, various forms of informatics penetrate almost into every human activity and the current issue tries to reflect this phenomenon. Inside this volume we attempt to submit papers written by authors not only from the Faculty of Management Science and Informatics and other faculties of the University of Zilina, but we appealed to professionals from other cooperating universities to contribute to the topic mentioned above.

In the frame of this issue you can find works dealing with optimisation techniques supported by means of informatics rather than pure problems of informatics. It is no surprise that most of the works are devoted to applications of informatics to transport. This issue continues with topics concerning intelligent transportation systems and various sorts of routing and scheduling problems. But, the attention is also paid to security and speech to text system and other non-transport problems of informatics.

I would like to express my opinion that this volume would attract attention of professionals from relevant scientific branches and ignite their interest in some future cooperation in the area of sophisticated decision support tools as an enhancement of information systems.

Jaroslav Janacek

Jaroslav Janacek *

A SECURITY MODEL FOR AN OPERATING SYSTEM FOR SECURITY-CRITICAL APPLICATIONS IN SMALL OFFICE AND HOME ENVIRONMENT

Personal computers are often used in small office and home environment for different purposes ranging from general web browsing and e mail processing to processing data that are sensitive regarding their confidentiality and/or integrity. Common operating systems do not provide sufficient protection. We present a security model combining the well known benefits of mandatory access control in classified information processing systems with the typical home and small office computer use. We use a simple two-dimensional data classification scheme and present a security model with provable properties that significantly reduces the risks of confidentiality and/or integrity protection violation.

1. Introduction

During the past few years we have been observing a significant increase in the number of people who use computers to perform tasks where security is important. Typical such applications (denoted *security-critical* applications in this paper) that are of interest to general public and can be expected to be used on home and office computers, include Internet banking, e-government applications, electronic signature creation and verification applications. Organizations use information systems to store and process confidential business data and personal data. Unauthorized access to information stored or processed by all of the mentioned applications (and many others) can often cause a substantial loss to the affected person or organization. It is usually the user's responsibility to protect the sensitive data. However, common users are not information security experts and can only follow some guidelines given to them. Even that is usually possible only if the guidelines are simple enough.

While a larger organization can dedicate some computers to security-critical applications and protect them against unauthorized access, modification or software installation, it can hardly be expected in home or small office environments. In such environments, the same computer is usually used for many different purposes – such as web browsing, e mail processing, running programs from untrustworthy sources, etc. – alongside running security-critical applications. We will discuss the typical examples of the applications used in these environments and consider the data they use in terms of security requirements in the next section.

We will concentrate on two security aspects – *confidentiality* and *integrity*. The goal of confidentiality protection is to prevent unauthorized subjects from obtaining the protected information. It includes the protection of stored information as well as the protection of information being transferred (e.g. by means of a computer network). The goal of integrity protection is to prevent unauthorized modifications of the protected information from taking place, or, if they cannot be prevented, to provide means of detection of such modifications.

Both confidentiality and integrity can be protected using different security mechanisms. We will concentrate on *access control* in this paper. Other popular security mechanisms used to protect confidentiality and integrity include cryptography – encryption, message authentication codes and digital signatures.

Applications processes can directly manipulate data in their memory. In order to access other data, processes have to use the services of an operating system. This enables the operating system to control access to the data and to enforce a security policy based on various attributes associated with the application process and with the data object, and thus makes the operating system a good place where to deal with security.

Currently common operating systems used in the target environment, such as the Professional edition of Microsoft Windows XP, Vista or various distributions of Linux, provide access control features to protect the data stored in files within the most often used native filesystems. The access control provided by the mentioned operating systems is based on the following principles:

- Every process – *subject* performs its actions on behalf of a user.
- Every filesystem object (i.e. a file, a directory, ...) is *owned* by a user (or a group of users) – its *owner*.
- The owner of an object can specify a discretionary access control list for the object. Each entry in the access control list specifies the permitted *operations* for processes running on behalf of an identified user (or a group of users).

* **Jaroslav Janacek**
Department of Computer Science, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Slovakia,
E-mail: janacek@dcs.fmph.uniba.sk

The set of operations depends on the operating system – Linux uses three basic operations (read, write, execute/use directory) while Windows uses more finer-grained operations. The common properties of this type of access control are:

- it can be used to protect data against access and/or modification by processes of a different user,
- but it cannot prevent a process from accessing/modifying data owned by the user that the process runs on behalf of.

If a process can communicate with an external system (e.g. the Internet), it can transmit the data from any file readable by the user the process runs on behalf of. It can also modify any file writeable by the user. The behaviour of certain types of applications can be significantly influenced, as will be discussed later, by external entities. This opens a way for external attackers to gain unauthorized access to the user's data.

We present a security model designed to protect confidentiality and integrity of data classified in terms of the confidentiality and integrity protection requirements in the home and small office environments.

2. Security needs of the typical data in the small office and home environment

We will consider the typical classes of applications used in the target environment in this section, and we will specify the security needs, in terms of confidentiality and integrity, of the data they process.

2.1. Examples of applications

General Internet access

One of the typical classes of applications is the class of the applications used to access Internet resources. It includes web-browsers, e-mail clients, and various other communication systems (e.g. ICQ, IRC, Voice over IP, video-conferencing systems, ...). The common feature of these applications is that they communicate with external systems that cannot be trusted to protect the confidentiality of the data transferred to them, nor can they be trusted not to send *malicious data* back to our system. By the term *malicious data* we mean data that have been prepared in a way to abuse a weakness of the application receiving them. Applications often contain programming errors (bugs) that can be abused by providing specially prepared data. These errors can often be abused to execute arbitrary code as if it were a part of the application, i.e. with the same privileges as the application itself. Therefore, even if the application itself is believed not do anything unwanted, its behaviour may be changed, by processing malicious data, in an unpredictable way.

It has to be assumed that the applications of this class:

- export any data available to them to the Internet,
- import (potentially) malicious data from the Internet, and therefore, their output has to be considered (potentially) malicious too,
- may become malicious by processing the malicious input.

Malicious applications

A special class of applications is the class of malicious applications. These are applications that have been intentionally programmed to perform malicious activities. The typical examples are computer viruses, worms, Trojan horses and other kinds of so called malware. They can be downloaded from the Internet by the user, received as an attachment of an e-mail message, or a vulnerable application may be turned into a malicious one by processing malicious data. The user is usually unaware of the fact that a particular application is malicious.

It has to be assumed that the malicious applications do anything not prevented by the operating system or the environment of the computer (e.g. a network firewall).

Local applications

The class of local applications contains the applications that are used to process data stored in a local filesystem. These applications generally do not need network access to perform their tasks. The typical examples are text processors, spreadsheets, presentation software, graphic editors, ...

Local applications are used to process data with varying requirements regarding the confidentiality and integrity protection. If they process malicious data, they may become malicious due to programming errors.

Sensitive web access

Web browsers are often used to access remote services that process data requiring confidentiality and/or integrity protection. A typical example is an Internet-banking system. It provides access to financial information; it allows the user to submit transaction orders to the bank, etc. It also processes authentication data (e.g. passwords). All such data may be considered confidential by the user, and therefore, are to be adequately protected. The confidentiality and the integrity of the data during their transmission is usually protected by means of cryptography. Cryptography is usually also used to provide authentication of the remote system. But the data is also to be protected while stored in the memory or in a file on the local computer. Consider an instance of a web browser used for general Internet access. It may have processed some malicious data, and therefore, it may have become a malicious application exporting everything to an attacker. If the instance of the web browser is later used to access an Internet banking system, all the confidential information may leak.

Digital signature and data encryption/decryption

Digital signature creation applications, as well as data decryption applications need access to a private key. Digital signature verification applications, as well as data encryption applications need access to a public key.

The private key is a very sensitive piece of information the confidentiality of which has to be protected. The integrity of the private key has to be protected as well because its modification can lead not only to the loss of ability to create correct digital signatures or to decrypt data, but also to the leak of information that is sufficient to compute the corresponding private key in certain cases.

The public key requires no confidentiality protection, but it does require integrity protection. If attackers were able to modify the public key used to verify a digital signature, they would be able to create a digitally signed document that would pass the signature verification process. In the case of encryption, if the public key were modified by an attacker, the attacker would be able to decrypt the encrypted data instead of the intended receiver.

The encrypted output of a data encryption application may be transmitted via communication channels that do not provide confidentiality protection even if the confidentiality of the original data is to be protected. The output of a data decryption application may also require confidentiality protection.

2.2. Data classification scheme

We can conclude, from the previous subsection, that the need of confidentiality protection and the need of integrity protection are independent on each other. Some data need integrity protection while they can be disclosed to the public, some data need both, some need none. We will, therefore, use a two-dimensional classification scheme for the data consisting of the *confidentiality* level and the *integrity* level.

We will use three confidentiality levels:
- 0 – public data,
- 1 – normal data (C-normal),
- 2 – sensitive data (C-sensitive).

The public data require no confidentiality protection. They may be freely transmitted via communication channels and/or to remote systems that provide no confidentiality protection. An example of public data is the data downloaded from public Internet.

The normal data are to be protected by means of discretionary access control against unauthorized reading by other users than the owner of the data.

The C sensitive data are the data that their owner (a user) wishes to remain unreadable to the others regardless of the software the user uses, and even if the users makes some mistakes (such as setting wrong access rights for discretionary access control). Examples of C sensitive data are private and secret keys, passwords for Internet banking, etc.

We will also use three integrity levels:
- 0 – potentially malicious data,
- 1 – normal data (I-normal),
- 2 – sensitive data (I-sensitive).

The requirement of the integrity protection of data is tightly coupled to the trustworthiness of the data. The trustworthiness of data can be thought of as a metric of how reliable the data are. If some data can be modified by anyone, they cannot be trusted not to contain wrong or malicious information. If some data are to be relied on, their integrity has to be protected.

The potentially malicious data require no integrity protection, and can neither be trusted to contain valid information, nor can be trusted not to contain malicious content.

The normal data are to be protected by means of discretionary access control against unauthorized modification by other users than the owner of the data.

The I sensitive data are the data that their owner wishes to remain unmodified by the others regardless of the software the user uses, and even if the user makes some mistakes. The I sensitive data are to be modifiable only under special conditions upon their owner's request. A special category of I sensitive data is the category of the shared system files such as the programs, the libraries, various system-wide configuration files, the user database, ... Some of these files may be modifiable by the designated system administrator, some of them should be even more restricted.

3. Security model

A common approach to ensuring the confidentiality and/or the integrity of information in systems that deal with data classified into several confidentiality/integrity levels, is to define an information flow policy, and then to enforce the policy. In order to enforce an information flow policy, subjects are divided into two categories – trusted and untrusted. A trusted subject is a subject that is trusted to enforce the information flow policy (with exceptions) by itself; an untrusted subject is a subject that is not trusted to enforce the policy by itself, and therefore the policy has to be enforced on the subject's operations by the system.

A typical information flow policy protecting confidentiality (e.g. one based on Bell-LaPadula model [1]) states that a subject operating at a confidentiality level C_S may only read from an object with a confidentiality level C_O if $C_S \geq C_O$, and may only write to an object with a confidentiality level C_O if $C_S \leq C_O$. If a subject is to be able to read from a more confidential object, and to write to a less confidential object, it has to be a trusted subject.

A typical information flow policy protecting integrity (e.g. one based on Biba model [2]) states that a subject operating at an integrity level I_S may only read from an object with an integrity level I_O if $I_S \leq I_O$, and may only write to an object with an integrity level I_O if $I_S \geq I_O$. Only a trusted subject can read from an object with a lower integrity level, and write to an object with a higher integrity level.

The problem with the division of subjects into the two categories is that it would lead to the need of too many trusted subjects in the home and office environment. Considering the examples given in the previous section, many of the identified applications would have to be trusted.
We will divide the subjects into three categories:
- untrusted subjects,
- partially trusted subjects, and
- trusted subjects.

An *untrusted* subject is a subject that is not trusted to enforce the information flow policy. It is assumed to perform any operations on any objects unless it is prevented from doing so by the operating system.

A *trusted* subject is a subject that is trusted to enforce the information flow policy by itself. A trusted subject may be used to perform tasks that require violation of the policy under conditions that are verified by the trusted subject. A trusted subject can, therefore, be used to implement an exception to the policy.

A *partially trusted* subject is a subject that is trusted to enforce the information flow policy regarding a specific set of objects, but not trusted to enforce the information flow policy regarding any other objects. In other words, a trusted subject is

- trusted not to transfer information from a defined set of objects (designated inputs) at a higher confidentiality level to a defined set of objects (designated outputs) at a lower confidentiality level in a way other than the intended one, and
- trusted not to transfer information from a defined set of objects (designated inputs) at a lower integrity level to a defined set of objects (designated outputs) at a higher integrity level in a way other than the intended one, but
- not trusted not to transfer information between any other objects.

The sets of designated inputs and outputs regarding confidentiality are distinct from the sets regarding integrity. Any of the sets may be empty. A partially trusted subject, like a trusted one, can be used to implement an exception to the policy, because it can violate the policy (and it is trusted to do it only in an intended way).

The most important difference between trusted and partially trusted subjects is in the level of trust. While trusted subjects are completely trusted to behave correctly, partially trusted subjects are only trusted not to abuse the possibility of the information flow violating the policy between a defined set of input objects and a defined set of output objects.

The presented version of the model is limited to two basic operations: read and write. It can be conservatively extended to support other operations on objects, as well as operations on subjects, however.

3.1. Information flow policy objectives

We will first specify the policy objectives in an informal way, and then we will define the policy formally.

In accordance with the classification of objects, the information flow policy has the following objectives:

1. Prevent reading of C-sensitive objects by subjects of other users than the owner of the object.
2. Prevent modification of I-sensitive objects by subjects of other users than the owner of the object.
3. Prevent information passing from objects with a higher confidentiality level to objects with a lower confidentiality level by untrusted subjects with the exception stated below.
4. Allow the user to explicitly allow a subject to read a C-normal object on per request basis. The user's approval in such case must be obtained via a mechanism independent on the subject. The idea of this objective is to allow the user to perform operations such as submitting a C-normal document to a remote system, that is not trusted to process C-normal data in general and is considered a public object with respect to our classification scheme, without the need to reclassify the document first (and, therefore, to expose its content to any subject). Because this approach is very prone to the user's mistakes, it should be limited to C-normal objects and not applicable to C-sensitive objects.
5. Prevent information passing from objects with a lower integrity level to objects with a higher integrity level by untrusted subjects.
6. Allow the user to specify the maximal integrity level for each subject and prevent the subject from writing to objects with a higher integrity level.
   The idea of this objective is to prevent modification of objects with a high integrity level unless required by the user.
7. Allow the user to define four sets of special input and output objects (two sets for confidentiality protection and two sets for integrity protection) and two special confidentiality levels (for reading and writing respectively) and two special integrity levels associated with the sets for each partially trusted subject, and apply the same rules to partially trusted subjects with the following exceptions:
   a) Allow a partially trusted subject to transfer information from an object O_{in} with a confidentiality level c_{in} to an object O_{out} with a confidentiality level $c_{out} < c_{in}$ if the object O_{in} is in the input set for confidentiality protection, the object O_{out} is in the output set for confidentiality protection, cin is at most the special confidentiality level for reading, and c_{out} is at least the special confidentiality level for writing.
   b) Allow a partially trusted subject to transfer information from an object O_{in} with an integrity level i_{in} to an object O_{out} with an integrity level $i_{out} > i_{in}$ if the object O_{in} is in the input set for integrity protection, the object O_{out} is in the output set for integrity protection, i_{in} is at least the special integrity level for reading, and iout is at most the special integrity level for writing.
8. Allow the user to define the maximal confidentiality level for reading ${C^{max}}_S$, the minimal confidentiality level for writing ${C^{min}}_S$, the minimal integrity level for reading ${I^{min}}_S$ and the maximal integrity level for writing ${I^{max}}_S$ for each trusted subject S, and allow the trusted subject S to read from an object O with a confidentiality level C_O and an integrity level I_O only if $C_O \leq {C^{max}}_S$ and $I_O \geq {I^{min}}_S$, and allow the trusted subject S to write to the object O only if $C_O \geq {C^{min}}_S$ and $I_O \leq {I^{max}}_S$.

The trusted subjects are, therefore, able to transfer information between any objects within some limits.

3.2. Information flow policy formal definition

Let C_O, I_O denote the confidentiality and integrity levels associated with an object O, U_O denote the owner of O, and L_O denote a label assigned to O. The labels will be used to specify the sets of special input and output objects mentioned in the objective 7 above. Let CR_S, CW_S denote the highest confidentiality level the subject S can normally read from and the lowest confidentiality level it can normally write to; let CRL_S denote the highest confidentiality level the subject S can read from if the respective object is a member of the special input set for confidentiality protection; let CWL_S denote the lowest confidentiality level the subject S can write to if the respective object is a member of the special output set for confidentiality protection; let $CRLS_S$ and $CWLS_S$ denote the sets of labels defining the special input and output sets of the subject S for confidentiality protection (an object O is a member of the special input set if the label L_O is in $CRLS_S$). Let IR_S, IW_S, IRL_S, IWL_S denote the lowest integrity level for reading, the highest integrity level for writing, the lowest integrity level l for reading from the special input objects, and the highest integrity level for writing to the special objects of the subject S, and let $IRLS_S$ and $IWLS_S$ denote the sets of labels defining the special input and output sets of the subject S for integrity protection. Let U_S denote the user the subject S is running on behalf of. Let $IRUS_S$ and $CWUS_S$ denote the sets of users that are trusted by S to maintain trustworthy integrity and confidentiality levels respectively on objects they own. $IRUS_S$ would typically include the special system user identifiers used as owners of system files that need to be relied on by applications (e.g. system shared libraries, system programs, ...). $CWUS_S$ would typically include the special system user identifiers used as owners of confidential files that have to be writeable by processes accepting input from users (e.g. the owner of the password database).

Using the notations above a subject S can read from an object O only if **read**(S, O) is true, and S can write to O only if **write**(S, O) is true:

$$\begin{aligned}\textbf{read}(S, O) = {} & [CR_S \geq C_O \vee (CRL_S \geq C_O \wedge L_O \in CRLS_S) \vee \\ & (C_O \leq 1 \wedge \textbf{UserApprovedRead}(S,O))] \wedge \\ & [IR_S \leq I_O \vee (IRL_S \leq I_O \wedge L_O \in IRLS_S)] \wedge \\ & [U_S = U_O \vee C_O \leq 1] \wedge \\ & [U_S = U_O \vee U_O \in IRUS_S \vee IR_S \leq 1]\end{aligned}$$

$$\begin{aligned}\textbf{write}(S, O) = {} & [CW_S \leq C_O \vee (CWL_S \leq C_O \wedge L_O \in CWLS_S)] \wedge \\ & [IW_S \geq I_O \vee (IWL_S \geq I_O \wedge L_O \in IWLS_S)] \wedge \\ & [U_S = U_O \vee I_O \leq 1] \wedge \\ & [U_S = U_S \vee U_O \ \ CWUS_S \vee CW_S \leq 1]\end{aligned}$$

The **UserApprovedRead**(S, O) is the function returning true if the user has approved the exception specified in the objective 4 in the previous subsection.

Each untrusted subject S must obey the following conditions:

- $CW_S = CWL_S \geq CR_S = CRL_S$
- $IW_S = IWL_S \leq IR_S = IRL_S$
- $CWLS_S = CRLS_S = IWLS_S = IRLS_S = \varnothing$

Each partially trusted subject S must obey the following conditions:

- $CW_S \geq CR_S$
- $CW_S \geq CRL_S$
- $CWL_S \geq CR_S$
- $IW_S \leq IR_S$
- $IW_S \leq IRL_S$
- $IWL_S \leq IR_S$

The structure of the **read** and **write** predicates is as follows. The subexpressions in the first pair of square brackets deal with the objectives 3, 4, 7a, and 8, i.e. with preventing the unintended flow of information from a more confidential object to a less confidential one. The subexpressions in the second pair of square brackets deal with the objectives 5, 6, 7b, and 8, i.e. with preventing the unintended flow of information from an object with a lower integrity level to an object with a higher integrity level. The subexpressions in the third pair of square brackets deal with the objectives 1 and 2. The subexpression in the fourth pair of square brackets of **write** prevents a subject running on behalf of a user U_1 from passing information from a C-sensitive object owned by U_1 to a C-sensitive object owned by another user U_2, where it could be read from by subjects of U_2, thus violating the objective 1. The subexpression in the fourth pair of square brackets of **read** has a similar role for integrity protection – it prevents reading of I-sensitive data prepared by another user. The exceptions using the sets of trusted users are to support reading of I-sensitive system files and writing to C-sensitive system objects.

3.3. Security properties of the information flow policy

We have defined the information flow policy with its objectives in mind. It does not, however, provide a guarantee that it really meets the objectives. We will now state some basic security properties in the formal way. The theorems can be proven although the proofs are out of the scope of this paper.

First, we will formally define the flow of information within the system.

Definition 1: Let $\boldsymbol{S}$ be a set of subjects and $\boldsymbol{O}$ be a set of objects. Let O_0, $O_{out} \in O$ be any two objects. We say that a policy allows an information flow from O_0 to Oout within the system ($\boldsymbol{S}$, $\boldsymbol{O}$) and denote it as **flow**($\boldsymbol{S}$, $\boldsymbol{O}$, O_0, O_{out}) if there exists a finite sequence of pairs (S_1, O_1), (S_2, O_2), ..., (S_n, O_n) where $\forall\ i \in [1, ..., n]$: $O_i \in \boldsymbol{O} \wedge S_i \in \boldsymbol{S}$ such that

$$\forall\ i \in [1, ..., n]: \textbf{read}(S_i, O_{i-1}) \wedge \textbf{write}(S_i, O_i) \text{ and } O_n = O_{out}$$

where **read**(S, O) and **write**(S, O) are the functions of the policy determining whether the subject S can read from, or write to the object O.

Using the formal definition of flow, we can precisely define what we mean by an information leak – a violation of the confidentiality protection requirements.

Definition 2: Let $\boldsymbol{S}$ be a set of subjects and $\boldsymbol{O}$ be a set of objects. We say that our policy allows an information leak within the system ($\boldsymbol{S}$, $\boldsymbol{O}$) if

$$\exists\, O_a, O_b \in \boldsymbol{O}\colon C_{Oa} > C_{Ob} \wedge \mathbf{flow}(\boldsymbol{S}, \boldsymbol{O}, O_a, O_b)$$

We can also define the precise meaning of a violation of the integrity protection requirements – information spoiling.

Definition 3: Let $\boldsymbol{S}$ be a set of subjects and $\boldsymbol{O}$ be a set of objects. We say that our policy allows information spoiling within the system ($\boldsymbol{S}$, $\boldsymbol{O}$) if

$$\exists\, O_a, O_b \in \boldsymbol{O}\colon I_{Oa} < I_{Ob} \wedge \mathbf{flow}(\boldsymbol{S}, \boldsymbol{O}, O_a, O_b)$$

Having the definitions, we can state the basic security properties using the following theorems. The first two theorems deal with the case when there are only *untrusted* subjects. In such case, the information flow policy guarantees that no information from a more confidential object can end up in a less confidential object, and that no information from an object with a lower integrity level can influence any object with a higher integrity level.

Theorem 1: If $\boldsymbol{S}$ is a set of *untrusted* subjects and $\boldsymbol{O}$ is a set of objects, our policy does not allow any information leak within the system ($\boldsymbol{S}$, $\boldsymbol{O}$) unless approved by the user.

Theorem 2: If $\boldsymbol{S}$ is a set of *untrusted* subjects and $\boldsymbol{O}$ is a set of objects, our policy does not allow any information spoiling within the system ($\boldsymbol{S}$, $\boldsymbol{O}$).

Another two theorems deal with the case when there may be some *partially trusted* subjects as well. The first of these theorems says that in order to pass information from an object with a higher confidentiality level to an object with a lower confidentiality level using only untrusted and partially trusted subjects, each subject which passes information from an object with a higher confidentiality level to an object with a lower confidentiality level, must be a partially trusted subject and it must be passing the information from its specially labelled input to its specially labelled output. Assuming that no partially trusted subject passes information from its special inputs to its special outputs in an unintended way, any information leak allowed by the policy within a system without trusted subjects is intended.

Theorem 3: Let $\boldsymbol{S}$ be a set of *untrusted* and/or *partially trusted* subjects and let $\boldsymbol{O}$ be a set of objects. Let O_0, $O_{out} \in \boldsymbol{O}$ be two objects such that $C_{O0} > CO_{out}$ and $\mathbf{flow}(\boldsymbol{S}, \boldsymbol{O}, O_0, O_{out})$. For every finite sequence of pairs (S_1, O_1), ..., (S_n, O_n) such that $\forall\, i \in [1, ..., n]$: $S_i \in \boldsymbol{S} \wedge O_i \in \boldsymbol{O} \wedge \mathbf{read}(S_i, O_{i\text{-}1}) \wedge \neg\, \mathbf{User\ ApprovedRead}(S_i, O_{i\text{-}1}) \wedge \mathbf{write}(S_i, O_i) \wedge O_n = O_{out}$, for each pair (S_j, O_j) such that $C_{Oj\text{-}1} > C_{Oj}$:

S_j is a partially trusted subject, and
$L_{Oj\text{-}1} \in CRLS_{Sj}$, and
$C_{Oj\text{-}1} \leq CRL_{Sj}$, and
$L_{Oj} \in CWLS_{Sj}$, and
$C_{Oj} \geq CWL_{Sj}$

The last theorem says that in order to pass information from an object with a lower integrity level to an object with a higher integrity level using only untrusted and partially trusted subjects, each subject which passes information from an object with a lower integrity level to an object with a higher integrity level, must be a partially trusted subject and it must be passing the information from its specially labelled input to its specially labelled output. Assuming that no partially trusted subject passes information from its special inputs to its special outputs in an unintended way, any information spoiling allowed by the policy within a system without trusted subjects is intended.

Theorem 4: Let $\boldsymbol{S}$ be a set of *untrusted* and/or *partially trusted* subjects and let $\boldsymbol{O}$ be a set of objects. Let O_0, $O_{out} \in \boldsymbol{O}$ be two objects such that $IO_0 < IO_{out}$ and $\mathbf{flow}(\boldsymbol{S}, \boldsymbol{O}, O_0, O_{out})$. For every finite sequence of pairs (S_1, O_1), ..., (S_n, O_n) such that $\forall\, i \in [1, ..., n]$: $S_i \in \boldsymbol{S} \wedge O_i \in \boldsymbol{O} \wedge \mathbf{read}(S_i, O_{i\text{-}1}) \wedge \mathbf{write}(S_i, O_i) \wedge O_n = O_{out}$, for each pair (S_j, O_j) such that $I_{Oj\text{-}1} < I_{Oj}$:

S_j is a partially trusted subject, and
$L_{Oj\text{-}1} \in IRLS_{Sj}$, and
$I_{Oj\text{-}1} \geq IRL_{Sj}$, and
$L_{Oj} \in IWLS_{Sj}$, and
$I_{Oj} \leq CWL_{Sj}$

4. Conclusions

The presented security model supplements the traditional discretionary access control by providing protection of confidentiality and integrity of sensitive data against malicious applications running on behalf of the owner of the data. When extended to cover other operations (e.g. creation and deletion of objects, object attributes' manipulation, interaction between subject, etc.) and implemented, it will allow a user to run security critical applications alongside potentially malicious applications while assuring the user that the malicious applications cannot interfere with the sensitive data, whether regarding the confidentiality or the integrity aspect. This would be a significant improvement of the current state in the security of common operating systems in home and small office environment.

References

[1] BELL D. E., LA PADULA L. J.: *Secure Computer System: Unified Exposition and Multics Interpretation,* Technical report, 1976.
[2] TIPTON H. F., KRAUSE M. (editors): *Information Security Management Handbook*, 5th edition, CRC Press LLC, 2004, ISBN 0-8493-1997-8.

Stanislav Paluch *

A MULTI LABEL ALGORITHM FOR *k* SHORTEST PATHS PROBLEM

The paper presents an algorithm for computing k shortest walks or k shortest paths in a directed graph G = (V, A). The proposed algorithm can be applied for solving the k shortest paths problem in an undirected graph G = (V, E), too, by transforming the graph G = (V,E) to the digraph G' = (V, A) where the arc set A contains a couple of arcs (u,v), (v,u) for every edge [u,v] E.

1. Introduction

Many problems related to transportation or communication lead to the problem to find in a network relatively short point to point path which has to fulfill certain special constraints that cannot be simply formulated by means of graph theory. Such problems can be solved by successive enumeration of k shortest paths and consequently by choosing that one from them which complies with given constraints.

This work was motivated by the problem of finding a bus and train connection using digitalized bus time table and train time table. The user requires not only one but several proposals of such connections and he chooses among them that one which fulfills his special personal requirements. The connection searching problem can be formulated as a shortest path problem in a huge acyclic digraph G having a vertex for every pair (*stop*, *minute*) where *stop* is a bus stop or railway station and *minute* $\in \{0, 1, ..., 1439\}$ is a minute of the day. The arc set of digraph G is the set of all ordered pairs

((*deperature_stop*, *deperature_time*), (*arrival_stop*, *arrival_time*)).

Such a digraph G for the Slovak Republic contains several millions vertices and enormous number of arcs.

The k shortest path problem is frequently studied in literature. Plesnik presents in [3] a procedure based on prohibiting edges of till now best paths, Yen gives in [4], [5] a deviation algorithm with running time $O(kn(m+n\log n))$ – this worst case assessment has been the best known for long time. Algorithm by Gotthilfa, and Lewenstein [1] (issued in March, 2009) runs in time $O(kn(m+ +n\log\log n))$.

Algorithms treated in literature have very good theoretical and practical complexity, but none of them seemed to us to be suitable for our problem – some of them seemed to be too tricky or seemed to be difficult for implementation namely because of time shortage. The requirements for our algorithm were

1. Theoretically clear and simple
2. Easy and fast implementable
3. Relatively fast even in huge graphs

The result is algorithm with complexity $O(Km(\log(Km)+n))$ for general directed graphs and $O(Km \cdot \log(Km))$ for directed acyclic graphs, where K is the number of shortest paths, n is the number of vertices and m is the number of arcs. The only nontrivial structure used in proposed algorithm is the Fibonacci heap with operations `insert` and `extract_min`,

2. Terminology

A **digraph** (a directed graph) is an ordered pair $G = (V, A)$, where V is a nonempty finite set and A is a set of ordered pairs of the type (u, v) such that $u \in V$, $v \in V$ and $u \neq v$.

The elements of V are called **vertices** and the elements of A are called **arcs** of the digraph G.

A **graph** (an undirected graph) is an ordered pair $G = (V, E)$, where V is a nonempty finite set and E is a set of unordered pairs of the type $\{u, v\}$ such that $u \in V$, $v \in V$ and $u \neq v$.

The elements of V are called **vertices** and the elements of E are called **edges** of the graph G.

A (v_1, v_2) – **walk** in digraph $G = (V, A)$ is an alternating sequence of vertices and arcs of the form

$$\mu(v_1, v_k) = (v_1, (v_1, v_2), v_2, (v_2, v_3)\, v_3, ..., v_{k-1}, (v_{k-1}, v_k), v_k)$$

A **trivial walk** is a walk $\mu(v, v) = v$ containing only one vertex.

* **Stanislav Paluch**
Faculty of Management Science and Informatics, University of Zilina, Slovakia, E-mail: Stanislav.Paluch@uniza.sk

A (v_1, v_k) - **trail** in G is a (v_1, v_k) - walk in G with no repeated arcs. A (v_1, v_k) - **path** in G is a (v_1, v_k) - walk in G with no repeated vertices.

A walk, trail and path in undirected graph can be defined in an analogical way.

An **arc weighted digraph** $G = (V, A, c)$ is an ordered triple where $G = (V, A)$ is a digraph and $c : A \to R$ is a real function defined on the arc set A, the value $c(a)$ for $a \in A$ is called the **weight** of the arc a (or sometimes the arc-weight, the length or the cost of the arc a). An **edge weighted graph** can be defined similarly. In this paper we will assume that $c(a) \geq 0$. This condition is fulfilled in many practical applications.

The **length of the walk** $\mu(u, v)$ in a digraph $G = (V, A, c)$ is the total sum of arc-weights of its arcs, whereas the arc weight is added to the total sum so many times how many times it appears in the walk $\mu(u, v)$. The length of the walk in undirected graph is defined similarly.

Suppose that $\mu(v_1, v_k) = (v_1, (v_1, v_2), ..., (v_{k-1}, v_k), v_k)$ and $\mu(u_1, u_l) = (u_1, (u_1, u_2), ..., (u_{l-1}, u_l), u_l)$ are two walks and let $v_k = u_1$.

The **concatenation** $\mu(v_1, v_k) \oplus \mu(u_1, u_l)$ of walks $\mu(v_1, v_k)$, $\mu(u_1, u_l)$ is the walk

$$\mu(v_1, v_k) \oplus \mu(u_1, u_l) = (v_1, (v_1, v_2), v_2, ..., (v_{k-1}, v_k), v_k) \equiv$$
$$\equiv (u_1, (u_1, u_2), ..., v_{l-1}, (v_{l-1}, v_l), v_l) .$$

Denote $V^+(v) = \{w | (v, w) \in A\}$, $A^+(v) = \{(v, w) | (v, w) \in A\}$. The **forward star** FSTAR(v) of the vertex $v \in V$ is the subgraph of $G = (V, A)$ with vertex set $V^+(v) \cup \{v\}$ with arc set $A^+(v)$, i. e. FSTAR(v) = $(V^+(v) \cup \{v\}, A^+(v))$.

3. Algorithm

Assume that $V = \{1, 2, \ldots, n\}$.

Denote

`n`	- the number of nodes of the digraph G = (V, A, c),	(1)
`m`	- the number of arcs of the digraph G = (V, A, c),	(2)
`s, f`	- starting and finishing vertex	(3)
`K`	- the number of searched shortest paths,	(4)
`c(u,v)`	- the length of the arc $(u, v) \in A$	(5)

The `k`-th shortest `(s,w)`-path will be denoted by μ_k `(s,w)` for arbitrary `w` $\in V$.

For every vertex `w` $\in V$ and for `k` = 1, 2, ..., `K` we will define step by step at most `K` definitive three-dimensional labels

```
deflab[w][1][1], deflab[w][1][2], deflab[w][1][3]
deflab[w][1][2], deflab[w][2][2], deflab[w][2][3]
. . . . . .
deflab[w][k][1], deflab[w][k][2], deflab[w][k][3]
. . . . . .
deflab[w][K][1], deflab[w][K][2], deflab[w][K][3].
```

Let `deflab[w][k][1-3]` be the `k`-th three-dimensional label assigned to a vertex `w`. The number `k` is called the rank of the label `deflab[w][k][1-3]`.

The above mentioned labels have the following meaning:

`deflab[w][k][1]` - the length of the `k`-th shortest `(s,w)`- path μ_k`(s,w)`	(6)
`deflab[w][k][2]` - the last but one vertex of the `k`-th shortest `(s,w)`- path μ_k`(s,w)`	(7)
`deflab[w][k][3]` - the rank of the label of the last but one vertex of the `k`-th shortest `(s,w)`- path μ_k`(s,w)`, i. e. `(s,w)` is a concatenation of the l-th shortest `(s,v)`- path μ_l`(s,v)` and the path `(v,(v,w),w)`, where l = `deflab[w][k][3]` and `v` = `deflab[w][k][2]`, i.e. μ_k`(s,w)` = μ_l`(s,v)` $\oplus$ `(v,(v,w),w)`.	(8)
`n_deflab[w]` - the number of determined definitive labels of the vertex `w` $\in V$.	(9)

When algorithm starts, we set initially `n_deflab[s]=1`, `deflab[s][1][1]=0`, `deflab[s][1][2]=0` and `deflab[s][1][3]=0` for the starting vertex `s`. Further we set initially for all `w` $\in V$ such that `w`$\neq$`s` `n_deflab[w]=0` (no definitive label is assigned to `w`) and we consider all labels `deflab[w][j][1]`, `deflab[w][j][2]` and `deflab[w][j][3]` to be undefined for all vertices `w` $\in V$ such that `w`$\neq$`s` and `j`$\neq$`1`.

Labels `deflab[w][k][1]`, `deflab[w][k][2]` and `deflab[w][k][3]` will be calculated step by step in course of algorithm work until `deflab[f][K][1]`, `deflab[f][K][2]` and `deflab[f][K][3]` are determined for the final vertex `f`.

The proposed algorithm will make use of an auxiliary set E of ordered quadruples of the form `(wtemp,ttemp,xtemp,ktemp)` where `wtemp` is a vertex `wtemp` $\in V$, `ttemp` is the length of the `(s,wtemp)`- path which was obtained by concatenation of the `ktemp`-th shortest `(s,xtemp)`- path $\mu_{ktemp}(s,xtemp)$ and the path `xtemp,(xtemp,wtemp),wtemp`, i.e.

$$\mu_{ktemp}(s,xtemp) \oplus \texttt{(xtemp,(xtemp,wtemp),wtemp)}.$$

The set E will contain initially exactly all the quadruples of the form `(w,c(s,w),s,1)` where `(s,w)` `FSTAR(s)`, i.e.

$$E = \{(w, c(s, w), s, 1 \mid (s, w) \in A\}$$

Now E contains all the quadruples `(w,t,s,k)` where `w` is a final vertex of a `(s,w)`- path -namely the path `s,(s,w),w`; where `t` is the length of this path, `s` is the last but one vertex of this path, `k=1` and this path is concatenation of k-th shortest `(s,s)`-path (i.e. shortest `(s,s)`-path) $\mu_k(s,s) = \mu_1(s,s)$ and `s,(s,w),w` = $\mu_k(s,s) \oplus$ `(s,(s,w),w)`.

The essential idea of the proposed algorithm is as follows:

While n_deflab[wmin]<K and E ≠ ∅ repeat the following procedure:

Start procedure.

Extract from E a quadruple (wmin,tmin,xmin,kmin) having the minimum value of the second item tmin in E. This quadruple determines a (s,wmin)- path $\mu_{\text{kmin}}(s,x \text{ min}) \oplus$ (xmin,(xmin, wmin),wmin) with the length tmin and last but one vertex xmin.

If n_deflab[wmin] < K, we have just found the subsequent shortest (s,wmin)- path with the length tmin.

In this case:

1. Set: n_deflab[wmin]= n_deflab[wmin]+1;
   k=deflab[wmin];
   deflab[wmin][k][1]= tmin;
   deflab[wmin][k][2]= xmin;
   deflab[wmin][k][3]= kmin;

2. For all vertices w ∈ V^+(wmin) such that n_deflab[w]< K create the quadruple (w, tmin+c(wmin,w), wmin, k)

3. a) Examine whether the path μ_k(s,wmin) contains the vertex w. If yes, the walk

   μ_k(s,wmin) ⊕ (wmin,(wmin,w),w)

   contains a cycle and therefore it is not a path. In this case do nothing.

   b) If the path μ_k(s,wmin) does not contain the vertex w, the walk

   μ_k(s,wmin) ⊕ (wmin,(wmin,w),w)

   is a (s,w)-path with the length tmin+c(wmin,w), with the last but one vertex wmin and with the rank of the label of the last but one vertex equal to k.
   In this case insert the quadruple (w, tmin+c(wmin,w), wmin, k) into the set E.

End procedure

The k-th shortest (s,f) path $\mu_k(s,f) = (s \equiv v_1, (v_1, v_2), v_2, ..., v_{l-1}, (v_{l-1}, v_l), v_l \equiv f)$ can be calculated by making use of labels deflab[][][] as follows:

v_l = f;
k_l = k;

v_{l-1} = deflab[v_l][k_l][2];
k_{l-1} = deflab[v_l][k_l][3];

v_{l-2} = deflab[v_{l-1}][k_{l-1}][2];
k_{l-2} = deflab[v_{l-1}][k_{l-1}][3];
. . . . . .
v_2 = deflab[v_3][k_3][2];
k_2 = deflab[v_3][k_3][3];

v_1 = deflab[v_2][k_2][2];

The same procedure can be used in step 3. a) for checking whether the path μ_k(s,wmin) contains the vertex w.

4. The complexity of proposed algorithm

Recall that n = |V|, m = |A|, suppose m>n. A quadruple (w,t,x,k) can be inserted into the set E at most K.m times. In the case that E is organized as a Fibonacci heap all insertions require O(K.m) steps. Before every insertion a cycle occurrence check is necessary. A single cycle occurrence check requires O(n) steps; all cycle occurrence checks will require at most O(K.m.n) steps. Every quadruple can be extracted from E at most once, a single extraction requires (in the case of Fibonacci heap) O(log(K.m)) steps, all extractions will need at most O(Km log(Km)) steps. So the complexity of proposed algorithm is O(Km(log(Km)+n)). Let us remark that we get the same complexity if E is organized as a binary heap.

5. Modification of algorithm for directed acyclic graphs – DAGs

There are many applications where the studied digraph $G = (V, A, c)$ is a directed acyclic graph – DAG. For example – the underlying digraph for CPM and PERT methods is DAG, the digraph used for optimum bus and/or train connections search is also DAG. Since there are no cycles in DAGs step 3. a) is not necessary – it can be skipped – and therefore the complexity of the proposed algorithm is O(Km log(Km)).

Let's remark that omitting the step 3. a) in the proposed algorithm leads in a digraph which is not acyclic to an algorithm which computes K shortest walks.

6. Conclusion

The proposed algorithm was used for bus and/or train connection search problem with great success. Our further plans are to examine the efficiency and performance of this algorithm in huge general graphs and digraphs, where step 3.a) may cause considerable slow down.

Acknowledgement

The author is pleased to acknowledge the financial support of the Scientific Grant Agency of the Slovak Republic VEGA under the grant No. 1/0135/08.

References

[1] GOTTHILFA, Z., LEWENSTEIN, M.: *Improved algorithms for the k simple shortest paths and the replacement paths problems,* Information Processing Letters, Vol. 109, 7/2009, Pages 352–355
[2] MARTINS, E., Q., W., PASCOAL, M., M., B., SANTOS, J., L., E.: *A New Implementation of Yen's Ranking Looples Path Algorithm*, Investigacao Operacional, 2000
[3] PLESNIK, J.: *Graph Algorithms* (in Slovak), VEDA Bratislava, 1983
[4] YEN, J., Y.: *Finding k Shortest Loopless Paths in a Network,* Managements Science, 17:712 760, 1971
[5] YEN, J., Y.: *Shortest Paths Network Problems,* Mathematical Systems in Economics, Heft 18, Meisennheim am Glan, 1975.

Karol Grondzak – Penka Martincova *

PARALLEL BACKTRACKING ALGORITHM FOR HAMILTONIAN PATH SEARCH

The speed of calculations is a common problem to tackle in many areas of scientific research and real life. This paper presents an implementation of a parallel backtracking algorithm. The performance of the proposed algorithm is demonstrated on the problem of Hamiltonian Path search. Obtained results exhibit significant improvement of the parallel algorithm over the sequential one. Different aspects of parallelization of backtracking algorithm are studied and presented.

1. Introduction

Recently, there are still many scientific problems unsolved. One of the areas with many challenging problems is graph theory. Challenge lies in the speed of the used algorithms. Many of them are proven to be NP-complete. For practical usage any possibility of speed-up is appreciated [1], [2].

A lot of problems solved by graph theory can be characterized as combinatorial problems. To solve them, combinatorial search algorithm is quite commonly applied. It can be described as a process of searching a finite mathematical structure (usually set) for a solution satisfying given criteria [3].

To define combinatorial search algorithms, let us consider a discrete set X, a function $F: X \rightarrow \Re$, and a set of feasible solutions S, where $S \subseteq X$. The feasible solution is defined in terms of given constraints specific for a particular problem. Generally combinatorial search algorithms can be divided into two sets.

The first is the set of optimization algorithms [4]. The goal of these algorithms is to find the feasible solution $x \in S$ such that its value of function F is extreme. To accomplish this task, the set S must be constructed, evaluating all the elements of the set X and checking given constrains. Simultaneously while constructing the set S, values of the function F are calculated for all its elements. Then the element with an extreme (e.g. either minimal or maximal, depending on the character of the problem) value of the function F is the solution of the problem.

The second is the set of solution finding problems [4]. The task is again to iterate through the elements of set X, but in this case the goal is to find at least one element of the set S. In other words, we are looking for an element $x \in X \wedge x \in S$.

For many practical problems the set X is large, which leads to long execution times of programs solving these problems. To reduce the execution times, many approaches were proposed and implemented. Recently parallelization of the task is a common technique to be applied [5].

2. Distributed Backtracking algorithm

While designing the algorithm to solve a combinatorial search problem, a backtracking approach can be used [6]. It is based on the sequential construction of the elements of the set X and evaluation of their feasibility. During this process the elements which are identified as not to produce a feasible solution are pruned. This can significantly reduce the computational time.

To formalize the backtracking approach, let us consider n sets:

$$E_k = \left\{e_1^k, e_2^k, \ldots, e_{m_k}^k\right\}, k = 1 \ldots n\,, \tag{1}$$

where m_k is a number of elements of the set E_k and e_i^k is i-th element of the set E_k. Then the above mentioned set X is defined as Cartesian product of the sets E_i, $\forall i$:

$$X = E_1 \times E_2 \times \cdots \times E_n\,, \tag{2}$$

with cardinality $Q = \prod_{i=1}^{n} m_i$. It is obvious that the set X consist of n-tuples:

$$\left(x_1^c, x_2^c, \ldots, x_n^c\right), c = 1, 2, \ldots Q\,, \tag{3}$$

such that $x_j^c \in E_i$. These n-tuples are constructed recursively extending the set of $(n-1)$-tuples.

* **Karol Grondzak, Penka Martincova**
Department of Informatics, Faculty of Management Science and Informatics, University of Zilina, Slovakia, E-mail: Karol.Grondzak@fri.uniza.sk

The backtracking algorithm starts with an empty n-tuple. In the stage i the $(i-1)$-tuple is extended using elements of the set E_i. Newly obtained i-tuples are then checked for feasibility and then expanded to $(i+1)$-tuples to form the set E_i+1.

This process is actually a depth-first search of a search-tree. Nodes of the search tree at i-th level consist of i-tuples. An example of the search-tree is in Fig. 1.

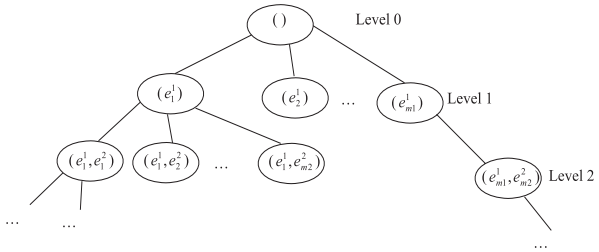


Fig. 1 Example of a search tree

It is obvious, that just described the backtracking algorithm is suitable for distributed processing. One strategy to distribute tasks among processors could be as follows. Let us construct all i–tuples for some small value of i, chosen to correspond the amount of available processors. Then assign those i-tuples to processors, such that each processor will perform depth-first search of a sub-tree of the search tree starting from a given i-tuple.

The properties of the search-tree and the strategy of search can significantly influence the performance of the backtracking algorithm [7]. When constructing the backtracking algorithm, we do not have usually any a priori information about the structure of the search-tree.

In any stage of the backtracking algorithm, we perform depth-first search of some sub-tree. If there was a solution found in the searched sub-tree, the algorithm finishes. If not, then the backtracking algorithm has to choose another sub-tree to search.

To formalize the properties of a search-tree from the point of view of the backtracking algorithm, let us denote depth of the sub-tree:

$$D_i^k = \sum_{l=1}^{m_k} D_i^{k+1}, k = 1, 2, \ldots n; i = 1, 2, \ldots m_k, \qquad (4)$$

where n is the number of sets used to construct n-tuples and m_k is the number of elements in the set k. Depth of a sub-tree is a function of search strategy and represents a number of n-tuples to construct and process during depth-first search of that sub-tree.

3. Hamiltonian Path and Circle

One of the well-known problems of graph theory is the problem of finding a path on a graph which visits each of the nodes exactly once. If the starting and final nodes are different, the problem is known as the Hamiltonian Path problem [6]. Special case when starting and final nodes are identical is known as the Hamiltonian Circle problem.

It is known that both the Hamiltonian Path and Hamiltonian Circle are NP-complete problems. We can characterize them as decision problems – for a given graph the goal is to determine if the Hamiltonian Path or Circle exists.

Among many problems of the Hamiltonian Circle problem let us mention a well-known Knight's Tour problem. The goal is to find a path of knight on a chessboard of a standard dimension $N = 8$, such that will visit all the squares, each exactly once. This problem can be generalized to chessboards of any dimension.

```
WHILE workers not finished
DO
  wait for a message from worker
  IF solution found
    notify all workers
    finish
  ELSE
  IF request for a job
    send job to worker
END
```

(a)

```
REPEAT
  ask for a job from master
  IF there is a job
    process assigned job
    IF solution found
      notify master
  UNTIL there is a job
```

(b)

Fig. 2 Pseudo-code of master (a) and worker (b) processes

3.1 Distributed Hamiltonian Path Search Algorithm

As mentioned above, the Hamiltonian Path problem is a decision problem. The backtracking combinatorial search algorithm can be applied to solve it. The problem is, that even for a relatively small dimension of the graph, the size of the n-tuples to be searched is huge. Let us consider the problem of Knight's Tour problem for a chessboard of 64 squares. First let us number the squares of the chessboard starting form 1 to 64. Our goal is to construct a n-tuple containing a number of squares visited during the tour. On each of the squares, the knight has at most eight possible ways to move. So there are 8^{64} tuples to be searched. This is a rough estimation and many of these tuples can be pruned during the depth-first search of the search-tree. But still there are many of tuples to be checked. There is too much work for a single processor.

To improve the performance of the backtracking algorithm for search of the Hamiltonian path, several processors can be involved. We can expect linear increase of the performance, but for some situations even super-linear increase was reported [7]. It depends on the properties of the search-tree of the solved problem.

Parallelization is quite straightforward, because of the nature of the search-tree. On each level of the search tree, there are nodes to be searched. Let us denote a number of nodes on each level as

$$Q_i = \prod_{k=1}^{i} m_k, \qquad (5)$$

where m_k is the number of elements of the set E_k. The nodes form a set of disjunctive sub-trees. These sub-trees can be assigned to at most Q_i processors to perform parallel search on them.

For a given amount of P processors it is reasonable to determine a level of the tree to start parallel search such that:

$$Q_{i-1} < P <= Q_i . \quad (6)$$

Then the maximum amount of processors is involved in search. In fact, when $P = Q_i$, all the processors will search exactly one sub-tree. In other cases some of the processors will search several sub-trees. This situation can lead to better load-balancing comparing the case when $P = Q_i$, because usually the sub-trees are of different depth. Then if all the sub-trees are assigned at the beginning of the algorithm, those processors which finish their task are sitting idle waiting for those processors, whose sub-trees are deeper and require more time to finish the search.

4. Experiment and obtained results

To test the proposed distributed algorithm, we have modeled the following routing problem. Let us consider a regular rectangular mesh of cells of the dimension *N*. In this mesh, we have to find a route from a given start point to a given finish point such that each of the cells will be visited exactly once. This problem can be described in terms of graph theory as a problem of finding a Hamiltonian path in a graph of a special regular form (Fig. 3).

This problem can represent a task to route some small maintenance vehicle (e.g. robot) on a set of office cubicles to perform daily routine tasks (e.g. cleaning, etc.). It can also model the route of some machine in a factory to perform a prescribed operation in different parts of some product, e.g. to drill holes, etc. Once the route is found, it can be embedded into a device, which will then use this fixed route regularly. Or in case when the layout is subject to change, the algorithm can be implemented in the device. When the situation changes, the device will apply the algorithm to find a new route corresponding the actual situation.

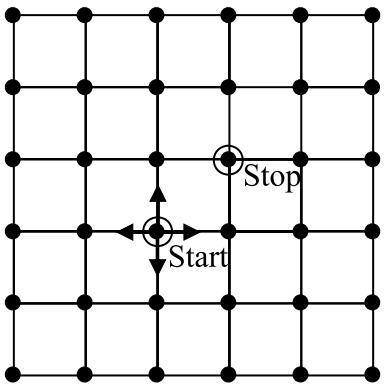


Fig. 3 Example of graph representation of solved problem

The algorithm mentioned in section 3, was implemented using OPENMPI Project [8], [9]. It is freely-available, high performance implementation of Message Passing Interface (MPI) standard. MPI is a standard for communication and synchronization of distributed entities. It was proposed by a consortium of companies for high performance on both massively parallel machines and on workstation clusters.

The presented results were obtained on a commodity workstation cluster. The cluster consists of twenty personal computers equipped with a processor Intel Core 2 Duo and 1024MB of RAM memory. Computers are interconnected by 100Mb/s local area network. The connection is sufficient because of the small communication requirements of the application.

The performance was studied for problems of the dimension $N = 6$. The obtained results are summarized in the form of tables. Each table presents data for different study. Data in each table represent amount of iterations necessary to find a solution. This representation was preferred to the time measurement because it represents the performance of the algorithm and is not influenced by the actual load of processors. It corresponds to the depth of the search tree when considering the starting level 0, as it was defined above (4).

Three different characteristics of the problem were studied. First, we studied the influence of the chosen depth-first search strategy to the performance of the algorithm. When solving backtracking problems, the strategy of the next step choice is embedded in the code. It is obvious that for the solved problem there are at most four different directions to continue from a given position. Let us denote those directions L(eft), R(ight), U(p) and D(own). So there are 24 different strategies to choose from, considering a different order of directions (e.g. LURD is one strategy, ULDR is another). Because of some symmetry of the solved problem, the obtained results are same for couples of strategies. This is the reason, why there are only 12 results presented, other 12 results are the same (row Direction in Table 1). When comparing results for different strategies it can be seen that the choice of strategy has significant impact on the performance (Table 1, Table 2 Table 3). For some strategies the number of iterations is significantly smaller than for other ones. There is no a-priori information about the behavior of the strategies, so usually when designing the backtracking algorithm, we randomly choose one of the available strategies.

Secondly, we studied the performance improvement with respect to the number of processors involved in calculation. It can be seen (Fig. 4) that the obtained results are highly non-linear. This figure shows the number of iterations needed to find the solution of the problem. The smaller number of iterations means the better performance. The results are presented for strategy number 4 (DULR). It indicates that the search tree is unbalanced. It is also indicating that a better load-balancing strategy should be applied to distribute load among the nodes.

Third, the study assessed the influence of the depth at which the search starts. It can be seen that the number of iterations needed

Number of iterations and relative speed-up for different strategies for search started in level 1. Table 1
Number of processors involved for parallel search is 5

Strategy	0	1	2	3	4	5	12	13	14	15	16	17
Direction	DLRU	DLUR	DRLU	DRUL	DURL	DULR	RLDU	RLUD	RDLU	RDUL	RUDL	RULD
Sequential	240	821958	9638	4353	101981	2515876	908134	138265	264258	8476	204707	6597
Parallel	240	204	2018	3486	19481	915	24285	138265	24321	858	81255	6597
Speedup	1	4029.2	4.8	1.2	5.2	2750.0	37.4	1.0	10.9	9.9	2.5	1.0

Number of iterations and relative speed-up for different strategies for search started in level 2. Table 2
Number of processors involved for parallel search is 17

Strategy	0	1	2	3	4	5	12	13	14	15	16	17
Sequential	240	821958	9638	4353	101981	2515876	908134	138265	264258	8476	204707	6597
Parallel	240	204	2018	3486	19481	915	1136	457	1108	616	559	457
Speedup	1	4029.2	4.8	1.2	5.2	2750	799.4	302.5	238.5	13.8	366.2	14.4

Number of iterations and relative speed-up for different strategies for search started in level 3. Table 3
Number of processors involved for parallel search is 38

Strategy	0	1	2	3	4	5	12	13	14	15	16	17
Sequential	240	821958	9638	4353	101981	2515876	908134	138265	264258	8476	204707	6597
Parallel	240	204	3488	1350	409	915	1136	457	1108	858	559	457
Speedup	1	4029.2	2.8	3.2	249.3	2750	799.4	302.5	238.5	9.9	366.2	14.4

to find a solution is either the same (strategies 0, 1 and 5) for all the starting depths, or is improving with the increasing starting depth (strategies 3, 4, 12, 13, 14,16 and 17). For the rest of strategies (2 and 15) the number of iterations to find solution has increased for level 3. It could be caused by the fact, that only 38 processors were involved in the calculation instead of 64, which is the number of sub-trees in level 3. When using at least 64 processors, we would expect to get either the same or better results for strategies 2 and 15.

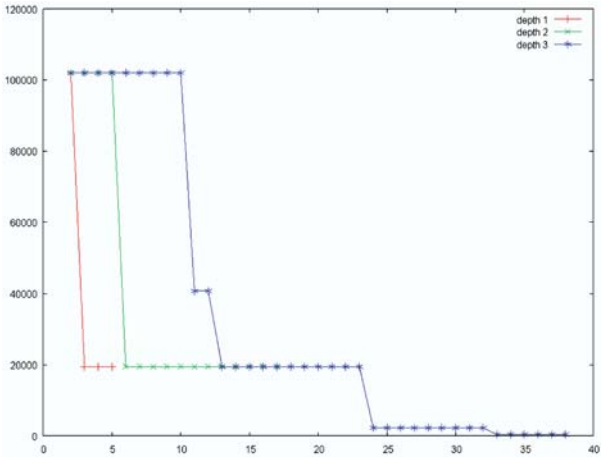


Fig. 4 Number of processors versus number of iterations needed to find solution.

Fig. 4 demonstrates also the influence of the depth at which the search starts to the performance of the algorithm. This graph presents the dependence of the number of iterations needed to find a solution with respect to the number of processors for different starting depths. It demonstrates that when the start depth is increasing, the number of iterations decreases (depth 1 curve versus depth 3 curve).

The smaller the number of iterations, the better performance of the algorithm

5. Conclusion and Future Work

Parallel paradigm is recently a very popular approach for solving time-consuming scientific problems. It is widely adopted in many scientific areas starting from scientific calculations, through modeling [10] up to computational biology [11].

In this paper, the study of the parallelization of the backtracking algorithm was presented. The general parallel backtracking algorithm was proposed and implemented using MPI implementation OPENMPI. It was tested on the problem of Hamiltonian path search.

The obtained results are in accordance with our expectation [12]. This study helped the authors to better understand the properties of the parallel backtracking algorithm.

We can expect that the performance can be improved by involving more processors into calculation. The obtained results also indicate that the better load balance is achieved when the search starts deeper in the search tree.

The future work will be to modify the algorithm for grid environment. The potential for improving the performance of algorithm is in grid technologies. Study of different load-balancing techniques to the performance of the backtracking algorithm will also be performed.

Acknowledgement: The authors would like to thank Dr. Michal Kaukic for the opportunity to use the laboratory of parallel applications. This work was partially supported by VEGA Grant No. 1/0761/08 "Design of Microwave Methods for Materials Nondestructive Testing" and VEGA Grant No. 1/0796/08 "Large Data Modeling and Processing".

References

[1] GENDRON, B., CRAINIC, T. G.: *Parallel Branch and Bound Algorithms: Survey and Synthesis,* Operations Research, 42, pp. 1042–1066, 1994.
[2] CRAINIC, T. G., LE CUN, B., ROUCAIROL, C.: *Parallel Branch-and-Bound Algorithms,* Wiley Interscience, October 2006, ch. 1.
[3] MEZMAZ, M., MELAB, N., TALBI, E-G.: *A Grid-enabled Branch and Bound Algorithm for Solving Challenging Combinatorial Optimization Problems,* In Proc. of 21st IEEE Intl. Parallel and Distributed Processing Symp., Long Beach, California, 2007.
[4] QUINN, M. J.: *Parallel Programming in C with MPI and OpenMP.* Mc Graw Hill, 2003. ISBN 007-123265-6.
[5] WILKINSON, B., ALLEN, M.: *Parallel Programming Second Edition.* Pearson Education, 2005. ISBN: 0-13-140563-2.
[6] KUCERA, L.: *Combinatorial algorithms (in Slovak)*, SNTL, 1989.
[7] NAGESHWARA RAO, V., KUMAR, V.: *On the Efficiency of Parallel Backtracking,* IEEE Trans. Parallel Distrib. Syst. 4(4): 427–437 (1993).
[8] http://www.open-mpi.org/
[9] GROPP, W., LUSK, E., SKJELLUM, A.: *Using MPI,* Second Edition. The MIT Press, 1999. ISBN: 978-0-262-57134-0.
[10] KVASNICA, I., KVASNICA, P., IGAZOVA, M.: *Parallel Modeling in Computer Systems,* Acta Avionica, Vol. X, 2008, 16. ISSN 1335-9479, pp. 8–86.
[11] SCHMIDT, M. C., SAMATOVA, N. F., THOMAS K, PARK, B. H.: *A scalable, parallel algorithm for maximal clique enumeration,* J. Parallel Distrib. Comput. 69 (2009), pp. 417–428.
[12] GRONDZAK, K., MARTINCOVA, P., CHOCHLIK, M.: *Performance Analysis of Parallel Algorithm for Backtracking,* Proc. of 4th International Workshop on Grid Computing for Complex Problems, Bratislava, 2008.

Peter Matis *

HEURISTICS FOR THE SOLUTION OF A VERY LARGE STREET ROUTING PROBLEM WITH MIXED TRANSPORTATION MODE

Servicing a large number of customers in a city zone is often a considerable part of many logistics chains. This problem is called a Street Routing Problem (SRP). As presented, only using systems such as Geographical Information Systems (GIS) it is possible to effectively manage SRP. New heuristic for solving a very large SRP is evaluated on the real data. One of the key properties of GIS for use with the routing software is its flexible interactive and user-friendly environment. The paper presents several approximations of length for SRP with mixed transportation mode.

Keywords: SRP, VRP, GIS, heuristics, approximations

1. Introduction

The Street Routing Problem (SRP), as a problem of servicing a large number of customers in a city zone or several connected cities is often a part of many logistics chains. In this paper we will mostly address a special case of SRP that is a postal service delivery. A typical SRP uses samples with thousands of customers. We want to evaluate and find solutions for cases with more than 10 000 customers. In such agglomerations service is done using a mixed transportation mode. The service personnel is driven by a car or uses public transportation to a first point of the service area, then serves the area usually on foot and then returns to the depot using either public transportation or a car. In this paper we will focus on the transport of personnel by cars. To find a good solution for these cases we must be able to approximate the length of SRP. We need to aggregate customers to natural clusters and solve a special case of the Capacitated Vehicle Routing Problem (CVRP) for finding good routes of cars.

2. Experimental sample

The SRP is a less explored group of problems than the Vehicle Routing Problem (VRP) and there is not a good, publicly available sample of data for the method comparison. Authors, for example [7], are usually solving specific problems and it is difficult to compare solutions across these problems because each of them is somewhat unique.

It is necessary to use our own experimental sample, one that covers most of the typical SRP and, at the same time, it is possible to compare each of the solution methods. Our experimental sample was from four large cities and surrounded villages in Slovakia. Customers were houses in these cities and full street infrastructure is available. These cities were Bratislava, Kosice, Zilina, Presov. A number of customers for these four agglomerations vary from 12 000 to 29 000.

The data were collected manually from the source maps ZM 1:10 000, purchased from the Geodetic and Cartographic Institute of the Slovak Republic. Some data were collected using GPS receivers. These data are not publicly available at the moment.

3. Definition of a very large SRP with mixed transportation mode

There are cases of serving a large number of customers from one central depot. Centralizing all delivery operations to one depot for the large number of customers has some advantages and disadvantages. One of the key advantages to centralization is the ability to use more advanced and expensive technologies for preparation before the deliveries are picked up by the postmen. The preparation of deliveries usually takes more than 10 % of the total working time of postmen. By shortening this time combined with good transportation methods for placing the postmen in their districts we allow more time for a delivery.

Centralizing delivery operations creates a large aggregation of customers that need to be served from one depot. In Fig. 1 we present a possible area of the served region around the city of Zilina with 12 663 customers.

Our goal is to find a good solution for the SRP in such a large, centralized region. For each customer we have an average service time. Postmen can be driven by car to the starting point of the

* **Peter Matis**
Department of Transport Systems, Faculty of Managements and Informatics, University of Zilina, Slovakia, E-mail: Peter.Matis@uniza.sk

Fig. 1: Customers and depot in large Zilina region

service and at the end of the working day they can be driven back to the depot. Deliveries are made on foot, so the problem represents a special case of SRP with a mixed transportation mode. The car driver is also a postman and serves one district. There are no transportation related expenses. The delivery time is limited for each postman. Time gaps between the physical end of deliveries and the car arrival to drive the postman back should be tide.

4. The aggregation of customers to natural clusters and approximation of a route length

One of the key problems in solving a very large SRP with a mixed transportation mode is to aggregate customers to natural clusters. Each cluster is then treated as one district for the SRP and is served by one postman. The size of the cluster is determined by an estimated length of the SRP route, the travel speed of the postman, the delivery time for customers in the cluster and the total time available for serving the cluster.

There is one problem related to the fact that one logic city unit (vicinity village or city district) can be divided to several service districts and the last service district can take only a small amount of the postman's working time. It is hard to merge these residuum districts to one district, because they can be distributed across the whole studied area.

For aggregation of customers to natural clusters we used a part of Fuzzy Cluster Heuristics (FCH) [5]. This heuristics can be implemented in the following steps:

1. Estimate the minimum number of clusters p needed for serving all the customers using the following formula

$$p = \downarrow \left(\frac{\sum_{c=1}^{N} CST_c + \frac{\sum_{s=1}^{M} D_s}{V}}{SWT} \right) \quad (1)$$

   where symbol $\downarrow$ represents nearest smaller integer number, CST_c is a service time for the customer c, N is a number of customers, M is a number of street segments where customers are located, D_s is a length of street segment s, V is an average speed of the postman walking on foot, SWT is the available working time of one postman for service of customers.
2. Locate p medians, so they are uniformly distributed in the serviced area.
3. Create p clusters of customers around these medians using clustering by "fuzzy c-means" (FCM). The membership of customers in each cluster is set as a triangle fuzzy number. The triangle fuzzy number is based on route distance between the cluster median and customer.
4. For each cluster approximate the route length and estimate the average service time.
5. If there are many clusters (more than 20%) which have their service time over the time SWT, increase the number of clusters p and go to step 2, otherwise end the algorithm with the resulting clusters and these are used as SRP districts.

In this algorithm, we used an approximation of length for the SRP route of one postman. In our case this is actually the length of TSP or special TSP, where the postman does not necessarily need to return back to the point where his route started.

In literature there are some good samples as to how other authors estimate the length of TSP or VRP [1]. We present here two estimations from Toth, Vigo and Kwon et al.

$$TSP(D) = 0.765\sqrt{NA} \quad (2)$$

$$TSP(D) = \left[0.83 - 0.0011(N+1) + \right.$$
$$\left. + 1.11S/(N+1)\right]\sqrt{NA} \quad (3)$$

where $TSP(D)$ is the estimated length of the route, A is an area of a serviced cluster, S is a proportion of the length-width ratio for the outside rectangle of the cluster created such a way that the ratio is always larger than or equal to 1. The authors always use the Euclidean or Manhattan travel metric.

In our case neither Euclidian nor Manhattan metrics interpret the real distance for the traveling salesman. Our case is specific because the authors focus on the VRP and our problem is the SRP. The SRP has a couple of differences to VRP [3] and some of these differences are related to a distribution of customers and the network density. In our model we used the following estimation of the length for one postman's path:

$$TSP(D) = 1.31*SD \quad (4)$$

where *SD* is a total distance of all the street segments that are served by one postman. This formula was created from a simple regression using sample data with 125 service clusters compared to the best results from heuristics described by Matis [5]. Figure 2 represents use FCH in one large SRP case.

Fig. 2: Natural clusters in large Zilina region created by FCH

5. Transportation of postmen to districts

To get a solution for a very large SRP we need to distribute postmen to their service districts using cars. This is a variation of a normal Capacitated Vehicle Routing Problem (CVRP) with addition of an extra condition that we have to maximize utilization of vehicles. An additional condition is that the vehicle has to return to the depot using the same route. In our case all the vehicles have the same capacity of 5 postmen. The goal is to minimize the total length of routes for all the cars and, second, to minimize a number of cars that are necessary for all the service districts that are farther from the service center – the post office building. The car's driver serves the last district and then collects all the postmen that were in his car when they started in the service center. The postman that first leaves the car has more time for delivery than the last postman in the car. Knowing this we could change our algorithm for a creation of natural clusters, and make clusters that are closer to the service center larger and further clusters made smaller. In our case, we did not make this change. The difference in time when they leave the car is usually less than 10 minutes and because the delivery time has a probabilistic character, this may not have affect on the final result.

There is no method that could find an optimal solution for this problem for a practical size SRP. We used Unified Cluster First Route Second (UCFRS) heuristics [5] that was changed to this specific case where our goal was not only to find the minimum total length, but also to maximize the utilization of the car. Heuristics was named as Unified Cluster First Route Second for Postman Collection heuristics (UCFRSPC).

A major difference between the original UCFRS and the UCFRSPC is that a number of serviced clusters in the route is limited by a maximum number of passengers of the car. For the one trip's length calculation we use only lengths of segments from the depot to the last customer. The car has to return back to the depot using the same route. The UCFRSPC heuristics executes the following steps.

1. Mark all the centers of clusters (CC) as not visited.
2. Pick from all not visited CC the furthest one. Set the route's center equal to the position of the first picked CC.
3. If the number of CC in the current route is equal to the capacity of a car go to step 5.
4. Choose the closest CC to the route's center (using road distances). Put this CC to the current route and recalculate the position of the route's center as the closest point on the transportation network to the gravity point of all the CCs in the current route. Mark the CC as visited. Go to step 3.
5. If there are some not visited CC then change the route number to a new number and go to step 2, otherwise calculate the minimum total length for every route. The sum of these lengths is the total length of all the car trips.

Fig. 3 shows the resulting routes for 40 CC divided to 8 cars.

We calculate the minimum total length for every route evaluation of all permutation of possible orders of CC in the route. We can do it, because the size of the CC set is a small, usually there are 5 CC in one route.

6. Use of GIS as decision support system for the SRP

Recent development in the real street routing problems shows that there is a need to make routing software a part of a larger system. One of possible solutions to this is to integrate routing

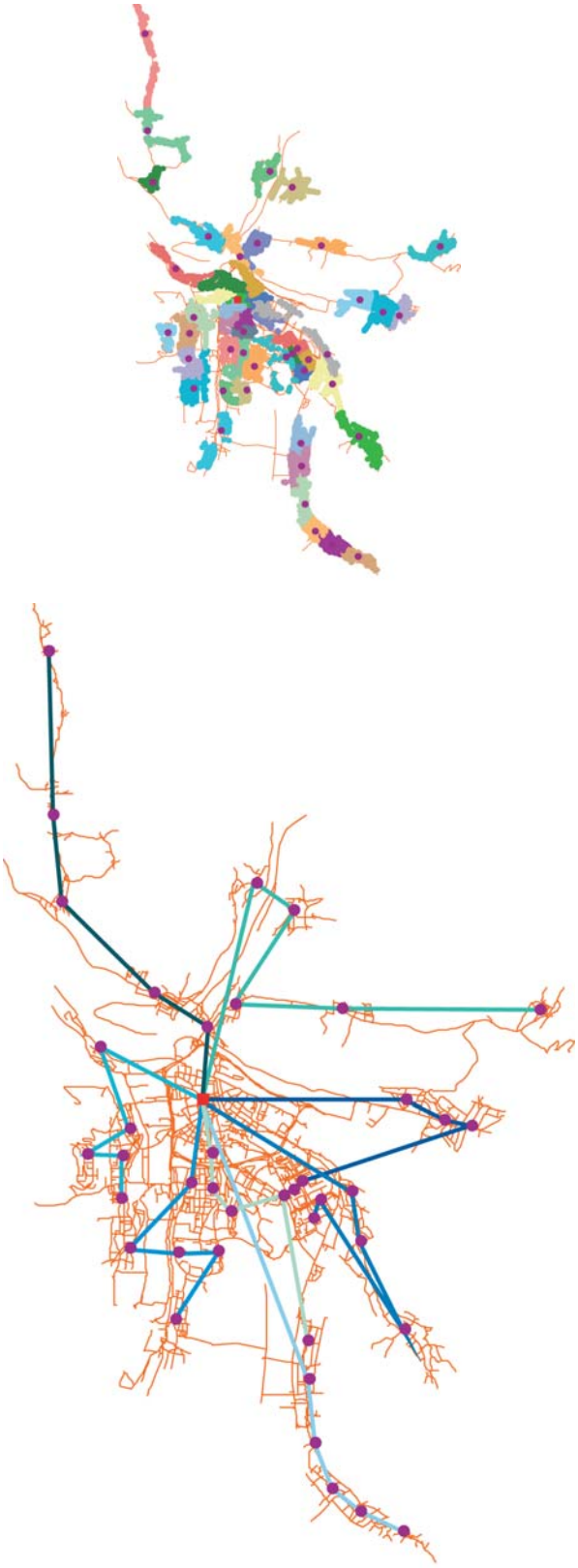


Fig. 3: Centers for natural clusters in the large Zilina region served by cars and optimal routes for 8 cars

software within GIS. GIS can be helpful in the collection, storing and management of large geographical databases used in the routing software. GIS can also be used for the creation of all outputs from the routing software. One of the key properties of GIS for the use of routing software is the interactive and user-friendly environment. Routing software can find a good solution and explore possibilities and an expert can change calculated routes to explore other possibilities based on the expert's judgment.

GIS has several useful features that could help improve the routing software performance. To mention only a few here – we may pay attention to the capabilities as safe database management, flexible symbols, map management, drawing capabilities, safety and interoperability.

Decisions support systems (DSS) are important for solving real SRPs. Software can be used for a solution of the problem, and also for exploring several other possibilities, or seeing how changes in the transportation network, regulations or policies could affect the routes, expenses and other parameters. Currently there exist two trends in the development of DSS for SRP.

- Independent software packages specialized for SRP with limited amount of DSS capabilities. Here we can note software like GeoRoute [8] from the Canadian software firm GIRO. It is not an open system. It is an expensive product that costs 1,000,000 EUR for one installation. As an example of other such systems, we can use TRANSCAD [9] created in the US, which has some similarities to GeoRoute. GeoRoute is more feasible for solving the SRP because it is specialized for the street routing. TRANSCAD is more specialized in the node routing.
- Integrated systems based on the GIS or CAD. As an example, we can cite ArcView [10] with its ArcGIS Network Analyst. It specializes in the node routing. The user can implement extension and then make it an integral part of the whole system.

An important feature of a decision support system is good visualization and a good editor. Tools that are able to visualize results are easily acceptable to users. Visualization also allows users to see any problems or discrepancies which are not easy to find or implement by heuristics. The user can change starting conditions in the problem to avoid these discrepancies, test the new model, and by several iterations, get an acceptable result.

To shorten development time, we have chosen a standard GIS environment as the base system for the data management. The integrator represents the main user interface. It controls each heuristic – the connection of GIS database to/from heuristics. In some cases the heuristics could be replaced by regular solvers, XPRESS for example. We used ArcView by ESRI as the main GIS tool, because it is an open system and it allows the user to program its functionalities. The GIS is managed from the integrator by the Avenue scripting language, C#, and VBasic, depending on the version of ArcView. The GIS is used in the system as a data management tool, editor, for visualization of results.

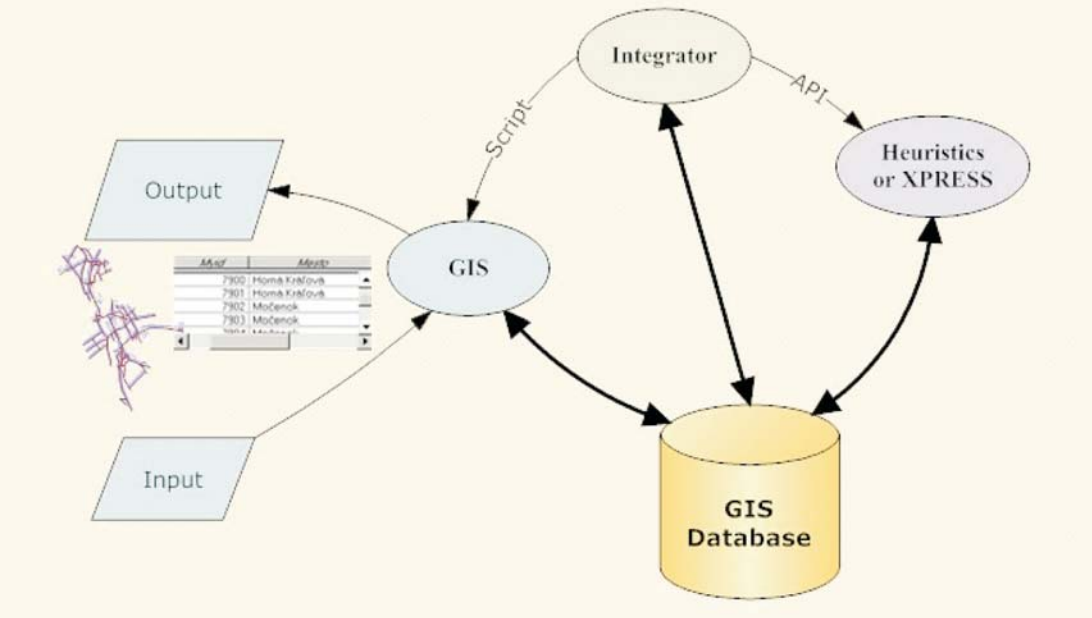


Fig. 4: Decision Support System using of GIS for SRP

7. Conclusions

There are expanding activities in the cities that can be presented as street-based tasks. We introduced some methodology for solving a specific SRP with a very large volume of customers. We used the samples from four cities for verification of our methodology and algorithms. This problem is not documented in the literature.

We presented a new heuristic for creation of natural clusters and estimation of the route's length. We have not compared the results of the heuristics with any other results because there is no literature dealing with this problem. From the small sample we can conclude that an average calculation time for this part of the solution is 232sec using a PC with Dual Core Processor Intel Pentium. For most of the practical applications this is acceptable and results are promising.

We created a new heuristics for the distribution of postmen to their service districts using the cars. The results show that heuristics can be used also for the cases with a heterogeneous vehicle park. This is very fast heuristics.

We created DSS for solving these specific SRP which was tested only on the sample data.

Acknowledgement: This work has been supported by the grant VEGA 1/0591/09.

References

[1] FIGLIOZZI, M. A.: *Planning Approximations to the Average Length of Vehicle Routing Problems with Varying Customer demands and Routing Constraints.* Transportation Research Record: Journal of the Transportation Board, No 2089, p. 1–8.
[2] JANACEK, J.: *Optimization in transport networks (in Czech),* EDIS-ZU, Zilinska univerzita, Zilina, 2003, 248 p.
[3] MATIS, P.: *Management of street routing problems using decisions support system,* Communications 3, Zilinska univerzita, Zilina, 2006, p. 5–8
[4] MATIS, P.: *The relationship between quantitative and qualitative measurements in solving of street routing problems,* 15th Internatinal Scientific Conference on Mathematical Methods in Economics and Industry, Herlany, 2007, p. 144–152.
[5] MATIS, P.: *Decision support system for solving the street routing problem,* TRANSPORT 2008 23(3), ISSN 1648-4142, p. 230–236.
[6] POOT, A., KANT, G., WAGELMANS, A.: *A Saving based method for real-life vehicle routing problems,* Journal of the Operational Research Society, Vol. 53, 2002, p. 57–68
[7] RUIZ, R., MAROTO, C., ALCARAZ, J.: *A decision support system for a real vehicle routing problem,* European Journal of Operational Research 153, 2004, p. 593–606.
[8] GeoRoute, http://www.giro.ca/en/products/georoute/index.htm
[9] TransCAD, http://www.caliper.com/tcovu.htm
[10] ArcView, http://www.esri.com/software/arcview/

Vladimir Pribyl *

SOLUTION OF THE BUS ROUTE DESIGN PROBLEM

The paper deals with a single bus route design problem. It consists of two stages. The first one is to choose the set of stops fulfilling a defined constraint. The second one is the precising of the order of the stops on the route. Both exact and heuristic methods are proposed and verified on 9 randomly generated networks. Very high computational complexity of the exact method and some ways how to reduce it are discussed in the paper. Comparison of the experimental results is presented in the final table.

Keywords: bus, design, network, optimization, route, method, heuristics

1. Introduction

The problem we deal with belongs to the "family" of network reduction problems. The common feature of all of them is that a subgraph fulfilling some constraint has to be found for the given graph. The oldest member of the family is the minimum spanning tree problem, reducing only the edge set and keeping the vertex set unchanged.

A. Cerna et al [2] studied the shortest subgraph not lengthening any important trip more than the given percentage.

A. Cerna [5] introduced the problem of a bus route design in an area with a weak passenger demand. She outlined a basic idea of the exact method and of possible heuristics.

The main purpose of this paper is to present computational experience with exact and heuristic methods applied to randomly generated networks.

The bus route design problem in our formulation is different from the ones described in "classic" paper [6] or the "modern" [1]. E.g. in [1] the basic type of a bus route is "the shortest path between large demand vertex pairs". In [6] it is supposed that there exists a "wide" set of possible routes (constructed, e. g., manually), and the goal is to choose the optimal subset. We shall not use an "absolute" limit of accessibility, since in [3] and [4] it is shown that such an approach leads to extremely expensive or to user unfriendly solutions. We prefer a measure of a "mean" accessibility expressed in the constraint (1). Of course, the used methods can be modified for other accessibility measures.

2. Problem

Let $G = (V, E, q, d)$ be a (non-oriented) graph with the demand function q: $V \to \langle 0; \infty)$ and the length d: $E \to (0; \infty)$. Let $d(u, v)$ be the distance of u, $v \in V$ obtained by extension of the length of the edges. Let $\delta(S)$ be the length of the shortest path connecting the vertices of S on G for each $S \subset V$. Let $\lambda \in (0; \infty)$ and $q = \sum_{v \in V} q(v)$.

The problem is to find $S_{opt} \subset V$ such that

(1) $\mu(S_{opt}) = \frac{1}{q} \sum_{v \in V} q(v) d(v, S_{opt}) \leq \lambda$

(2) $\delta(S_{opt}) \to \min$

Appended problem

If there were several solutions of the problem 2, then the problem would be to take the set S_{opt} with a minimal number $|S_{opt}|$ of elements in it.

Note

The expression $\mu(S_{opt})$ in (1) represents the mean walking distance of passengers to the nearest stops.

The appended problem posses an alternative formulation: In the case of several solutions of the problem 2 first to take the one with the smallest $\mu(S_{opt})$ and only afterwards the one with minimal $|S_{opt}|$. Such a change would need small modification of the exact method of solution whereas the heuristics can remain unchanged.

3. Methods of Solution

Since the problem contains the open travelling salesman problem (OTSP) as a subproblem, it is obviously NP-hard. Therefore, we have to propose some heuristic method for its solution in addition to the exact method we shall start with. It is of the "Depth-First-Search" type.

* **Vladimir Pribyl,**
Faculty of Management, University of Economics, Prague, Czech Republic, E-mail: pribyl@fm.vse.cz

3.1 Exact Method (EM)

Initial step: We find first S fulfilling (1). Then we find a first record $\delta(S)$ solving OTSP.

Recursive step: Once a set S fulfilling (1) is found, each its extension is omitted since it cannot shorten the length $\delta(S)$. Then we look for the next S fulfilling (1) in the adjacent branch of solution structure.

3.2 Neighbor Greedy Heuristics (NGH)

We shall use these denotations:

$N(S') = \{w \in V: w \notin S', \exists\, s \in S', (w, s) \in E\}$ for the neighborhood of S' in G and

$m = m(G)$ for the median of G i.e. such $m \in V$ that $\sum_{v \in V} q(v)d(v,m) \longrightarrow \min$

Initial step: We put $S = S' = \{m\}$

Recursive step: If S fulfills (1), then we consider S the solution. If it does not then we look for such a $s' \in (V - S) \cap N(S')$ that

$$\sum_{v \in V} q(v)d(v, S \cup \{s'\}) = \min_{s \in (V-S) \cap (S')} \left\{ \sum_{v \in V} q(v)d(v, S \cup \{s'\}) \right\}$$

If such a vertex does not exist the heuristics is not able to solve the problem and we stop. If it does then we put $S = S \cup \{s'\}$ and further if $S' = \{m\}$, then we put $S' = \{m, s'\}$, if not (and thus the set S' contains at least two elements), then $S' = S' - \{s''\} \cup \{s'\}$ where $s'' \in S'$ and $d(s', s'') = \min_{s \in S'} \{d(s', s)\}$. After doing that we return to the recursive step.

4. Implementation of the methods and computational experience

Both above mentioned methods were implemented. The environment of the Visual Basic and database system Microsoft Access was used, even though it is highly probable that there are quicker environments. These are the reasons:

- built-in effective and easy to use algorithms and methods for importing, sorting, searching, exporting and presenting the data
- availability
- Visual Basic is a built-in programming environment of the MS Access and it is very simple to work with the Access objects in it

The both methods were tested on 9 randomly generated small networks. The diameter of each network is about 30 km and the number of vertices is 20. All the implemented algorithms were tested on the same hardware configuration.

4.1 Exact Method

The Exact Method consists of two basic stages. Each stage was implemented separately in order to gain possibility to work with intermediate data. It is very important for finding ways how to reduce the time consumption of the exact method.

The first stage of the method is the algorithm which creates the set of all subsets $S \subset V$ fulfilling the constraint (1). The number of all possible combinations of vertices is 2^n, where n is the number of vertices of graph G. Even for our small test networks it is more than 1 million combinations. The algorithm is of the "Depth-First-Search" type as mentioned above. Since it implements the principle described in 3.1, it is not necessary to test the constraint (1) for all the possible combinations of vertices. For our networks with 20 vertices, the number of tested combinations is about 300 thousands and finally the number of all subsets $S \subset V$ fulfilling the constraint (1) is about 150 thousands. It varies in accordance with the network topology and the value of the limit λ of course. The time consumption of this algorithm was several minutes for all our tested networks. But it is necessary to see that the computational complexity of this algorithm is $O(2^n)$ and, therefore, it can take tens of hours to finish this stage of the EM for the network with, say, 30 vertices.

The second stage of the method is the algorithm for solving the OTSP for each subset S found in the previous stage of the method. It finds the set S_{opt} fulfilling the constraint (2). The exact method for solving the OTSP on the set S is going through all the permutations of vertices in the set S in order to find $\delta(S)$. The computational complexity of this algorithm is $O(n!)$, where n is the number of vertices in S. The computational time of this algorithm is about 3 seconds for $n = 10$. However for one of our test networks (having set $\lambda = 4$) there are about 5000 subsets S fulfilling the constraint (1) with the $n \in \{11;12;13\}$. It means that solving the OTSP in this way can takes about 86 hours only for these 5000 subsets. The main goal of the EM is to obtain the optimal solutions of our problem. These are the platforms for testing and improving proposed heuristics. Computational times about 100 hours are unacceptable for this purpose. Therefore some ways to shorten the computational time were examined. These are the analyzed and tested ways:

- **Reducing the number of subsets *S*** fulfilling the constraint (1). The principle is to find such subset S fulfilling (1) that there exists another subset S' fulfilling (1) such that $S' \subset S$. Considering the principle mentioned in 3.1 it is possible to omit such subsets S. The experimental computations carried out on the test networks made out that it is possible to reduce the number of subsets S approximately two times. Unfortunately, this method is relatively time consuming. The computational time was several hours for our test networks. This is the reason why this method is not finally used in the EM.
- **Speeding up the algorithm for solving OTSP.** The algorithm for generating the permutations of vertices in the subset S was implemented as a recursive one. It means that this algorithm selects the vertex for a certain position and then it calls the same algorithm recursively to select the vertex for the next position. It is possible to calculate the length of the part of permutation before a recursive call. If this length is greater than the length of the best solution of the OTSP achieved till this time for all the tested subsets S then all the permutations starting with the same

sequence of vertices are skipped. In addition, the algorithm starts with the subsets S with the minimal number of vertices. This principle significantly reduces the computational time of the second stage of the EM and makes EM the suitable method for small networks with the maximum number of vertices about 25.

4.2 Heuristic Method

The heuristic method was implemented as described in 3.2. The initial step of the algorithm is very important. The described NGH starts with initial step $S = \{m\}$, because we assume $m \in S_{opt}$. However, this hypothesis does not have to be fulfilled in all cases. It is clear from the experimental results obtained on test networks, which are shown in Tab. 1. If $m \notin S_{opt}$, then the result of NGH cannot be an optimal solution. Tab. 1 shows that better results of NGH were achieved in the cases when $m \in S_{opt}$. However, the quality of solution obtained by NGH does not depend only on the initial step, improving the strategy of the starting vertex selection is one possible way how to improve the proposed heuristic method.

4.3 Experimental results

Table 1 summarizes the results obtained by using described methods on test networks. The distance limit λ was set to 4 (less than 1/7 of the diameter of the network). Having these parameters given, the computational time depends on the network topology. For the Exact Method it varies between 50 and 120 minutes. For the heuristic method it is less than 1 second. The optimal solutions obtained by the Exact Method (EM) are emphasized by the bold font.

The results of the described methods Tab. 1

Net No.	Method	Route length	NGH/EM route length ratio	The real distance mean value (limit was set to 4)	$m \in S_{opt}$
1	NGH	41.52	1.00	3.8379	YES
	EM	41.52		3.8379	
2	NGH	52.52	1.21	3.2148	YES
	EM	43.44		3.9476	
3	NGH	74.84	1.70	3.3527	NO
	EM	44.14		3.9500	
4	NGH	48.57	1.13	3.3288	YES
	EM	42.82		3.7987	
5	NGH	53.02	1.32	3.2731	NO
	EM	40.31		3.9646	
6	NGH	55.99	1.34	3.2126	YES
	EM	41.66		3.8744	
7	NGH	36.16	1.09	3.7522	YES
	EM	33.14		3.9798	
8	NGH	51.58	1.23	3.8771	YES
	EM	41.91		3.9822	
9	NGH	45.98	1.02	3.7727	YES
	EM	45.22		3.9604	

Fig. 1 and Fig. 2 depict the resulting routes for the selected test networks. The numbers in brackets are numbers of the passengers randomly generated between 0 and 100. Fig. 1 depicts the resulting routes in test network number 3. I have selected this example, because in this case the NGH gives the worst result.

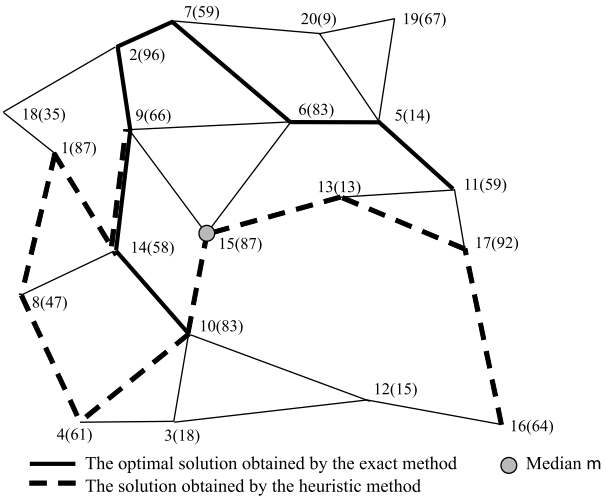


Fig. 1 The resulting routes in the network No. 3

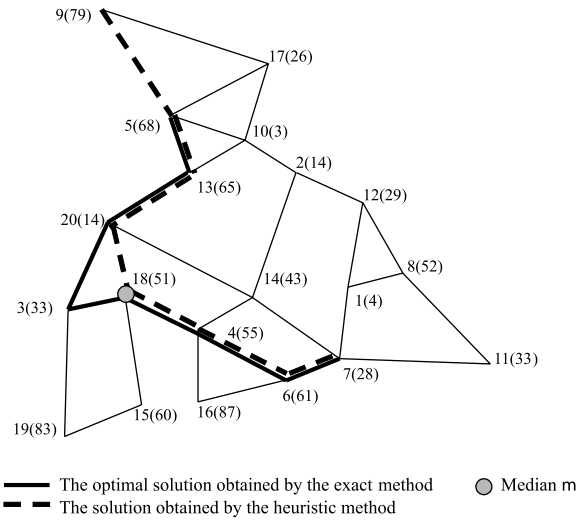


Fig. 2 The resulting routes in the network No. 7

5. Conclusion

Two methods for solving the problem described in 2 were discussed in this paper. The first one was the EM. Computational complexity of this method is very high. It is suitable only for a small network. But the optimal solutions achieved by this method are very important, especially for testing the heuristic method.

The heuristic method called "Neighbor Greedy Heuristics" was proposed too. Experimental computations carried out on the test networks made out that NGH is a very quick method. But it turned out that this method does not give good results in some cases. The main goal for my further work is to find ways for improving the heuristics in order to gain the solutions closer to optimal ones.

References

[1] CIPRIANI, E., FUSCO, G., GORI, S., PETRELLI, M.: *A procedure for the solution of the urban bus network design problem with elastic demand,* E-Proc. of 10th EWGT Meeting and 16th Mini-EURO Conference, Poznan, Poland, 2005.

[2] CERNA, A., CERNY, J., PESKO, S:, CZIMERMAN, P.,: *Network Reduction Problems, Journal of Information,* Control and Management Systems, Vol. 5, No 2/1,2007, pp. 139–145, ISSN 1336-1716

[3] CERNA, A.: *Intensification MHD and her Potential conflict with attendance limit (in Slovak)*, Proc. 6. medzinarodnej konferencie o verejnej osobnej doprave, Bratislava, 2003, pp. 43–47, ISBN 80-233-0485-2

[4] CERNA, A.: *Note to Transit Stops Accessibility Constraint (in Slovak),* Horizonty dopravy 2/2003, VUD Zilina, pp. 23–24

[5] CERNA, A.: *Bus Route Design in a Weak Demand Area,* Presented on Czech-Slovak Seminar on Transport Optimization, Jindrichuv Hradec, 2007, can be requested from cerna@fm.vse.cz.

[6] ERLANDER, S., SCHEELE, S.: *A Mathematical Programming Model for Bus Traffic in a Network.* Transportation and Traffic Theory (ed. Buckley), REED, Sydney, 1974, pp. 581–605.

Jaroslav Janacek – Michal Sibila *

OPTIMAL EVACUATION PLAN DESIGN WITH IP-SOLVER

This paper deals with two different computer-supported approaches to an evacuation plan design, which should assign the available vehicles to endangered dwelling places so that the total time of evacuation is minimal. It is assumed that some safe place is pre-assigned to each endangered dwelling place and there are determined locations of homogenous fleets of available vehicles, which can be used for transport of the endangered population from their dwelling places to the pre-assigned places. In this paper, we suggest and compare two approaches to the evacuation plan design. The first approach assigns all vehicles of a fleet to one evacuated dwelling place and the second one enables to assign individually any part of a fleet to the dwelling places.

1. Introduction

Let us consider an emergency situation when population of a given set of towns and villages is endangered by some threat. The casualties can be avoided to some extent by evacuation of the endangered population to some safe places, which have been predestinated for each evacuated place in advance. To perform the evacuation, some available vehicles are disposable at several places located in the neighborhood of the endangered dwelling places. It is necessary to determine a route for each used vehicle so that the population is evacuated from its original places to the predetermined refuges. The evacuation should be performed so that the time of evacuation is as short as possible. The end of evacuation is given by the time when the last inhabitant reaches his/her predestinated shelter.

If any vehicle is used to provide an evacuated community with this service, the route of vehicle may take a prescribed form. The route starts at the original vehicle location, continues to the served village or town, picks up a portion of the evacuated inhabitants and takes them to the predetermined refuge [4]. If necessary, the vehicle may return back to the evacuated place and save another portion of its population by taking them to the refuge. This cycle can be repeated several times.

Even under this simplifying assumption about a form of route, time optimal assignment of the vehicles to the evacuated places represents a hard combinatorial problem, whose solution must be usually found in a short time of several minutes. In this paper, we present different approaches to this problem. Each of these approaches enables to employ a commercial IP-solver, to obtain a final concrete set of decisions on the vehicle assignment. All these approaches consist of a linear programming model formulation and solving process performed by commercial software with usage of its particular characteristics. The presented approaches differ in models and following quality of obtained solutions. These properties were studied by numerical experiments and their results are presented in the concluding part of this paper.

To formulate the following mathematical models for the individual approaches, we shall use a common denotation, where symbol I denotes the set of all considered homogenous fleets of vehicles. Each homogenous fleet $i \in I$ is characterized by a number N_i of vehicles and by vehicle capacity K_i. We shall assume that the fleet i is located at a node $u(i)$ of a road network covering the serviced area. The endangered dwelling places form a set J and each dwelling place $j \in J$ is described by a number b_j of its population and by a road network node $v(j)$, where the village or town j is located. We assumed for the purpose of evacuation that a destination place $w(j)$ is assigned to each dwelling place $j \in J$. Furthermore, let t_{ij} denote the time, which is necessary for a vehicle from the fleet i to traverse the distance between the nodes $u(i)$ and $v(j)$. In addition, let s_j denote the time necessary for traversing the distance between the nodes $v(j)$ and $w(i)$. Using these denotations, we can introduce two approaches to the evacuation plan design problem. The first approach assumes that each considered homogenous fleet is indivisible, i.e. all the vehicles of one fleet perform simultaneously the same activities like a convoy or vehicle train. The second approach is based on the assumption that each fleet can be split into arbitrary integer parts and only vehicles of one part have to act as a convoy.

The first approach leads to a simpler but larger model, whereas the second one pays for its smaller size by non-linearity of the associated model. Advantages and disadvantages of the both approaches are studied in the next two sections and some results of numerical experiments are presented in the concluding part of this paper to demonstrate efficiency of the both approaches, in

* **Jaroslav Janacek[1], Michal Sibila[2]**
[1] Department of Transportation Networks, Faculty of Management and Informatics, University of Zilina, Slovakia, E-mail: jaroslav.janacek@fri.uniza.sk,
[2] Department of Technical Sciences and Informatics, Faculty of Special Engineering, University of Zilina, Slovakia

the case when a commercial software tool is used for obtaining final decisions on the evacuation plan.

2. Evacuation problem with indivisible fleets

2.1 Problem formulation and model building

Let us consider that a set J represents evacuated dwelling places where each element $j \in J$ is characterized by a location $v(j)$, a number of population b_j and a location $w(j)$, to which all population must be transported. For this purpose a set I of indivisible fleets is disposable where each fleet $i \in I$ is characterized by a location $u(i)$, a number N_i of identical vehicles and a vehicle capacity K_i. Let t_{ij} and s_j be travelling times between the locations $u(i)$ and $v(j)$ and between $v(j)$ and $w(j)$ respectively. The objective is to determine a route of each indivisible fleet so that all population is evacuated and the longest fleet route is minimal [3].

Coming out of the assumption on the possible form of a fleet route, we introduce the variable $z_{ij} \in \{0,1\}$ defined for each pair of the fleet i and dwelling place j. This variable takes the value of 1 if and only if the fleet i is assigned to the dwelling place j.

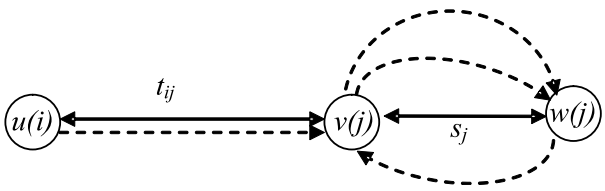


Fig. 1 A route with two visits at v(j)

Taking into account that the fleet i can visit an evacuated place several times, we denote the number of journeys of fleet i from $v(j)$ to $w(j)$ by variable $x_{ij} \in Z^+$. Now, when the fleet i is assigned to the dwelling place j, its travelling time (see Fig. 1) is equal to:

$$t_{ij} + s_j + 2s_j(x_{ij} - 1) \tag{1}$$

The last introduced variable is $T \geq 0$, which denotes an upper bound of all route times.

We can take into account that a maximal sensible time T^{max} of the evacuation can be given. In this case the maximal possible number of visits of fleet i at the place j can be evaluated to comply with this limit. The inequality (2) must hold for the x_{ij}.

$$t_{ij} + s_j + 2s_j(x_{ij} - 1) \leq T^{\max} \tag{2}$$

The inequality (2)can be rewritten as:

$$x_{ij} \leq \left\lfloor \frac{T^{\max} - t_{ij} + s_j}{2s_j} \right\rfloor \tag{3}$$

Denote $P_{ij}(T^{\max}) = \left\lfloor \frac{T^{\max} - t_{ij} + s_j}{2s_j} \right\rfloor$.

If $P_{ij}(T^{\max}) \leq 0$, then the fleet i is not able to service the place j. To minimize the size of a built model, we introduce a set $J(i)$ of all the places from J, for which the inequality $P_{ij}(T^{\max}) > 0$ holds. Similarly, we define a set $I(i)$ of all the fleets from I, for which $P_{ij}(T^{\max}) > 0$ holds.

The linear model of an evacuation plan design can be completed now as follows:

$$\textit{Minimize } T \tag{4}$$

$$\textit{Subject to } (t_{ij} - s_j)z_{ij} + 2s_j x_{ij} \leq T \quad \textit{for } i \in I, j \in J(i) \tag{5}$$

$$\sum_{j \in J(i)} z_{ij} \leq 1 \quad \textit{for } i \in I \tag{6}$$

$$x_{ij} \leq P_{ij}(T^{\max}) z_{ij} \quad \textit{for } i \in I, j \in J(i) \tag{7}$$

$$\sum_{i \in I(j)} N_i K_i x_{ij} \geq b_j \quad \textit{for } j \in J \tag{8}$$

$$z_{ij} \in \{0,1\} \quad \textit{for } i \in I, j \in J(i) \tag{9}$$

$$z_{ij} \in Z^+ \quad \textit{for } i \in I, j \in J(i) \tag{10}$$

$$T \geq 0 \tag{11}$$

The constraints (5) assure that the travelling time of each fleet is less or equal to the upper bound T. The constraints (6) enable for the fleet i to be assigned at most to one evacuated dwelling place. Constraints (7) cause that if the variable z_{ij} is equal to zero, then the variable x_{ij} is also zero, which means that if the fleet i is not assigned to the place j, then the fleet cannot visit this place at all. The constraints (8) ensure that each dwelling place j is provided with a sufficient capacity, which enables to evacuate all the population of the size b_j.

2.2 Numerical experiments

To verify the suggested use of the IP-solver, we formulated ten different instances of the problem. One of the instances denoted as "Hradza" comes out of the possible emergency situation, which can occur if the dam Liptovska Mara breaks. Then, under given assumptions, 26 communities would have to be evacuated to 26 predestined places. For this evacuation, 411 vehicles of different capacities located at three bigger towns of the area are available. The other instances were formulated in nine areas of the Slovak Republic in a similar way. These instances are denoted by names of the biggest towns of the areas. The numbers and capacities of available vehicles were generated similarly to the first instance for the vehicles to be able to satisfy the demand on evacuation. These benchmarks were used to verify the suggested method, which consists in a particular model building and employing the general IP-solver for obtaining of a good solution of the problem. To be able to perform the computation in a given time, we used the general

optimisation software environment XPRESS-IVE for our study [5], [6]. This software system includes the branch-and-cut method and it also enables exploitation of the premature stopping rules. The software is equipped with the programming language *Mosel*, which can be used for both the input of model and writing of input and output procedures. Furthermore, the language has its own tools for the stopping rules adjustment.

The main disadvantage is born-in to the branch-and-bound method, which is the basic solving method of any IP-solver. This is the possibility that the list of determined but unfathomed solution subsets may grow exponentially instead of being reduced to an empty set. When a special IP-solver is designed, all specific properties of the problem can be used to improve quality of the upper and lower bounds, whereas only a general problem relaxation can be employed in the case of this general solver. We used the possibility of the solver, which enables a premature termination of the searching process whenever a fixed time limit is exceeded. The experiments were performed on a personal computer equipped with Intel Core 2 6700 with parameters: 2.66 GHz and 3 GB RAM. The first series of experiments corresponds with the case, when each vehicle is considered as one individual fleet. The best results obtained in the computational time of 20 minutes are presented in Table 1, where "No. of F" denotes the considered number of fleets, "No. of EP" denotes the number of evacuated dwelling places, Rows denotes the number of structural constraints of the model and Columns denotes the number of used variables including the auxiliary ones, which are automatically introduced by the solver. In the row denoted as "Tmax", there are reported the predetermined values of T^{max} in minutes. The symbol "Tbest" denotes the row, where the best-found time of evacuation is plotted.

As the preliminary experiments showed that decreasing the parameter T^{max} to diminish the numbers of constraints and vari-

Results of numerical experiments for indivisible fleets of one vehicle — Table 1

Instances:	Bratislava	Dubnica nad Vahom	Hradza	Kosice	Liptovsky Mikulas	Leopoldov	Michalovce	Nitra	Nove Zamky	Puchov
No. of F.	635	183	411	483	107	281	105	211	107	435
No. of EP.	25	25	26	25	25	25	25	25	25	25
Rows	16958	4966	11485	11751	2914	7331	2860	5722	2914	10853
Columns	31539	9151	21324	22836	5351	13770	5251	10551	5351	20834
Tmax [min]	162	938	202	184	668	140	1420	423	1497	278
Tbest [min]	**81**	**434**	**93**	**92**	**294**	**101**	**586**	**182**	**704**	**139**

Results of numerical experiments for indivisible fleets of two vehicles — Table 2

Instances:	Bratislava	Dubnica nad Vahom	Hradza	Kosice	Liptovsky Mikulas	Leopoldov	Michalovce	Nitra	Nove Zamky	Puchov
No. of F.	323	92	209	243	56	144	55	107	54	219
No. of EP.	25	25	26	25	25	25	25	25	25	25
Rows	8638	2509	5789	5852	1494	3769	1504	2914	1483	5456
Collums	16043	4601	10780	11417	2758	7057	2745	5351	2701	10469
Tmax [min]	162	938	202	184	668	140	1420	423	1497	278
Tbest [min]	**81**	**495**	**93**	**92**	**328**	**108**	**584**	**194**	**750**	**139**

Results of numerical experiments for indivisible fleets of four vehicles — Table 3

Instances:	Bratislava	Dubnica nad Vahom	Hradza	Kosice	Liptovsky Mikulas	Leopoldov	Michalovce	Nitra	Nove Zamky	Puchov
No. of F.	171	46	113	126	35	78	33	57	31	115
No. of EP.	25	25	26	25	25	25	25	25	25	25
Rows	4576	1267	2864	3024	826	2053	898	1564	854	2745
Columns	8485	2301	5551	5898	1538	3823	1633	2851	1543	5366
Tmax [min]	162	938	202	184	668	140	1420	423	1497	278
Tbest [min]	**81**	**510**	**93**	**92**	**381**	**105**	**796**	**198**	**1091**	**139**

Results of numerical experiments for indivisible fleets of eight vehicles Table 4

Instances:	Bratislava	Dubnica nad Vahom	Hradza	Kosice	Liptovsky Mikulas	Leopoldov	Michalovce	Nitra	Nove Zamky	Puchov
No. of F.	98	28	71	71	23	48	24	33	22	65
No. of EP.	25	25	26	25	25	25	25	25	25	25
Rows	2555		1652	1687		1273		888		1376
Columns	4785		3331	3296		2353		1623		2847
Tmax [min]	162	938	202	184	668	140	1420	423	1497	278
Tbest [min]	**81**	*	**93**	**92**	*	**119**	*	**272**	*	**139**

ables of the model had no effect on obtaining a better solution in the given time limit, we decided to explore another way of a model size diminishing. We formed models with a smaller number of fleets so that we grouped the vehicles by two, four and eight if possible to fleets of several vehicles with the same capacity. The derived problems were solved and the results are presented in tables 2, 3 and 4 respectively.

It can be observed that the process of forming the bigger fleets ended at eight-vehicle fleets for some instances, when the solving process fails in finding a solution due to infeasibility. These cases are denoted by asterisk in the row "Tbest". Furthermore, the found resulting evacuation times turned worse for instances, where a feasible solution was found. This result evoked an idea to formulate the evacuation problem for divisible fleets, which is the topic of the next section.

3. Evacuation problem with divisible fleets

3.1 Problem formulation and model building

Similarly to the previous section, we consider that the set J represents evacuated dwelling places, where each element $j \in J$ is characterized by a location $v(j)$, a number of population b_j and a location $w(j)$, to which all population must be transported. For this purpose the set I of divisible fleets is disposable, where each fleet $i \in I$ is characterized by a location $u(i)$, a number N_i of identical vehicles and a capacity K_i of an individual vehicle. Let t_{ij} and s_j be travelling times between the locations $u(i)$ and $v(j)$ and between $v(j)$ and $w(j)$ respectively. The objective is to determine a route of each part of divisible fleet so that all the population be evacuated and the longest route be minimal.

Coming out of the assumption on the possible form of a route, which stays the same as depicted in Fig. 1, we introduce the variables $z_{ij} \in \{0,1\}$ and $x_{ij} \in Z^+$ defined for each pair of fleet i and dwelling place j as before. In addition to these variables, we introduce the variables $q_{ij} \in Z^+$, which denote a number of vehicles of the fleet i assigned to the dwelling place j.

Taking into account expressions (1) and (3) a model of the evacuation plan design with divisible fleets can be written as follows:

$$\textit{Minimize } T \tag{12}$$

$$\textit{Subject to } (t_{ij} - s_j)z_{ij} + 2s_j x_{ij} \le T \ \textit{ for } i \in I, j \in J(i) \tag{13}$$

$$x_{ij} \le P_{ij}(T^{\max})z_{ij} \ \textit{ for } i \in I, j \in J(i) \tag{14}$$

$$q_{ij} \le N_i z_{ij} \ \textit{ for } i \in I, j \in J(i) \tag{15}$$

$$\sum_{j \in J(i)} q_{ij} \le N_i \ \textit{ for } i \in I, \tag{16}$$

$$\sum_{i \in I(j)} K_i q_{ij} x_{ij} \ge b_j \ \textit{ for } j \in J \tag{17}$$

$$z_{ij} \in \{0,1\} \ \textit{ for } i \in I, j \in J(i) \tag{18}$$

$$x_{ij} \in Z^+ \ \textit{ for } i \in I, j \in J(i) \tag{19}$$

$$T \ge 0 \tag{20}$$

$$q_{ij} \in Z^+ \ \textit{ for } i \in I, j \in J(i) \tag{21}$$

The constraints (13) assure that the travelling time of each part of the fleet is less or equal to the upper bound T. The constraints (14) represent binding constraints between the variables x_{ij} and z_{ij}. These constraints cause that if the variable z_{ij} is equal to zero, then the variable x_{ij} is also zero, which means that if the fleet i is not assigned to the place j, then the fleet cannot visit this place at all. The constraints (15) represent binding constraints between the variables q_{ij} and z_{ij}. These constraints cause that if the variable z_{ij} is equal to zero, then the variable q_{ij} is also zero, which means that if the fleet i is not assigned to the place j, then no vehicle of the fleet i can visit this place. The constraints (16) assure that the total number of designed vehicles of the fleet i does not exceed the number N_i.

The constraints (17) ensure that each dwelling place j is provided with a sufficient capacity, which enables to evacuate all the population of size b_j. Unfortunately, these constraints are non-linear and thus this model cannot be input to the IP-solver. To be able to solve this much smaller problem, it must be rearranged to a linear form. An approach to the model linearization is shown in the next section.

3.2 Problem reformulation to a linear model

To rewrite the model into a linear form, we make use of the fact that the variable x_{ij} may take only one of several few values from the range of 0, 1, ..., P_{ij}. We introduce auxiliary variables m_{ij}, which serve as a lower bound of the number of visits at the dwelling place j, which are performed by the vehicles of fleet i. Then the following constrains must hold.

$$q_{ij}x_{ij} \geq m_{ij} \quad for\ i \in I, j \in J(i) \tag{22}$$

Now the series (17) of constraints can be replaced by the constraints (23).

$$\sum_{i \in I(j)} K_i m_{ij} \geq b_j \quad for\ j \in J \tag{23}$$

Now the constraint $q_{ij}x_{ij} \geq m_{ij}$ can be replaced by the following system of logical constraints:

If $x_{ij} = 0$ then $0 \geq m_{ij}$.
If $x_{ij} = 1$ then $q_{ij} \geq m_{ij}$.
If $x_{ij} = 2$ then $2q_{ij} \geq m_{ij}$.
...
If $x_{ij} = k$ then $kq_{ij} \geq m_{ij}$.
...
If $x_{ij} = P_{ij} - 1$ then $(P_{ij} - 1)q_{ij} \geq m_{ij}$.

It holds here that if some of the constraints is fulfilled for $k = p$, then they are fulfilled for each $k > p$ and, vice versa, if a constraint is not fulfilled for $k = p$, then they cannot be fulfilled for any $k < p$.

Now we introduce variables $y_{ij}^p \in \{0, 1\}$ for $i \in I$ and $j \in J(i)$ and $p = 0, 1, ..., P_{ij} - 1$ and the system of logical constraints can be replaced by the series (24) of constraints.

$$N_i P_{ij} y_{ij}^p + pq_{ij} \geq m_{ij}$$

$$for\ i \in I, j \in J(i), p = 0, 1, ..., P_{ij} - 1 \tag{24}$$

If the system (24) holds for given values of variables q_{ij} and k is the minimal value of subscript p for which $y_{ij}^k = 0$, then the system must also hold for such a setting of variables y_{ij}^p, where $y_{ij}^p = 1$ for $p = 0, 1, ..., k - 1$ and $y_{ij}^p = 0$ for $p = k, ..., P_{ij} - 1$, i.e. the setting of variables y_{ij}^p fulfils the equation (25).

$$\sum_{p=0}^{P_{ij}-1} y_{ij}^p = k \quad for\ i \in I, j \in J(i) \tag{25}$$

Then the following linear model describes the evacuation plan design problem with divisible fleets:

$$Minimize\ T \tag{26}$$

$$Subject\ to\ (t_{ij} - s_j)z_{ij} + 2s_j \sum_{p=0}^{P_{ij}(T^{\max})-1} y_{ij}^p \leq T$$

$$for\ i \in I, j \in J(i) \tag{27}$$

$$\sum_{p=0}^{P_{ij}(T^{\max})-1} y_{ij}^p \leq P_{ij}(T^{\max}) z_{ij} \quad for\ i \in I, j \in J(i) \tag{28}$$

$$q_{ij} \leq N_i z_{ij} \quad for\ i \in I, j \in J(i) \tag{29}$$

$$\sum_{j \in J(i)} q_{ij} \leq N_i \quad for\ i \in I \tag{30}$$

$$\sum_{i \in I(j)} K_i m_{ij} \geq b_j \quad for\ j \in J \tag{31}$$

$$N_i P_{ij}(T^{\max}) \geq m_{ij} \quad for\ i \in I, j \in J(i) \tag{32}$$

$$N_i P_{ij} y_{ij}^p + pq_{ij} \geq m_{ij}$$

$$for\ i \in I, j \in J(i), p = 0, ..., P_{ij}(T^{max}) - 1 \tag{33}$$

$$z_{ij} \in \{0, 1\} \quad for\ i \in I, j \in J(i) \tag{34}$$

$$T \geq 0 \tag{35}$$

$$m_{ij} \geq 0 \quad for\ i \in I, j \in J(i) \tag{36}$$

$$y_{ij}^p \in \{0, 1\}$$

$$for\ i \in I, j \in J(i), p = 0, ..., P_{ij}(T^{max}) - 1 \tag{37}$$

The constraints (27) have the same meaning as the constraints (13) in the non-linear model (12) - (21), i.e. they assure that the travelling time of each part of the fleet is less than the upper bound T. These constraints were derived from the constraints (13) by substitution of the left-hand-side of the equality (25) for x_{ij}. The constraints (28) are the binding constraints, which assure relations between the variables z_{ij} and the sum of y_{ij}^p, which corresponds with the number x_{ij} of visits of the part of the fleet i at the community j. The constraints (29) were equivalent to the former constraints (15), which assure that if no part of the fleet i is designated to the place j ($z_{ij} = 0$), then the number q_{ij} of vehicles is equal to zero. The constraints (30) assure that the total number of designed vehicles of the fleet i does not exceed the number N_i.

As m_{ij} represents a lower bound of the product $x_{ij}q_{ij}$, which is the number of visits of vehicles from the fleet i at the place j, then the constraints (31) ensure that population of the place j can be evacuated. The constraints (32) ensure that the estimation m_{ij} does not exceed the upper bound $N_i P_{ij}(T^{max})$ of visits and the constraints (33) ensure that the sum of y_{ij}^p over p corresponds with the number of necessary visits of a group of q_{ij} vehicles of the fleet i at j.

3.3. Numerical experiments

To find characteristics of the second approach involving the linear model of evacuation by divisible fleets, we solved the same instances, which are described in section 2.3. We used the same setting of T^{max} and we focused on the evacuation times, which were obtained in the computational time of 20 minutes. The results associated with these experiments are reported in Table 5.

Results of numerical experiments for divisible fleets obtained in 20 minutes of computational time Table 5

Instances:	Bratislava	Dubnica nad Vahom	Hradza	Kosice	Liptovsky Mikulas	Leopoldov	Michalovce	Nitra	Nove Zamky	Puchov
No. of F.	14	10	16	13	9	11	10	10	9	12
No. of EP.	25	25	26	25	25	25	25	25	25	25
Rows	2083	3449	5000	1508	3555	1135	4185	2357	4022	2323
Columns	1457	1517	2564	1243	1412	965	1581	1339	1435	1543
Tmax [min]	162	938	202	184	668	140	1420	423	1497	278
Tbest [min]	**81**	**368**	**81**	**92**	**262**	**93**	**486**	**168**	**637**	**139**

Results of numerical experiments for divisible fleets obtained in 40 minutes of computational time Table 6

Instances:	Bratislava	Dubnica nad Vahom	Hradza	Kosice	Liptovsky Mikulas	Leopoldov	Michalovce	Nitra	Nove Zamky	Puchov
No. of F.	14	10	16	13	9	11	10	10	9	12
No. of EP.	25	25	26	25	25	25	25	25	25	25
Rows	2083	3449	5000	1508	3555	1135	4185	2357	4022	2323
Columns	1457	1517	2564	1243	1412	965	1581	1339	1435	1543
Tmax [min]	162	938	202	184	668	140	1420	423	1497	278
Tbest [min]	**81**	**368**	**81**	**92**	**245**	**93**	**486**	**168**	**613**	**139**

To analyze the sensitivity of the best-found solution on the permitted computational time, we performed these numerical experiments once more for the time limit of 40 minutes. The results are shown in Table 6 and it can be easily found that they were improved only in two instances (Liptovsky Mikulas and Nove Zamky).

4. Comparison and conclusions

We suggested two approaches to the evacuation plan design problem. The first approach was based on the concept of indivisible fleets, which enables a direct formulation of the linear model and easy use of the IP-solver. Nevertheless, the linear model was too large and that is why we suggested a process of its shrinking by clustering evacuation vehicles into bigger fleets.

This process reduced the size of the associated model, but it brought an additional condition in the model and it caused that the resulting evacuation times turned worse. The overview in Table 7 shows that the best results were achieved for the biggest models. This result evoked an idea to formulate the evacuation problem for divisible fleets. This task was successfully solved by a complicated reformulation of the originally non-linear model to the linear one. Even if the new model is much more complicated than the first approach model, it has a much smaller size and enables to reach better results in the same computational time. This fact is demonstrated in Table 8, where the row "Divisible" reports on the best evacuation times obtained by the first approach and the row "Indivisible" reports on the times obtained by the second approach.

As we have the possibility to estimate the optimal solution using an iterative approach, we are able to compare the obtained results of both suggested approaches to the best-known results, which are reported in the row "Best Known" of Table 8. The row "Gap" of this table, where differences between the divisible fleet and the iterative approaches in percentage are given, shows that the suggested approaches constitute a promising way to the evacuation plan design.

An overview of the evacuation times achieved by the approach based on the indivisible fleet concept Table 7

Instances:	Bratislava	Dubnica nad Vahom	Hradza	Kosice	Liptovsky Mikulas	Leopoldov	Michalovce	Nitra	Nove Zamky	Puchov
1 vehicle	81	434	93	92	294	101	586	182	704	139
2 vehicles	81	495	93	92	328	108	584	194	750	139
4 vehicles	81	510	93	92	381	105	796	198	1091	139
8 vehicles	81	*	93	92	*	119	*	272	*	139
Minimum	**81**	**434**	**93**	**92**	**294**	**101**	**586**	**182**	**704**	**139**

Table 7

Instances:	Bratislava	Dubnica nad Vahom	Hradza	Kosice	Liptovsky Mikulas	Leopoldov	Michalovce	Nitra	Nove Zamky	Puchov
Indivisible	81	434	93	92	294	101	586	182	704	139
Divisible	**81**	**368**	**81**	**92**	**262**	**93**	**486**	**168**	**637**	**139**
Best Known	81	360	81	92	236	93	469	167	568	139
Gap [%]	**0**	**2.2**	**0**	**0**	**11**	**0**	**3.6**	**0.6**	**12.1**	**0**

References

[1] JANACEK, J.: *Service System Design in the Public and Private Sectors,* Proc. of the international conference Quantitative Methods in Economics (Multiple Criteria Decision Making XII), 2004, Virt, ISBN 80-8078-012-9, pp. 101–108.

[2] JANACEK, J.: *Time Accessibility and Public Service Systems,* Proc. of conference Quantitative Methods in Economics, Bratislava, 2006, pp. 57–63.

[3] SIBILA, M.: *Management and Transport of Inhabitant Evacuation (in Slovak)*, Dissertation thesis, Zilinská univerzita, Fakulta specialneho inzinierstva, Katedra technickych vied a informatiky, Zilina, 2009, p. 136.

[4] TEICHMANN, D.: *Contribution to the Problems of Inhabitant Evacuation and Usage of Mathematical Programming for the Evacuation Planning (in Czech)*, Proc. Krizovy management, Vol. 7, 2/2008.

[5] XPRESS-MP Manual "Getting Started". Dash Associates, Blisworth, UK, 2005, p. 105.

[6] XPRESS-Mosel "User guide". Dash Associates, Blisworth, 2005, UK, p. 99.

Acknowledgment: This work was supported by project "Centre of excellence for systems and services of intelligent transport" ITMS 26220120028

Project Part-Financed by the European Union

European Regional Development Fund

Jan Pelikan – Jan Fabry *

PICKUP AND DELIVERY PROBLEM

Vehicle routing problem and traveling salesman problem are classical problems in operational research; this modification of those problems consists of a transport among nodes of the communication network using cyclical routes of vehicles with a given capacity. A transportation demand is given by the place of pickup, the place of delivery and quantity of goods. The goal is to find cyclical routes of a minimal length which ensure the transport requirements. In the paper there are two models proposed for the problem, both are demonstrated on an example. The problem is based on a case study from practice.

Keywords: Pickup and delivery problem, integer programming, heuristic methods

1. Introduction

Traveling salesman problem and vehicle routing problem and their modifications are frequently solved practical examples. In such problems the goal is to assure pick-up or/and delivery of goods in a distribution network. While in those applications it is necessary to find cyclical routes starting and ending in a depot, in the studied problem it is required to transport goods between nodes of the network. Each requirement is specified by the pickup point, the delivery point and the amount of goods which has to be transported. In [1] and [2] the problem is called pickup and delivery problem.

In the practice time windows are often given for each node in the network and the order of pickup and delivery processes has to be defined, because the goods being loaded as the last ones will be unloaded as the first ones. The origin and the destination nodes need not be identical. Vehicles with different capacities can assure the transport of goods. The objective is to minimize the total transportation cost. The column generation method is used for selecting routes in the mathematical model.

In the paper, no time windows and no conditions for order of pickup and delivery are considered. All the vehicles have the same capacity. Each route has to be cyclical for all the included nodes. Two models are proposed in the paper; the first one is based on the optimal flow theory, the second one uses a model of set covering problem.

Let a distribution network be given by $G = \{V, E\}$, where V is a set of n nodes and E is a set of undirected arcs. Each arc (i, j) is evaluated by distance c_{ij}. Let us denote q_{kl} the amount of goods that has to be transported from the node k to node l. Vehicles with the capacity V are used for pickup and delivery and they can start in any node. All the routes have to be cyclical, each vehicle has to come back to the node it starts from. The objective is to minimize the length of all the routes.

Example

Let us consider the distribution network with 4 nodes; the distance matrix C and the transportation requirements matrix Q are given:

$$C = \begin{bmatrix} 0,30,40,60 \\ 30,0,60,50 \\ 40,60,0,20 \\ 60,50,20,0 \end{bmatrix} \quad Q = \begin{bmatrix} 0,0,7,5 \\ 0,0,5,7 \\ 5,6,0,0 \\ 8,0,8,0 \end{bmatrix}.$$

Vehicle capacity is $V = 12$.

2. Optimal multi-product flow model

Let us define two additional nodes in the network: the node 0 as the source and the node $n+1$ as the sink. The following variables are defined in the model:

$y_{ij} \geq 0$, integer – number of vehicles going through the arc (i, j) in the direction from i to j $(i, j = 0, 1, ..., n+1, i \neq j)$,

$x_{ij}^{kl} \geq 0$, amount of goods (part of the total amount q_{kl}) transported from the node i to the node j $(i, j = 0, 1, ..., n+1, i \neq j; k, l = 1, 2, ..., n, k \neq l)$.

The mathematical model follows:

$$z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} y_{ij} \longrightarrow \min \tag{1}$$

$$\sum_{i=1}^{n} y_{ij} = \sum_{i=1}^{n} y_{ji} \quad j = 1, 2, ..., n, \tag{2}$$

* **Jan Pelikan, Jan Fabry**
University of Economics Prague, Czech Republic, E-mail: Pelikan@vse.cz

$$\sum_{i=0}^{n} x_{ij}^{kl} = \sum_{i=1}^{n+1} x_{ji}^{kl} \quad j = 1, 2, ..., n, k, l = 1, 2, ..., n, k \neq l \qquad (3)$$

$$\sum_{k,l} x_{ij}^{kl} \leq V y_{ij}, \quad i, j = 1, 2, ..., n, i \neq j \qquad (4)$$

$$x_{0,k}^{kl} = q_{kl}, k, l = 1, 2, ..., n, k \neq l;$$

$$x_{0,j}^{kl} = 0, k, l = 1, 2, ..., n, k \neq l; j = 1, 2, ..., n+1, j \neq k,$$

$$x_{l,n+1}^{kl} = q_{kl}, k, l = 1, 2, ..., n, k \neq l; \qquad (5)$$

$$x_{j,n+1}^{kl} = 0, k, l = 1, 2, ..., n, k \neq l; j = 0, 1, ..., n+1, j \neq k,$$

$$x_{ij}^{kl} \geq 0, i, j = 0, 1, ..., n+1, i \neq j; k, l = 1, 2, ..., n, k \neq l,$$

$$y_{kl} \geq 0, \text{ integers}, k, l = 1, 2, ..., n, k \neq l \qquad (6)$$

The objective (1) corresponds to the sum of the evaluations of all the arcs in the solution, i.e. the total length of all the routes. Equations (2) assure the vehicle will leave the location that it will visit. With respect to equations (3), amount of goods being transported from k to l entering the node j leaves this node. Inequalities (4) disable exceeding the capacity of the vehicle transporting goods between the nodes i and j. Equations (5) assure that both the total flow from the source 0 to the node k and the total flow from the node l to the sink $n + 1$ are equal to the total requirement q_{kl}. All other flows from the source and to the sink are set to 0.

Example 1:

The application of the model (1) - (6) to the example introduced above leads to the objective value equal to 260 and to the following values of variables:

$y_{12} = y_{13} = y_{24} = y_{34} = 1, y_{31} = y_{43} = 2$ (see Table 1).

Hence, two routes are generated:
route A: 2 → 4 → 3 → 1 → 2 of the length 140
and route B: 1 → 3 → 4 → 1 of the length 120.

Optimal solution of example 1 Tab. 1

Route A			Route B		
Node	Pickup	Delivery	Node	Pickup	Delivery
2	$q_{24} = 7$ $q_{23} = 5$		1	$q_{14} = 5$ $q_{13} = 7$	
4		$q_{24} = 7$	3		$q_{13} = 7$
3	$q_{31} = 5$ $q_{32} = 6$	$q_{23} = 5$	4	$q_{41} = 8$	$q_{14} = 5$
1		$q_{31} = 5$	1		$q_{41} = 8$
2		$q_{32} = 6$			

Note: The values of matrix Y provide information about the arcs that are included in the optimal routes. Generation of the routes based on this information need not be unique. In addition it is necessary to determine the depot for each route.

3. Routes generation model

The model is based on the assumption there are proposed routes satisfying all the conditions of transportation, depots in all the proposed routes are determined. The goal is to select routes satisfying requirements given by the matrix Q and minimizing the total length of all the routes derived from the matrix C. A number of vehicles realizing the transport will be determined as well.

Let us assume S routes including arcs of a distribution network, the length of each route is denoted ds (s = 1,2,...,S). Parameter $a_{ij}^{kl}(s)$ equals 1 if goods transported on the route s from the node k to node l use the arc (i, j), 0 otherwise.

Example 2:

Parameters $a_{ij}^{kl}(s)$ for the route A: 2 → 4 → 3 → 1 → 2 presented in the previous chapter are defined in Table 2.

Parameters $a_{ij}^{kl}(s)$ for route A Tab. 2

(i,j) / (k,l)	2.4	2.3	4.3	4.1	3.1	3.2
(2,4)	1	1				
(4,3)		1	1	1		
(3,1)				1	1	1
(1,2)						1

In the model, the following variables are used:

$y_s \geq 0$, integer – number of vehicles on the route s (s = 1, 2, ..., S),

$x_{kl}^s \geq 0$, amount of goods (part of the amount q_{kl}) transported on the route s (k, l = 1, 2, ..., n, $k \neq l$, s = 1, 2, ..., S).

The mathematical model is:

$$z = \sum_{s=1}^{S} y_s d_s \longrightarrow \min \qquad (7)$$

$$\sum_{s=1}^{S} x_{kl}^s = q_{kl}, k, l = 1, 2, ..., n, k \neq l, \qquad (8)$$

$$\sum_{k,l} a_{ij}^{kl} x_{kl}^s \leq V y_s, i, j = 1, 2, ..., n, i \neq j, s = 1, 2, ..., S, \qquad (9)$$

$$x_{kl}^s \geq 0, k, l = 1, 2, ..., n, k \neq l, y_s \geq 0, \text{ integer},$$

$$s = 1, 2, ..., S. \qquad (10)$$

The objective function (7) corresponds to the total length of all the routes. Equations (8) assure transport of required amount of goods q_{kl} from the node k to node l. Inequalities (9) disable exceeding the capacity of the vehicle transporting goods on the

route s between the nodes i and j. The key issue in this mathematical model is generation of the routes and their number. The same routes with the different depots are considered as different routes in this model.

4. Case study and conclusions

The logistic company operating in the Czech Republic solves the problem of routes design. Eight big cities (hubs or transit places) are given, distances of the edges, estimation of future transport requirements and capacities of vehicles are known. Multi-product flow model was used for optimization of routes design. The model contains 30 integer variables and 2100 continuous variables. The computation took 1,5 hour with using CPLEX 10.

Values of y_{ij} give the numbers of vehicles going on the edge (i,j). They were used as the proposal for the optimal design of the routes and determination of depots. Results will bring remarkable decrease of the transportation cost.

In the article, new type of pickup and delivery problem is presented. The first model is based on multi-product flow formulation and enables transit between routes. In the second model, the generated routes are chosen without the possibility of transits.

Acknowledgment

This research was supported by GACR grant No. GACR 402/09/0041 and GACR402/06/0123.

References

[1] SAVELSBERGH, M., P., M., SOL., M.: *The General Pickup and Delivery Problem,* Transportation Res. 29, 1995, pp.17–29.

[2] HANG, X., ZHI-LONG, CH., RAJAGOPAL, S., ARUNAPURAM, S.: *Solving a Practical Pickup and Delivery Problem,* Transportation science, vol. 37, 3/2003, pp. 347–364.

[3] PELIKAN, J.: *Discrete Models in Operating Research (in Czech),* Professional Publishing.2001, ISBN 80-86419-17-7.

Stefan Pesko *

MINIMAL TOTAL AREA CONVEX SET PARTITIONING PROBLEM

The linear binary model for finding set partitioning of the points in the plane is studied.

We present heuristic which generates partitioning of points to clusters for a given number of seeds with searched seeds of clusters in a plane. For this cluster the convex hulls are constructed via Graham's scan algorithm. This approach is demonstrated on the real instance of Florida area with 235 points for 10 and 13 clusters.

Keywords: Convexity constraints, set partitioning problem, location problem, convex hull

1. Introduction

The set-partitioning problem is a well known NP-hard combinatorial problem. A real-world problem of this class for the vehicle routing problem in Florida area was formulated by Juricek [5]. He requires partitioning of customers to clusters. The clusters should be constructed as "optimal" areas for 1- vehicle routing. An intuitive criterion for the well-formed area of the cluster for one vehicle is the smallest convex polygon that encloses all points of cluster – convex hull.

We present two problems:

- *Convex set partitioning problem:* A set of points in a plane have to be partitioned to a given number of clusters so that every two convex hulls of cluster members have empty intersection and the total area of cluster's convex hulls is minimal.
- *Convex location problem:* A set of q seeds of clusters are searched with a bounded capacity of cluster and convex constraint for a pair of assigned points.

2. Convex hull problem

Finding the convex hull of a set of points in plane is one of most interesting elementary problems in computational geometry [1].

Let $\boldsymbol{P} = \{p_1, p_2, ..., p_n\}$, $n \geq 3$ be a set of distinct points in the Euclidean plane. The *convex hull* of $\boldsymbol{P}$ is the minimal convex area (bordered by points in P) that contains every point of $\boldsymbol{P}$. We will denote this area by *convh* ($\boldsymbol{P}$). For convenience we say that a point $p \in \boldsymbol{P}$ is a boundary point of the convex hull of $\boldsymbol{P}$ if p is a boundary vertex of the convex hull of $\boldsymbol{P}$. The *convex hull problem* is the problem of computing the convex hull of $\boldsymbol{P}$ and reporting the boundary points of the convex hull in the order in which they appear on the hull – we will note this set $\underline{convh}$ ($\boldsymbol{P}$).

Fig. 1 shows an example of the convex hull of set $\boldsymbol{P} = \{p_1, p_2, ..., p_{10}\}$. The points $p_4, p_6, p_7, p_8, p_9 \in (\boldsymbol{P} - \underline{convh}(\boldsymbol{P}))$ lie in the convex hull but they are not boundary points and the points $p_1, p_2, p_3, p_5, p_{10} \in \underline{convh}(\boldsymbol{P})$ are boundary points of the convex hull.

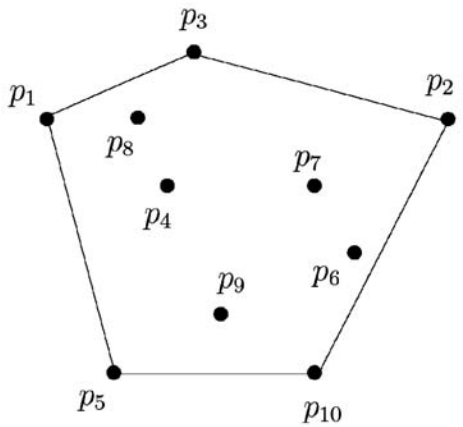


Fig. 1 The points on the convex hull $\underline{convh}$ (P) = [p_1, p_3, p_2, p_{10}, p_5]

As early as 1972, Graham gave a convex hull algorithm with $O(nlog(n))$ worst-case running time. In our application we used the Python program [3] implementing Graham's scan algorithm.

3. Convex set partitioning problem

The problem of partitioning customers (represented by the points on the plane) to the convex hulls of vehicle areas can be formulated as the following optimization problem (CSP):

Let us have the set of n points $\boldsymbol{P} = \{p_1, p_2, ..., p_n\}$ *in the plane and the integer q where* $n \geq 6$ *and* $2 \leq q \leq \lfloor n/3 \rfloor$. *The goal is to*

* **Stefan Pesko**
Department of Mathematical Methods, Faculty of Management Science and Informatics, University of Zilina, Slovakia,
E-mail: stefan.pesko@fri.uniza.sk

find a set partitioning of the set $\boldsymbol{P}$ *to q partitions - clusters of points -* $\boldsymbol{P}(q) = (P_1, P_2, ..., P_q)$ *such that every two clusters have empty intersection and the total area of the convex hulls of partitioning members is minimal.*

Fig. 2 shows an example of the convex set partitioning of set $\boldsymbol{P}$ to three clusters P_1, P_2, P_3 of points. Every two convex hulls

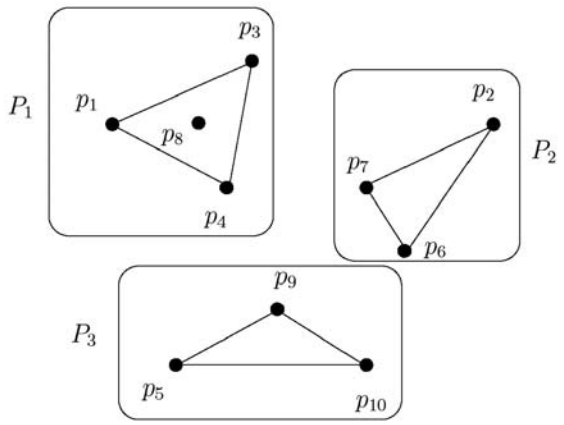


Fig. 2 Convex set partitioning P(3) = (P1, P2, P3) of points $\boldsymbol{P}$

of partitioning have empty intersection i.e. $\overline{convh}(P_i) \cap \overline{convh}(P_j) = \varnothing$ for $1 < i < j < 3$ and the total area of the convex hulls $|\overline{convh}(P_1)| + |\overline{convh}(P_2)| + |\overline{convh}(P_3)|$ is minimal, where $|p|$ denotes the area o the polygon p

Let $\mathcal{H} = \{H_1, H_2, ..., H_m\}$ be the set of all subsets of the boundary points from $\boldsymbol{P}$ which lie on the convex hulls. We will use the abbreviate notations $M = \{1, 2, ..., m\}$ and $N = \{1, 2, ..., n\}$. We can now define the binary variables

$$x_j = \begin{cases} 1 & \text{if solution has clusters with convex hull of points } H_j \in \mathcal{H} \\ 0 & \text{if otherwise.} \end{cases}$$

We define the point-to-cluster incidence matrix $A = (a_{ij})$ where

$$a_{ij} = \begin{cases} 1 & \text{if } p_i \in \overline{convh}(H_j), \\ 0 & \text{if otherwise} \end{cases}$$

and the cluster-to-cluster incidence matrix $B = (b_{ij})$ where

$$b_{ij} = \begin{cases} 1 & \text{if } \overline{convh}(H_i) \cap \overline{convh}(H_j) \neq \emptyset, \\ 0 & \text{otherwise} \end{cases}$$

Let us remind that $|A|$ denotes the area of object A. Now we can formulate and solve the CSP problem as the following binary programming problem (BCSP):

$$\sum_{j \in M} \left|\overline{convh}(H_j)\right| \cdot x_j \to \min, \tag{1}$$

$$\sum_{j \in M} x_j = q \tag{2}$$

$$\sum_{j \in M} a_{ij} x_j = 1 \qquad i \in N \tag{3}$$

$$\sum_{j \in M - \{i\}} b_{ij} x_j = 0 \qquad i \in M \tag{4}$$

$$x_j \in \{0, 1\} \qquad j \in M \tag{5}$$

The objective function (1) minimizes the total area of the convex hulls in the solution. The cardinality constraint (2) states that the partitioning contains exactly q clusters. The set partitioning constraints (3) ensure that every point of the set $\boldsymbol{P}$ lies in exactly one cluster. Constraint (4) tells us that every two convex hulls are disjunct.. The logical constrain (5) chooses a set of clusters in the solution $x_{jk} = 1$, $k = 1, 2, ..., q$. The optimal solution of the CSP problem has the following form

$$\mathcal{P}(q) = (P_1, P_2, \ldots, P_q),$$

$$P_k = \left\{p \in \mathcal{P} : p \in \overline{convh}(H_{jk}), x_{jk} = 1\right\}.$$

For the exact solution of the BCSP we have the known set $\boldsymbol{H}$ which has an exponential growth. Its size is too large for the solving of practical problems via accessible bivalent programming solvers.

One approach how to try to find an approximate solution is the reduction of the set $\boldsymbol{H}$. The natural way how to do it is based on the capacities c_i of the points p_i. Let C be the capacity of vehicles. We can define the capacity of a cluster as the sum of capacities of its points. The cluster of the points $\boldsymbol{P}$ is then a *feasible cluster* if

$$\sum_{p_i \in P} c_i \leq C. \tag{6}$$

Now we can formulate the problem which helps to reduce the set $\boldsymbol{H}$.

4. Convex location problem

When we approximate q convex hulls of the CSP problem as q-median of seeds of clusters we have the following optimization problem (CL):

Let us have the given set of n *points* $\boldsymbol{P} = \{p_1, p_2, ..., p_n\}$ *in the plane, integer q where* $n \geq 6$ *and* $2 \leq q \leq [n/3]$ *and positive integer numbers* $c_1, c_2, ..., c_n$, C. *The goal is to find a q seeds of feasible clusters of points -* $\boldsymbol{S}(q) = \{s_1, s_2, ..., s_q\}$ *such every two cluster hulls of cluster have empty intersection and the sum of Euclidean distances of points to seeds of clusters is minimal.*

This model measures the cluster by a sum of distances from the seed of cluster to the points of cluster as we can see in Fig. 3. The *convexity constraint* assigns the point p to the cluster P_1 because the distance from the point s_1 to point p is shorter than the distance from the point s_2 to point p.

Let $\boldsymbol{D} = (d_{ij})$ be the matrix of Euclidean distances between the points $\boldsymbol{P}$ and let us denote the maximum distance by d_{max}. We can define the bivalent variables $y_i = 1$ if the point p_i is a seed of

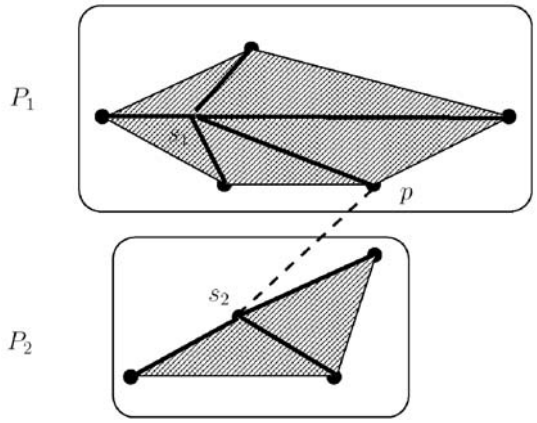


Fig. 3 Convexity constraint for clusters P_1 and P_2 with seeds s_1, s_2

the same cluster and $y_i = 0$ otherwise. We also define the bivalent variables $x_{ij} = 1$ if the point pj is assigned to the seed pi. It is easy to construct the set $\boldsymbol{S} = \{p_i \in \boldsymbol{P} : y_i = 1\}$.

Now we can formulate and solve the CL problem as the bivalent-programming problem (BCL):

$$\sum_{i\in N}\sum_{j\in N} d_{ij}x_{ij} \to \min, \tag{7}$$

$$\sum_{i\in N} y_i = q \tag{8}$$

$$\sum_{i\in N} x_{ij} = 1 \qquad j \in N, \tag{9}$$

$$\sum_{j\in N} c_j x_{ij} \le C y_i \qquad i \in N, \tag{10}$$

$$d_{ij}x_{ij} = (d_{kj} - d_{\max})y_k \le d_{\max} \qquad i, j, k \in N,\ i \ne j,\ j \ne k \tag{11}$$

$$x_{ij},\ y_i \in \{0, 1\} \qquad i, j \in N. \tag{12}$$

The objective function (7) minimizes the total distance of the points to the nearest seed. The cardinality constraint (8) states that the solution has q clusters. The assignment constraint (9) ensures that every point of the set $\boldsymbol{P}$ is assigned exactly to one computed seed. The capacity constraint (10) is valid for feasible clusters only. The convexity constraint (11) ensures that the constructed convex hulls are really of shape of a convex polygon. Finding such polygon is similar to the same phase in Voronoi algorithm. The constaraint describes a situation when we need to decide to which cluster a point belongs. When there is more possibilities it is always assigned to a cluster with nearest seedThe last logical constraint (12) chooses seeds and defines clusters with its seed.

It can be proved that the BCL problem is NP-hard. Janacek and Gabrisova studied the possible approach to an approximate solution of the capacitated location problem based on lagrangean relaxation in [4]. Computational experiments with program *glpsol* for solution of mixed linear programming problems show that using Open Source library GLPK (GNU Linear Programming Kit) [7] we can find a good approximate solution of a real-world problem.

5. Instances of Florida area

We verified the models on the two instances of Florida area with $n = 234$ points and the number of clusters $q = 10$ and $q = 13$. The capacities of points are daily weights of customer's demands. For every q we solved 7 instances of BCSP problems with

$$C \in \{C_k : k \in \{0, \ldots, 6\}, C_0 > C_1 > \ldots > C_6\}$$

$$where\ C_0 = \left\lfloor \frac{\sum_{i=1}^{n} c_i}{q+2} \right\rfloor$$

and $C_k + 1$ is the maximum of cluster capacities in the solution of instance with the vehicle capacity C_{k-1} for $k = 1, ..., 6$. The points on convex hulls define the reduced set $\boldsymbol{H}_q = \{H_1, H_2, ..., H_{7\text{-}q}\}$. By solving the BCSP problem, we got the clusters for ten vehicles as we can see in Fig. 4. The convex hulls were constructed by Graham's scan algorithm [3] implemented in *Python* [2], [6], [8]. An interesting property of the obtained solution is that the clusters have a relatively uniform distribution of the cluster capacity defined by the left side of inequality (6). Note that the total time of computing via [7] program *glpsol* was between 15 – 16 hours for one q.

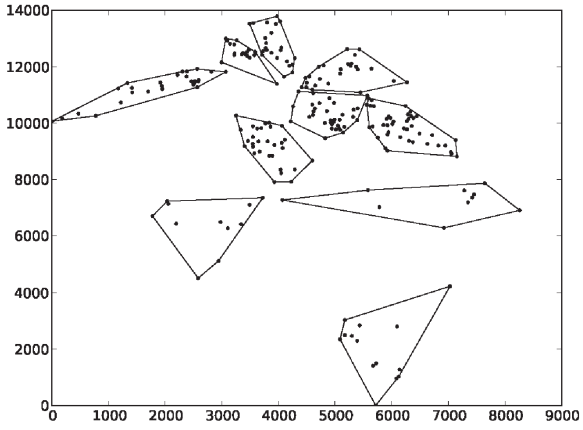


Fig. 4 Convex hulls for 10 clusters of Florida area

6. Open questions

The experiments with Florida instances show that real world BCSP problems with several hundreds of points can require long computational times. Reduction of the problem by solving some associated BCL problems is possible. Open questions are

- the approximate ratio for presented heuristic,
- the applicability of estimation of vehicle service with the aid of the area and weight of cluster,
- the possibility of weakening the formulation of the convexity constraint (11) in the BCL problem,

- the development of special methods for solving the BCSP and BCL problems.
- possibility of formulation and solution of BCSP via Voronoi diagrams

Acknowledgement: The research of the author is supported by the Slovak Scientific Grant Agency under grant VEGA 1/0135/08.

References

[1] BERG, M.: *Computational Geometry: Algorithms and Applications,* 3rd rev. ed., Springer-Verlag, 2008, ISBN 978-3-540-77973-5, 386 pp.
[2] DALE, D., DROETTBOOM, M., FIRING, E., HUNTER, J.: *Matplotlib*, 2008, 754 pp.
[3] GHERMAN, D.: *Recipe 66527: Finding the convex hull of a set of 2D points,* http://code.activestate.com/recipes/66527/
[4] JANACEK, J., GABRISOVA, L.: *Lagrangean Relaxation Based Approximate Approach to the Capacitated Location Problem.* In: Scientific Letters of the University of Zilina, Communications 3, 2006, ISSN 1335-4205, pp. 19–24
[5] JURICEK, M.: *Private communication,* Florida, USA
[6] KAUKIC, M.: *Basics of Programming in Pylab (in Slovak),* FEI TU Kosice, 2006, ISBN 80-8073-634-0, pp. 59
[7] MAKHORIN, A.: *GNU Linear Programming Kit,* Reference Manual, Version 4.29, Moscow, 2008, pp.154
[8] VAN ROSSUM, G., DRAKE, F.: *Python Tutorial,* Python Software Foundation, 2009, pp. 124

Peter Tarabek *

A CONTOUR APPROACH TO THINNING ALGORITHMS

Thinning algorithms are widely used in many image processing tasks. Many thinning algorithms were proposed but they usually tend to process all image pixels in every iteration. Two approaches to contour thinning are described and a short discussion about their features is given. These approaches can be implemented as sequential or parallel algorithms with different deletion rules. Results of comparison and analysis are presented in this paper.

Keywords: thinning, skeleton, skeletonization, vectorization

1. Introduction

Many image processing and pattern recognition problems use thinning as one of the processing step. These problems include vectorization of raster maps and engineering drawings [1], character recognition and analysis [2, 3] and more [4–7]. Thinning belongs to skeletonization techniques [12] which are used to create a skeleton sometimes called medial axis. Like other segmentation techniques, thinning has its advantages and disadvantages. One of the often mentioned disadvantages is a low computational speed. Although thinning algorithms are relatively fast compared to other skeletonization techniques, they are still slow for some tasks. Many thinning algorithms were proposed [8, 9, 10, 13] but they usually tend to process all image pixels in every iteration.

In this paper the thinning technique is briefly described in section 2. In sections 3 and 4, two approaches to contour thinning with different rules are presented. Section 5 describes experiments, section 6 shows results and section 7 presents conclusions.

2. Thinning

Thinning algorithms remove outer pixels layer by layer in iterative process to produce one pixel thick skeleton. The thinning shows good results for objects, the length of which is much larger than their thickness. The skeleton is ideal for this type of objects because it is represented by a set of lines which are a natural representation of objects such as roads and characters. The result of thinning algorithms is a modified binary bitmap which must be further processed to yield a vector representation. Thinning should fulfill these requirements:

- Skeleton should be one pixel thick
- Connectivity should be preserved
- Shape and position of the junction points should be preserved
- Skeleton should lie in the middle of a shape (medial axis)
- Skeleton should be immune to noise (especially to boundary noise)
- Excessive erosion should be prevented (length of lines and curves should be preserved)

The nature of thinning algorithms can be parallel or sequential. Parallel thinning algorithms make their decisions about deleting pixels based on a bitmap resulting from the previous iteration, while sequential algorithms use an actual bitmap.

3. Contour approach

Pixels in binary images can be divided into 3 categories:

- background pixels
- contour foreground pixels
- other foreground pixels

During each iteration only contour foreground pixels can be deleted and so no other pixels need to be tested. Contour pixels usually represent only a small portion of pixels in engineering drawings and raster maps (especially drawing maps) so when all pixels are processed a large amount of processing time is wasted. For example, the image shown in Fig.1 consists of 338 322 pixels, but contour pixels represent only 12.5% of them.

The base idea of a contour approach is to process only contour pixels when conditions for deletion are tested. There are several contour tracing algorithms [14] which can be used in this process. To be able to describe the contour approach to thinning used in this paper following definitions are given:

Definition 1: The foreground pixel in a binary image is a black pixel which is a part of the important object.

* **Peter Tarabek**
Department of Transportation Networks, Faculty of Management Science and Informatics, University of Zilina, Slovakia,
E-mail: Peter.Tarabek@fri.uniza.sk

Fig. 1 Example of drawing map

Definition 2: The background pixel in a binary image is a white pixel which is a part of the unimportant background.

Definition 3: The 3×3 neighborhood of the pixel P is represented by the pixels P_1–P_8 as shown in Fig. 2.

P_8 [i−1, j−1]	P_1 [i−1, j]	P_2 [i−1, j+1]
P_7 [i, j−1]	P [i, j]	P_3 [i, j+1]
P_6 [i+1, j−1]	P_5 [i+1, j]	P_4 [i+1, j+1]

Fig. 2 Neighborhood of pixel P

Definition 4: The Contour Pixel is the foreground pixel whose 3×3 neighborhood contains at most 7 foreground pixels.

Definition 5: When contour is processed, the contour pixels are processed in clockwise order. Let P[i, j] be the processing pixel and P_7[i, j−1] be the previously processed pixel. A successor of pixel P is the first foreground pixel in its 3×3 neighborhood found in clockwise order starting from P_7 position. This means that the neighbors of P are processed in the following order: P_8, P_1, P_2, P_3, P_4, P_5, P_6, and P_7 to find the first foreground pixel. A predecessor is used to find a successor to ensure that the contour pixels will be processed in right order.

A general contour thinning algorithm can be described in 2 steps:

- recognition of contours
- iterative thinning of contours

In the first step the image is scanned to find contour pixels. When the contour pixel is recognized the whole contour is traced using the successor function. This function returns the successor of current pixel based on definition 5. In order to trace the whole contour the current pixel is marked as a predecessor and the successor is marked as a current pixel after the successor is found. This process continues until the contour is traced and all the accessed contour pixels are marked (colored) as used. This prevents from finding the same contour multiple times during the scanning process. To be able to use the successor function two pixels must be known. When the first pixel of the contour is recognized, the positions P_1, P_5, P_3 and P_7 are used to find the first background pixel. This background pixel is marked as a predecessor of the current pixel so the successor function can be used. Every contour in the image is stored for further processing by the first recognized contour pixel and its successor.

After the whole image is scanned, all contours are recognized. Next, the thinning process can begin. In each iteration, all the contours which were not marked as thinned are processed. The contour pixels are tested for deletion based on the rules and templates used by a specific thinning algorithm. A contour is marked as thinned if none of its contour pixels were marked for deletion in the previous iteration. When all the contours are marked as thinned the thinning process is finished.

In Figs. 3 and 4, the results of contour thinning using the basic deletion rules are shown. These rules mark the pixel for deletion if its connectivity number (number of black to white translations) equals to 1 and if the number of the foreground neighbor pixels is higher than 1 and less than 7.

The contour thinning algorithm based on the presented deletion rules produces distortions in junction points shown in Fig. 4. This is caused by the successor function which does not recognize all the contour pixels and produces the contour shown in Fig. 5. This contour approach to thinning (CT1) has its advantages and disadvantages. To process the contour, a smaller number of contour pixels need to be processed and what is more important the successor function and the whole algorithm are relatively fast and easy to implement as will be shown later. The disadvantage of this approach is a possible distortion in junction points. To deal with this problem, correct rules for deletion should be used. One possibility is to use two sub-iterations for pixel deletion allowing to access contour pixels which were not accessible in first or second sub-iteration.

Fig. 3 Skeleton of drawing map

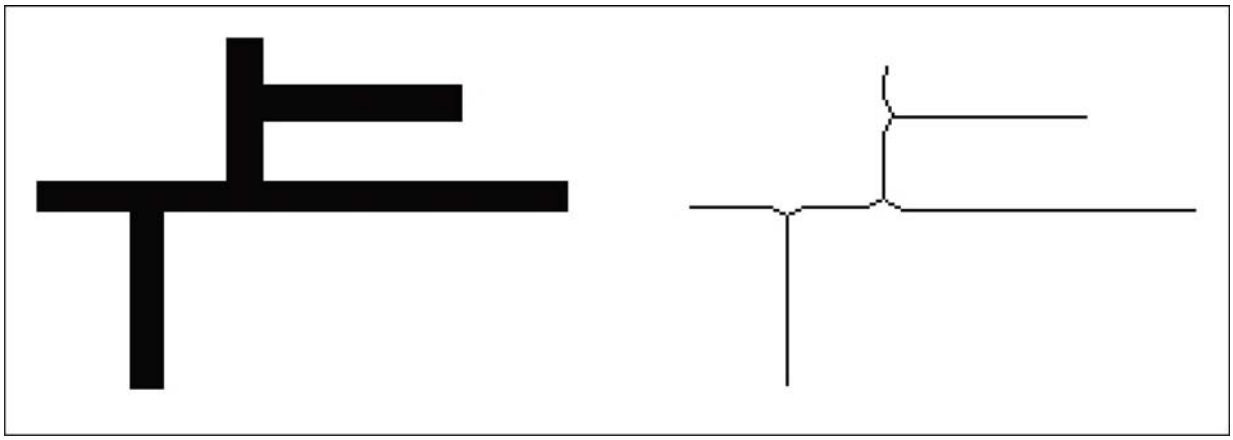


Fig. 4 Distortions in junction points

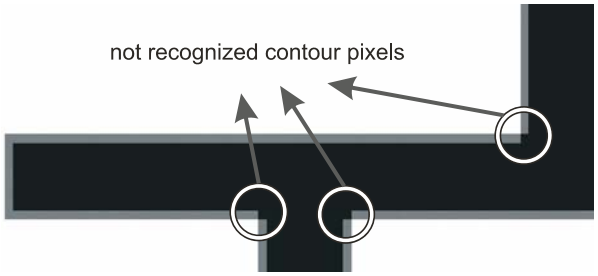


Fig. 5 Problem of contour pixels recognition

Another possibility is to improve the successor function to recognize all contour pixels. This function will use the same definition of a successor pixel (def. 5), but for all the successors which represent diagonal neighbors (P_2, P_4, P_6, P_8) of a current pixel, the next pixel in clockwise order is examined. If this pixel is a foreground pixel it is marked as a successor instead of the original one. Although this process seems to be easy, the contour approach to thinning based on such a successor function brings a lot of complications. The first problem is shown in Fig. 6. If the successor function is used, there is a high probability to create an infinite

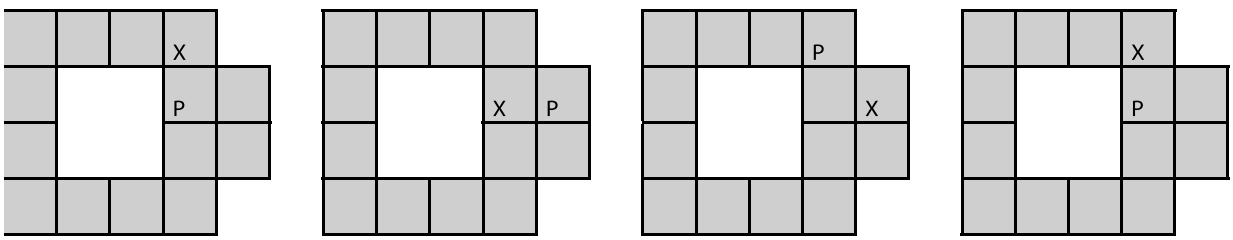


Fig. 6 Infinite loop

loop. In Fig. 6, four successive steps are shown. The current pixel is represented by 'P' and predecessor by 'X'. When the contour is processed using this successor function, an infinite loop is created.

So definition 5 of the successor has to be modified in order to prevent from creating the infinite loops.

Definition 6: When contour is processed, contour pixels are processed in clockwise order. Let P be the processing pixel, X be the previously processed pixel and XX be the predecessor from previous step (where X was actual pixel). The successor of pixel P is the first foreground pixel from its 3×3 neighborhood found in the clockwise order starting from X position which differs from XX pixel.

A new contour approach to thinning (CT2) can be defined using the new successor function and the successor definition 6. This approach uses information about the actual pixel (P), its predecessor (X) and a predecessor from the previous step (XX) to process contours. Using the information about XX pixel the situation in Fig. 6 can be solved (see Fig. 7).

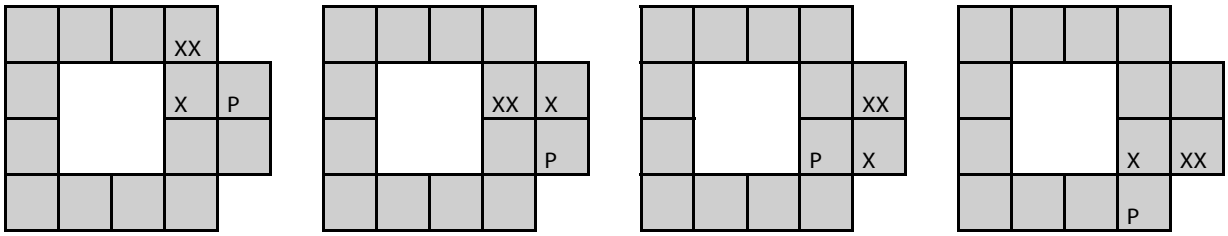


Fig. 7 Correct processing of contour

Information about all these pixels must be stored in order to correctly start and stop processing of contours. Sometimes X and XX pixels are at the same location. In this case XX pixel needs to be ignored when the successor function is used. Some other problems with implementation of CT2 are described in the next section. The result of CT2 using the same deletion rules as in CT1 is shown in Fig. 8.

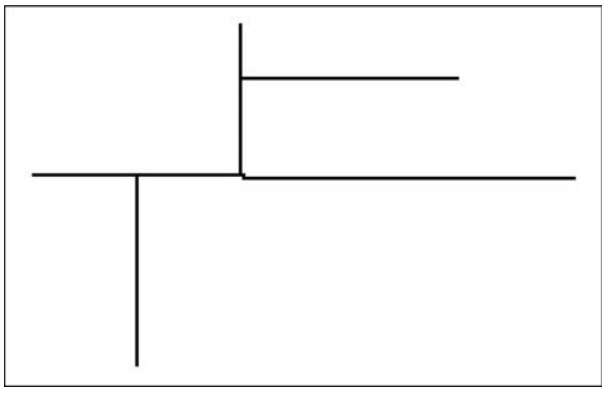


Fig. 8 Result of CT2

4. Implementation problems

Although some problems of CT1 and CT2 approaches were described in the previous section, there are still other issues to be solved. Critical points have to be defined before discussing these problems.

Definition 7: Critical points CP, CX and CXX are pixels which are stored for each contour and are used to start and stop contour processing. CP stands for the current pixel, CX stands for the predecessor and CXX represents the predecessor from previous step.

The first problem is to set "stopping rules". When the contour is processed, a position of the current pixel P is compared to CP and a position of the current predecessor X is compared to CX. When these positions match, processing of the contour is stopped. In CT2, also the position of XX is compared to CX.

When some of the critical points are deleted, their positions must be updated for further accurate contour processing. This process can influence quality of results in place of these critical points.

There are some other issues with deletion rules for both CT1 and CT2. One problem was shown in Figs. 4 and 5. In CT2, the most important problem is the problem of "staircase pixels". These pixels can belong to two contours and when they are deleted by one contour, critical points of other contour can be deleted too, making further processing of the second contour impossible. To deal with this problem we can search through all the contours to find out if this pixel represents a critical point of other contour or not. If the result is positive we can update critical points of the given contour. This process is time consuming. A better approach is to create rules which keep staircase pixels and remove them in a post-processing step. Because of the mentioned problems and some other issues with CT2 implementation, CT2 approach is hard to implement and its behavior is hard to predict for some cases. Also it is slower than CT1 so the CT1 approach was tested in experiments.

5. Experiments

Two tests were used to evaluate the CT1 quality. In both tests Zhang-Suen thinning algorithm [10] and CT1 are used. Zhang-Suen algorithm is often used as a reference algorithm and in our case it represents thinning algorithms which process all the image

pixels. Another advantage of this algorithm is that it uses two sub-iterations for pixel deletion. These deletion rules and two sub-iterations process can solve problems shown in Fig. 4 and so they were used in CT1 for these tests.

Test 1 compares quality of result based on performance measurements proposed in [11], where a thinning rate, number of components and noise sensitivity are evaluated. The second test consists of 4 cases and its purpose is to compare the processing time of both algorithms based on 4 different parameters of input images. These parameters represent percentage of background pixels (pBP), contour pixels (pCP) and non-contour foreground pixels (pFP) and a number of contours (NC). Detailed information about these images is shown in Table 1.

Parameters of input images used in test 2 Table 1.

	Dimensions	pBP	pCP	pFP	NC
Image 1	5000×8000	38.4%	7.4%	54.2%	11 984
Image 2	5000×8000	69.3%	4.0%	26.7%	8 270
Image 3	5000×8000	38.2%	0.8%	61.0%	138
Image 4	5000×8000	64.8%	0.6%	34.6%	155

6. Results

The image shown in Fig. 9 was used for test 1. Performance measurements show that the results for this input are the same for both algorithms (see Table 2.). More tests with the same criteria and manual verification were made. No fundamental differences were found in these tests.

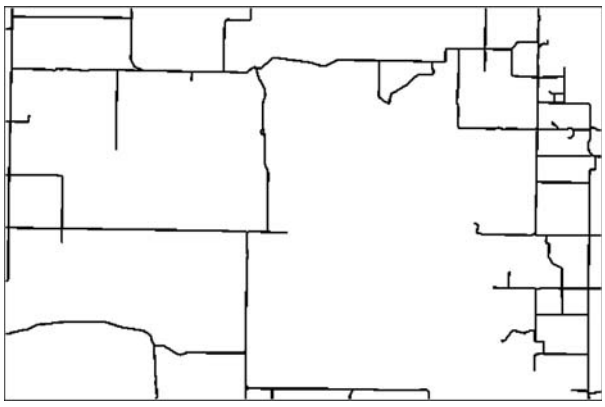


Fig. 9 Input image used in test 1

Performance measurements Table 2.

	Thinning rate	Number of components	Noise sensitivity
Zhang-Suen	693	1	62
CT1	693	1	62

The results of test 2 are shown in Table 3. For image 1 which consists of a large number of contours (11 984), the computation time is approximately the same for both algorithms. As the number of contours decreases and the number of background and non-contour foreground pixels increases, CT1 algorithm became faster than Zhang-Suen thinning algorithm. On the other hand, if images with more contours were tested, CT1 algorithm would perform much slower. This means that CT1 algorithm is much faster for shape objects the length of which is comparable to their thickness. When dealing with elongated objects, especially with objects the length of which is much larger than their thickness, the situation is more complicated. For example, engineering drawings and drawing maps usually have characteristics similar to image 2. They usually consist of a large percentage of background pixels, the number of contours is relatively high and they tend to have a lower percentage of non-contour foreground pixels. This situation can differ from image to image, but CT1 algorithm should perform faster for majority of them.

Results of test 2 (sec.) Table 3.

	Image 1	Image 2	Image 3	Image 4
Zhang-Suen	22.6	16.7	162.6	119.3
CT1	21.6	13.2	45.9	42.8

7. Conclusion

Two contour thinning principles CT1 and CT2 were presented in the paper. These approaches represent general principles which can be used with different deletion and implementation rules. Also they can be implemented as sequential or parallel algorithms. CT1 seems to be more robust. It can be easier to implement and perform faster than CT2. In section 6, CT1 and Zhang-Suen algorithms were compared. The CT1 algorithm was implemented with the same principles as Zhang-Suen algorithm (parallel nature, the same deletion rules and two sub-iterations). Both algorithms produce a skeleton with similar characteristics. Although when it comes to computational speed, CT1 tends to be faster for majority of input images (for all the images in our tests), there are still situations where the classical approach, processing all the pixels in the image, is faster.

References

[1] GOMIS, J. M., COMPANY, P., GIL, M. A.: *Vectorization in Recovering Engineering Drawings.* II Seminario Italo-Espanol, "Diseno y fabricabilidad de los productos industriales", Marina di Equa (Napoli), 24-26 de Junio de 1998.

[2] ARICA, N., YARMAN-VURAL, F. T.: *An Overview of Character Recognition Focused on Off-Line Handwriting.* IEEE Trans. Systems, Man, and Cybernetics-Part C: Applications and Rev., vol. 31, no. 2, pp. 216–233, 2001.
[3] PERVOUCHINE, V., LEEDHAM, G.: *Document Examiner Feature Extraction: Thinned vs. Skeletonised Handwriting Images.* Proceedings of The IEEE Region 10 Technical Conference, (TENCON05), November 2005.
[4] ZOU, J. J., HONG, Y.: *Vectorization of cartoon drawings.* Selected papers from the Pan-Sydney workshop on Visualisation – Volume 2, 2000.
[5] AH-SOON, CH., TOMBRE, K.: *Variations on the Analysis of Architectural Drawings.* Fourth International Conference Document Analysis and Recognition (ICDAR'97), 1997.
[6] YAO-YI CH., KNOBLOCK, C. A., CHING-CHIEN CH.: *Automatic Extraction of Road Intersections from Raster Maps.* In Proceedings of the 13th ACM International Symposium on Advances in Geographic Information Systems, 2005.
[7] HASTHORPE, J., MOUNT, N. J.: *The generation of river channel skeletons from binary images using raster thinning algorithms.* Proceedings of the GIScience Research UK 15th Annual Conference, 2007.
[8] NG, G. S., ZHOU, R. W., QUEK, C.: A *Novel Single Pass Thinning Algorithm.* IEEE Transaction on System Man and Cybernetics, 1994.
[9] BERNARD, T. M., MANZANERA, A.: *Improved Low Complexity Fully Parallel Thinning Algorithm.* Proceedings of the 10th International Conference on Image Analysis and Processing, 1999.
[10] ZHANG, T. Y., SUEN, C. Y.: *A fast parallel algorithm for thinning digital patterns.* Commun. ACM, 27(3), 1984.
[11] TARABEK, P.: *Performance measurements of thinning algorithms.* Journal of Information, Control and Management Systems, Vol. 6, 2008.
[12] TOMBRE, K., AH-SOON, C., DOSCH, P., MASINI, G., TABBONE, S.: *Stable and Robust Vectorization: How to Make the Right Choices.* Graphics Recognition – Recent Advances, 2000.
[13] LAM, L., LEE, S. W., SUEN, C. Y.: *Thinning Methodologies – A Comprehensive Survey.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 9, September 1992.
[14] GHUNEIM, A. G.: *Contour tracing.* Project for the Pattern Recognition course, http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/index.html

Acknowledgement:
This work has been supported by grant VEGA No. 1/3775/06.

Peter Caky – Juraj Boron – Martin Klimo – Katarina Bachrata *

MATHEMATICAL FORMULAS IN TEXT TO SPEECH SYSTEM

A modular based system for synthesis of a written text allows solving partial problems separately. If there is designed a logical structure with clear interfaces, it could be possible to prepare modules for variant processes by different members of the team, without necessity of the coordinated work. The paper deals with a module for engine reading of mathematical formulas in a Slovak text-to-speech system. The problem consists in the conversion of formulas to the written text in Slovak language and consequently additional conversion to the phonetic form. The main idea is to use the markup language LaTeX.

1. Introduction

The communication between computers and people as end-users is most frequently based on writing and reading. Computers can perform written instructions and the users can read the result of computer's proceeding on monitors or printing output. But there are many situations when people need vocal communication with computers. People could be concentrated on some activity and they are not able to read or write (e.g. driving, surgery major), or people could have some healthy problem, like weak eyesight or dyslexia problems. For such reasons there were created tools for speech recognition and speech synthesis. Computers are able to create artificial sound which is similar to real speech. The text to speech synthesizer (TTS) is a system for conversion of text documents to audio files. Such a synthesizer (TTS KIS) is developed in the Department of Information Networks at the Faculty of Management and Informatics at Zilina University. The TTS KIS system is a modular based system processing the text with modules, converts it to phonetic transcription and prepares it for audio processing.

The final step of conversion is a preparation of artificial speech composed of small blocks of natural speech. There are many properties which have to be keep to prevent the production of a synthetic machine sounded speech. Such a speech is produced with identical sounds. People cannot produce identical parts of speech. They are not able to say two identical vowels "A". Therefore, an important task for synthesis is the transcription of a text, e.g.: "OKNO SA POOTVORILO" in such a way. that every letter "O" sounds a little bit different. On the other hand, the database including all the words of the Slovak language (and all the different forms of every word) is very large. Contrary to the English language, here it is not possible to use the database with complete words. So, there is a database of diphones (two joined phones cut from an artificial word) and synthesized words are combined from these diphones. The phonetic transcription has to be prepared with specified properties for every word and sentence. The properties like melody, pronunciation, phrase breaks, volume, intensity, pitch must be described. These properties are controlled by separate modules in the TTS KIS system. Modules give instructions for the next process with the text and put them into phonetic transcription. The module described in this article deals with mathematic formulas and expressions. It translates formulas written in the LaTeX language into a phonetic form suitable for next processing.

The TLatex module which is responsible for reading mathematical formulas is designed and implemented into the TTS KIS real system and enhances the possibilities of the TTS synthesizer. In the paper we will describe general principles of modular processing of the text by using Speech Synthesis Markup Language (SSML). Then we will describe the LaTeX program as a markup language and will take advantage of their similarities.

2. Description of TTS KIS system

Speech synthesis is based on concatenation of diphones, the sound formed from two phones running from the center of the first phone to the center of the second one. In this sound a natural transition from the first phone to the second phone is recorded. Connection of the phones is realized in the centre of the phone, where the behavior of the phone as a time process is well predictable. The modular based program for text-to-speech uses a kernel developed by the team of researchers and students of the faculty. The database of diphones (approximately 2000 files) of the Slovak language was prepared for this system. The architecture of the system is based on SSML standards, Speech Synthesis Markup Language [1]. It is designed to provide a rich, XML-based markup language for assisting the generation of synthesis speech in Web and other applications. The important function of the markup lan-

* **Peter Caky, Juraj Boron, Martin Klimo, Katarina Bachrata**
Department of Information Networks, Faculty of Management and Informatics, University of Zilina, Slovakia, E-mail: peter.caky@gmail.com

guage is to provide a standard way to control properties of speech in different synthesis-capable platforms.

The input to the synthesis processor is a text prepared by a user or produced automatically. SSML standards define a form of the document. Next, there are few processing steps made by the synthesis processor to convert a marked-up text input into an automatically generated speech output. The markup language is designed for control over each of the steps described below and the document author can control the final voice output: XML parse, structure analyses, text normalization, text to phone conversion, prosody analyses and waveform production.

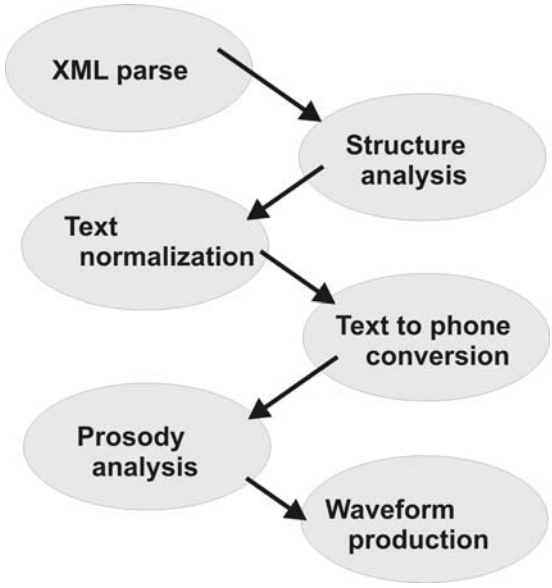


Fig. 1 Steps in the process of speech synthesis based on SSML

The marks in SSML language can be more or less structured. A very simple input to the synthesizer could have the following form:

```
<speak>
   Tento text ma byt precitany po slovensky.
Preto sme ho napisali po slovensky.
</speak>
```

The form of the input can be more complicated. The SSML is an annotation tool and it is possible to add next information. We can specify that between <speak> and </speak> elements there are two sentences. The input will be more specified:

```
<speak>
   <s> Tento text ma byt precitany po slovensky.
   </s>
    <s> Preto sme ho napisali po slovensky. </s>
</speak>
```

A user could continue and put next marks into the text for optimization the waveform output. After this processing the output will satisfy the user's conditions. The next example of a marked text illustrates a more detailed description of additional informational data:

```
<speak version”1.0” xml:lang=” en-US”>
   <lexicon
   uri="http://www.example.com/lexicon.file"/>
        <voice gender="female" variant="2">
          <p><s> Next text has to be read in
          Slovak language. </s></p>
          </voice>
</speak>
<speak version”1.0” xml:lang=”sk”>
        <voice gender="male" variant="1">
          <p><s> Tento text sme napisali po
          slovensky. </s></p>
          </voice>
</speak>
```

As shown the SSML gives possibility to set a lexicon, a voice, sentence boundaries that will be used for a grapheme to phonetic transcription. The annotations that SSML provides can be divided into three groups:

1. Document structure, text processing and pronunciation: <speak>, <language>, <lexicon>, <phoneme>, <say-as>, <p>, <s>.
2. Prosody and style: <voice>, <emphasis>, <break>, <prosody>.
3. Other elements: <audio>, <mark>, <desc>.

The requirements that appeared during the development of TTS KIS system lead to the extension of the SSML language to SSML+. This extension adapts the SSML to requirements for the TTS system which works with texts in the Slovak language. There are new elements and new attributes of the elements existing in the SSML+. It allows to structure texts in a special way typical for the Slovak language. Texts can be segmented more precisely to compound sentences, sentences, words, syllables and diphones. Due to this segmentation it is possible, for instance, to assign melody for particular sentences or words, which is important in Slovak pronunciation.

In order to achieve the complete SSML+ structure, there were defined and implemented modules for adding the necessary annotations into the input SSML text. Rule-based engines (RBE [2]) represent a way how this task can be accomplished. There was defined a set of atomically and independently executable modules. Each module M is determined by two sets of conditions:

1. Preconditions - a set of conditions that have to hold before a module code is executed
2. Postconditions - a set of conditions that must be true just after the execution of a module code

These modules are executed one by one in an undefined order depending on the result of module preconditions. The method of execution in which the order of module invocations is not explicitly defined is called the implicit invocation [3]. The order in which modules are executed is determined implicitly according to the

preconditions and postconditions and modules run till some precondition has changed.

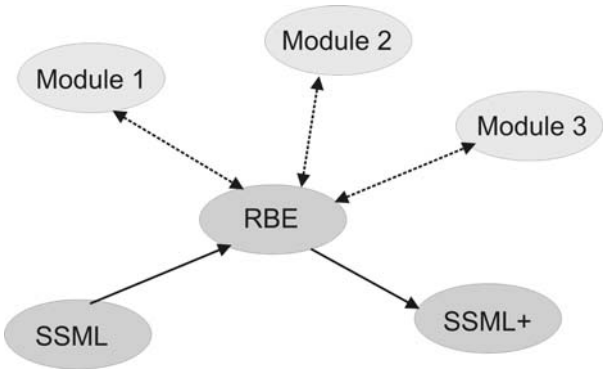


Fig. 2 Execution of modules based on preconditions

An advantage of the implicit invocation is its independence of the modules. Not only during the program running but also during the module development. The modules can be prepared separately. It is only necessary to define preconditions and postconditions for each module. Then it is easy to add the modules to the system. The TTS speech synthesizer uses a concatenative approach [4], [5] and some of the modules are used cause of the concatenative method:

1. Validation of the SSML and structure creation (speak, ssml, audio)
2. Structural analyses (lang, voice, paragraph, compound, sentence, sentence type, word, diphone, text norm, text, syllable, say as, sub, desc, mark, phoneme)
3. Grapheme to phoneme conversion (lexicon, ph, word join)
4. Prosody analyses (prosody, contours, range, pitch, duration, rate, volume, emphasis, voice, break)
5. Waveform production (diphone extract, diphone selection, diphone merge, contour application, prosody application)

But there are also modules for structural analysis and normalization of the text. The shortcuts, numbers, dates, marks are conversed into the text in these modules. We will focus on the Tcitac module[10] that executes mathematical expressions written in the LaTeX typesetting system.

3. Reading of mathematical expressions

From the synthesis point of view, a mathematical expression is a set of symbols, letters, numbers, formulas and relation signs which express some mathematical properties or data. Parts of sucha text could be numbers, variables, fractions, elementary functions, integrals, sums and whatever else that mathematicians can think up. People know the way how to read the expressions, they know where they have started and where the end of the expression is. If an expression is to be read automatically, people will have to teach computers how to converse a mathematical expression into a plain text. But there are many ways how to read the same expression also in the Slovak language.

A very easy example is the fraction 1/2. It is possible to say only “half”, but for the same expression we can use “one half”, “one over two”, “fraction one over two”, “fraction where numerator is one and denominator is two”. The most complicated way how to say 1/2 is the sentence:

> Begin of the mathematical expression start of the fraction start of the numerator one end of the numerator over start denominator two end of the denominator end of the fraction end of the mathematical expression.

People do not use the last way of reading the expression 1/2 but for complicated fractions this is a good solution how to do it. There are two choices in the Tcitac module for reading expressions: “simplified” reading (like “half”) and “normal” reading (as we could see above). The “simplified” reading is choice for reading texts with small numbers of expressions or for quick reading for first overview of the text. Technical articles require “normal” reading. The text must be read by separate steps. The next part of the text is read after finishing the previous part. If there are many levels in one part, they have to be read progressively from complicated to simpler parts. After reading the elementary part the system continues by reading on a higher level.

$$\lim_{n\to\infty}\sum_{k=1}^{n}\frac{1}{k^2}=\frac{\pi^2}{6} \qquad (1)$$

There is a scheme of the way of “normal” reading of the formula (1) in Figure 3. At this scheme there are arrows denoting progress of the steps of reading blocks of an expression.

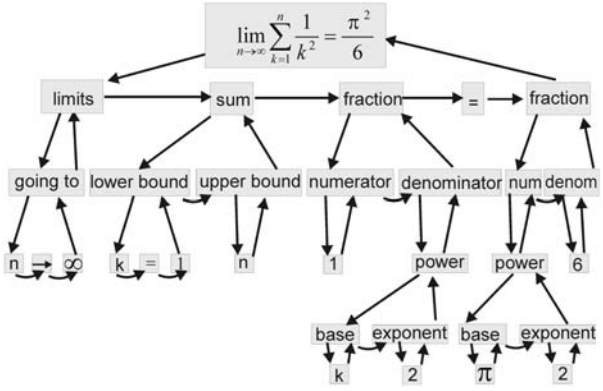


Fig. 3 Scheme of the steps in reading of mathematical formula

The final text for normal reading will be prepared in the form:

> Begin of expression limit for en going to infinity from sum for kej equal to one till en from fraction start of the numerator one end of the numerator over start denominator k power to two end of the denominator end of the sum end of the limits is equal to fraction start of the numerator pi power to two end of the numerator over start denominator six end of the denominator.

The "simplified" reading uses the same scheme of reading the expression. Short forms of the formulation are used. The final text for "simplified" reading will be prepared in the following form:

```
Limit for en going to infinity from sum for kej equal to one till en from one over square of is equal to square of pi over six.
```

A "simplified" form is easier to use but there can be some mistakes in understanding. A better solution is to use a combination of these two forms. The first "simplified" reading and then if there are some doubts about understanding the "normal" reading follows.

4. Mathematical expressions in LaTeX typesetting system

LaTeX is a markup language for preparing documents for the TeX typesetting program designed by Donald Knuth, [6], [7]. LaTeX is most widely used by mathematicians, scientists and academic society. The typesetting system offers extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout and bibliographies. LaTeX was originally written by Leslie Lamport [8]. Both of them LaTeX and TeX are open source systems. They say about LaTeX that it is a document preparation system for high-quality typesetting. LaTeX encourages authors not to worry too much about the appearance of their documents but to concentrate on getting the right content. One of the greatest motivating forces for Donald Knuth when he began developing the original TeX system was to create something that allowed a simple construction of mathematical formulas, whilst looking professional when printed. The expression (1) written in LaTeX can be seen in Figure 4.

$$\lim_{n\to\infty}\sum_{k=1}^{n}\frac{1}{k^2}=\frac{\pi^2}{6}$$

Fig. 4 Output of mathematical formula written in LaTeX

A LaTeX document consists of one or more source files with extension .text and files which can be written in an optional text editor. In this file there is a text and commands commonly start with a backslash and are grouped with curly braces. We will focus on writing mathematical formulas in LaTeX. LaTeX needs to know beforehand that the subsequent text does in fact contain mathematical elements. This is because LaTeX typesets math notation differently than a normal text. Therefore, special environments have been declared for this purpose. For declaration special marks are used allowing to recognize mathematical formulas in the text. There are special marks in LaTeX for mathematical symbols, variables and characters. Formula (1) can be expressed as:

```
$$\lim\limits^{n\to\infty}\sum\limits_{k=1}^n\frac{1}{k^2}=\frac{{\pi}^2}{6} $$
```

Other ways how to declare mathematical environment are pairs of mark like $ ... $, \begin{math}... \end{math}, \begin{displaymath}... \end{displaymath}, \begin{equation}... \end{equation}. The LaTeX commands are sorted to several classes according to the way in which they are read:

- signs without special commands in mathematical environment (numbers, letters, + * > < ( ) [ ] : " ' /)
- pairs of the marks for mathematical environment ($...$, \begin{displaymath}... \end{displaymath}, \begin{equation}... \end{equation}, \begin{eqnarray}... \end{eqnarray}) and similarly pairs of the marks for norm, absolute value and curly brackets
- non reading commands like spaces, labels, size and style of the characters, commands for citing, index and references
- non parametric commands (\infty, \alpha, \%, \dots, n \choose k, \subset, \to, \into, \partial, \bot, \rightarrow, \forall, \neq, \&)
- standard commands for functions (\lim, \det, \min, \cos, arsin)
- letters of Greek alphabet (\alpha, \beta, \gamma, \epsilon, \varepsilon, \pi, \phi, \varphi)
- commands after parameter (\overline{x}, \overbrace{x}, \underline{x}, \dot{x})
- commands with parameters like exponent, index, root, limit, product, sum, integral, fraction, vectors (x^{2\pi}, x_{ij}, \sqrt[3]{9}, \lim_{n\to \infty}{\fraction 1 n}, \prod_{k=0}^{5} {x_{k}}, \sum_{k=0}^{\infty} {n^2}, \int_a^b x^2 dx , \frac1 5, \vec{v})

We will describe principles of the Tcitac module of the TTS KIS synthesizer in the next section.

5. Description of Tcitac module

The Tcitac is a module working with mathematical expressions written in LaTeX. The module converts mathematical expressions to the SSML+ format prepared for reading. More precisely, the pre-condition of this module is at least one appearance of LaTeX mathematical environment. The postcondition of the Tcitac is a phonetic transcription of all the LaTeX formulas. The module consists of nine classes shown in the UML diagram in Figure 5. The Class Texformula is an interface of the module. In this class we detect if the precondition of the module is satisfied. Class Ttexty contains commands with their phonetic transcriptions, class TNumbersTransformer converts numbers, class TTexTransformer finds the beginning of the formulas and creates an instance TProstredie. Tprostredie finds LaTeX commands and creates an instance TCast for conversion of individual commands. Tcast executes the command recursively and after finishing it it returns the control back to the upper level class TProstredie and the results of TProstredie are returned to TTexTransformer and then to Texformula. Texformula continues in scanning all documents and returns the changed SSML document and finishes its proceeding. The instance of class Tprikaz is used to transmit information about the current proceeding of translation of the command. The instance of class Tznak helps in sequential reading of the signs of the document.

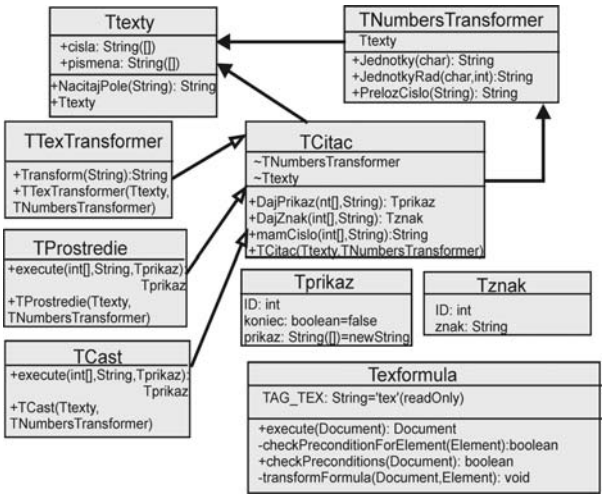


Fig. 5 UML diagram of the classes of module Tcitac

6. Conclusions

The Tcitac module does not have a fixed programmed list of the LaTex commands and a way of how to read them. The information about reading is loaded from the input files. It is easy to add new commands or to edit some existing ones. Possibilities of changes of the phonetic transcription are very important properties of the Tcitac module. There are many exceptions or different ways in reading mathematical formulas. The improvement of reading can be done by including exceptions to the lexicon of the TTS KIS system [11]. Exceptions and mistakes in the phonetic transcription can be included into the learning of neural networks. Works [12], [13] can be extended from reading the text to reading formulas and mathematical expressions.

References

[1] Speech Synthesis Markup Language (SSML) Version 1.0, http://www.w3.org/TR/speech-synthesis
[2] Rule Based Engine, http://en.wikipedia.org/wiki/Rules_engine
[3] AI Application Programming, M. Tim Jones, 2003 Charles River Media, ISBN 1584502789
[4] Improvements in Speech Synthesis: Cost 258: The Naturalness of Synthetic Speech, Eric Keller, 2002 John Wiley and Sons, ISBN 0471499854
[5] CAKY P., KLIMO M., MIHALIK I., MLADSIK R.: *Text, Speech and Dialogue 2004,* 7th International Conference, TSD 2004, Text to Speech for Slovak Language, pp. 291-298, ISBN 978-3-540-23049-6, 2004 Springer, ISBN 3540230491
[6] KNUTH, D. E.: *The TeXBook,* Massachusetts: Addison-Wesley, 1984), x+483pp. ISBN 0-201-13448-9
[7] KNUTH, D. E: *Typesetting Concrete Mathematics,* TUGboat 10 (1989), 31–36, 342. Reprinted as chapter 18 of Digital Typography.
[8] LAMPORT, L.: *LaTeX: A Document Preparation System,* Addison-Wesley. Retrieved on 2007-02-02, 1986.
[9] KLIMO, M., MIHALIK, I.: *Extending annotational SSML into structural content for speech synthesizer (in Slovak),* Scientific Papers of the University of Pardubice, 12 (2006): Series B-ISBN 978-80-7194-985-5, p. 193–199.
[10] NOVAK, A.: *Mathematical LaTeX Formulas Reading in TTS (in Slovak),* dipl. práca, 2007
[11] CAKY, P., KLIMO, M., MIHALIK, I., MLADSIK, R.: *Text-to-speech for Slovak language (in Slovak)* Proc. of Text, speech and dialogue: 7th international conference, TSD 2004, Brno, Czech Republic, 2004, Berlin: Springer, 2004, ISBN 3-540-23049-1, pp. 291–298.
[12] HUDEC, M.: *IT in Software Compensation Applications (in Slovak),* Ustav vedy a vyskumu UMB, 2006, Banska Bystrica ISBN 80-8083-196-3
[13] HUDEC, M.: Compensation Multi-voice Synthesis with Neuro-computing (in Slovak), Banska Bystrica, 2005, ISBN 80-8083-103-3

Hynek Bachraty – Emil Krsak – Marek Tavac *

ALGORITHM FOR GENERATING TEXT DESCRIPTIONS OF BIT CALENDARS

The article deals with the problematics of generating calendar text descriptions based on their bit map. We introduce the description of the algorithm we have designed, its basic ideas and parts. We also present the results of testing the successfulness of the algorithm before it is primarily used.

1. Problem characterization

Calendar data are a significant part of information on provided services, executed activities, frequent events etc. In connection to our work we come across them particularly in the area of transport, where processing and displaying them has been an important component of many information systems. Most frequently, they describe means of transport operation days or validity days of some further transport characteristics. Therefore, in our article we are going to talk about the calendars describing a train operation days. But the text-generating algorithms created and described hereby are to a great extend universal, and by changing the text being used it is possible to adapt the outputs also to other, different situations.

Most frequently, calendar data are entered and stored in two ways: in bit maps and by means of text descriptions. We use the term bit map to denote such form of information which, for each day of a particular period, sets whether a train operates or not on a given day. The calendar then can be expressed (and this is also how it is usually represented in a data way) as a sequence of ones (1s) (the train operates) and zeros (0s) (the train does not operate) corresponding to individual days [1]. A part of the information on calendar is the beginning and the end of its validity period. This, most often, also secures the link-up between the data and real calendar.

The calendar defined by a *bit map* is optimal from the viewpoint of informatics. By means of the bit map it is easy to edit or find out about a train operation days, to compare and combine data of several calendars, to find out about their characteristics etc. Basically it is possible to picture only graphically, which is reasonable to do only on a monitor of sufficient size. The user needs to have necessary equipment, as well as time to use such way of the data information display. This form of a calendar set-up is, on the other hand, unsuitable for mass print outputs, depiction on small screens (e. g. PDA devices), for other than graphical transfer of train rides information etc.

Therefore, the calendars defined by a bit map are suitable to work with for information systems and specialized users who, as a part of their job, create, change, analyse etc. date data. Reversely, they are not suitable for mass usage, for common human interaction and so on.

A calendar *text description* has been most frequently used in date annotations in various types of timetables and other information utilities. This date information description attributes are basically opposite to those with the bit map. The contents of the calendar are composed into a relatively short text. That is why it suits in print outputs or for small displaying devices. The calendar validity is possible to communicate verbally in an easy way. Regarding our experience with using this form of date description, we can easily gain some types of information from the text description, e. g. whether a train operates tomorrow, next Monday or on Christmas Day. We would, on the other hand, have problems, if we were supposed to use the text description to define the number of a train operation days within the observed period, to find out which of the two given trains operates more frequently, or on which days both of the trains operate.

A trully existing problem which we needed to solve lies in the algorithmic interconnection between the two means of presenting date information. To the ways of work used in the transport area mostly applies that first of all bit calendars are at disposal, and they consequently need to be expressed in a text form. Bit maps are gained either from information systems for transport planning, or gained from international, whole-Europe databases, where using text expressions is faced with language problems. In the article we are describing an algorithm for generating text descriptions from

* **Hynek Bachraty, Emil Krsak, Marek Tavac,**
Department of Software Technology, Faculty of Management and Informatics, University of Zilina, Slovakia, E-mail: Hynek.Bachraty@fri.utc.sk

a calendar bit map, which has been created for the above purposes, as well as the first results of testing the algorithm.

2. Solution starting points

The basic task is to generate text corresponding with a given calendar bit map. For the bit map we also suppose setting the initial and finishing days of the calendar validity period. These at the same time determine the length of the validity period. As long as we are considering several calendars, we assume identical initial and finishing days of the validity of all of them. In case this condition is not complied with, we are able to provide it formally by prolonging the validity periods of all calendars to the maximum used extent. The algorithm for genereting text we have designed can manage unifying the validity period by extending it without any difficulties.

Further on, we assume we are solving the problem for calendars used in the area of public personal transport. They therefore serve a large group of citizens as important access information in relation to a certain service. The service presupposes a certain degree of their regularity, simplicity and lucidity. English speaking people would say the operation days have to have created a certain ´pattern´. Otherwise date information not lucid enough would make using the service difficult. Searching and finding the regularity, or pattern, is the essence of the problem solution.

The algorithm is of course able to process a random calendar from a different area, too, e. g. a calendar of cargo or special transport, service calendar etc. But here we can expect lower effectivity and success rate of the algorithm. However, a narrow group of users of such special calendars enables to presume using the specific possibilities to display the calendar, e. g. graphical picturing, extensive text, possibility to enquire about individual operation days etc.

The calendars text descripitons which are to be the algorithm output also have to meet certain principles. The text needs to be clear, explicit, structured and standardized. As far as it is supposed to describe a wider type range of calendars, it will be necessary to use several text types. Nevertheless, their number should not be too high. Recognizing a calendar type and consequent assignement and specification of a particular text type is another key goal of the solution. The text needs to have a structure, preferably common for all of the types, and not extensively complicated. It should contain a ´pattern´ part where the information gets compressed most remarkably, by means of the found calendar regularity description. The regularity, however, will not always be possible to achieve absolutely exactly, and it will be necessary to specify exceptions to the regularity. Both information need to be expressed in a compact and inter-related way, not to confuse the user.

Important criteria of the created algorithm quality will be the algorithm ability to generate text as short as possible or at least texts of acceptable length, and as high as possible share of calendars which the algorithm will be able to generate a text for.

3. Determining the calendar regularity

Searching a pattern of calendar validity is based on the weekly rhythm of our life. The basic patterns correspond to the repetition of individual weekdays (Monday to Sunday). The basic bit map is compared to all 128 combinations of the seven basic calendars. They are so called sample calendars. If necessary, the 7-day cycle is possible to modify into a different number of days, although another one than the 7-day calendar is in fact not used [2]. This option can be used for example when processing calendars related to others than weekly maintenance cycles of different equipment.

To detemine the most apropriate sample calendar, as a criterion we have used the lowest possible number of exceptions, i. e. days of the entered calendar the validity of which does not correspond to the tested sample calendar. They might be so called positive exceptions (the train also operates on the days beyond the tested range) and negative exceptions (the train does not operate on the days of the tested sample). In optimal case the tested sample calendar fully agrees with the entered calendar and the number of exceptions equals zero. It that case it is possible to terminate the testing. On the contrary, one of the algorithm parameters is the value of the ultimate acceptable number of exceptions. As far as it is exceeded, the tested sample calendar is not considered as conforming. It is possible we will not consider any sample calendar as conforming, and searching for regularity will not be successful.

When testing calendars in connection to a particular country, it is apropriate to complement two more sample calendars. They are so called holidays (Sundays and bank holidays) and working days (Monday to Friday, holidays excluded), which we usually mark with + and X. Thus we gain as many as 9 basic calendars that we combine with each other. However, there is no point in combining the holiday's calendar with the calendars containing Sundays, and the working days calendar with the calendars containing only Monday to Friday days.

The description of the found sample represents the main component of the generated text. The way the week days are marked is optional, most frequently they are abbreviations or serial numbers of individual days (Mon to Sun with the sequence 1 to 7). With a particular algorithm implementation it is possible to use special fonts for the marking. Standardly the text looks as follows:

´The train departs on Mon, Thu, Fri´ or ´The train departs on (1), (4), (5)´.

In some special cases simplified and commonly utilized texts are used, such as *´The train operates daily´, ´The train does not operate´, ´The train operates on Mon to Fri´* or *´The train operates on Sat and Sun´*.

Eligibly it is possible to use also so called negative notes. As long as listing the weekdays is too long, a modified text can be used.

´The train operates daily except (3), (4)´ or *´The train does not operate on (3), (4)´* instead of *´The train operates on (1), (2), (5), (6), (7)´*.

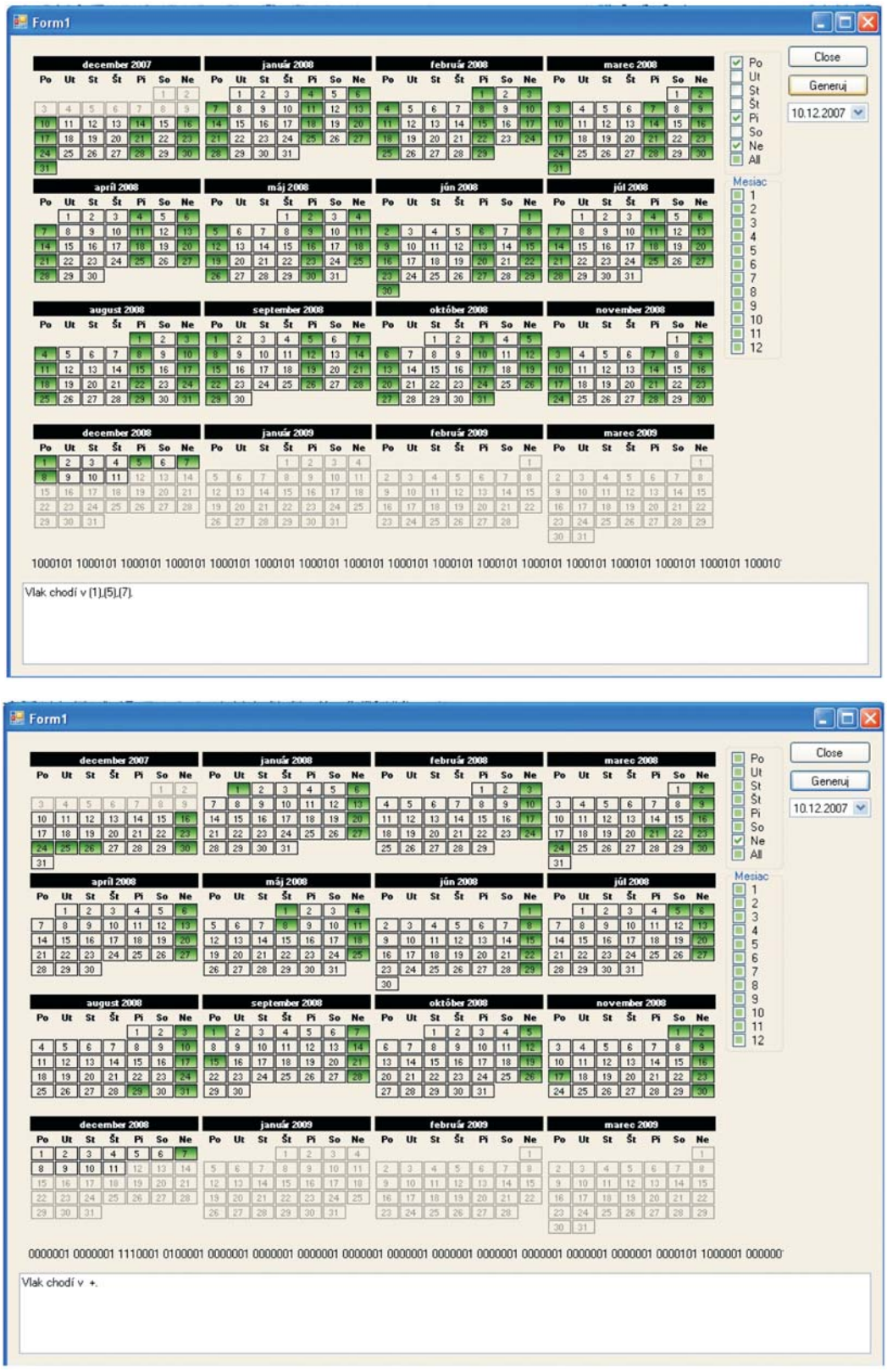


Fig. 1 GUI of bit map editting component.
Example of simple ´Mon+Fri+ Sun´ regularity and ´holidays´ sample calendar

With the second text above the information that the train operates on all days but Wednesdays and Thursdays is only implicit, therefore the text is shorter but less precise. A small explorative enquiry has shown that naming three days in a negative form is not accepted by users, that is why we use it only for one or two ´negative´ days.

The regularity description is followed by listing exceptions as follow:

´The train operates on (5), (6), except 20 XII, 14 III and 18 IV. The train operates on 15 I and 14 V´ or *´The train operates daily except (5) and does not operate on 20 XII, 14 III and 18 IV. The train operates on 15 I and 14 V´.*

We have selected the approach of the negative exceptions are introduced as rather positive, right after the regularity description. This is based on the experience that users, having gained positive information, are likely to finish analysing the calendar text description. Therefore they might wrongly assume the train operates on the given day, although the opposite is true. That is why the negative exceptions are introduced as soon as possible and within one sentence together with the regularity description. On the contrary, in case of not being successful our mind has a tendency of continuing to explore, thus the positive exceptions may ensue in the following sentence.

When listing the exceptions, the list of individual days conforms with specific rules. Continual stretches of the days following one after another are indicated just by the first and last days. Marking a month is used only when it gets changed, and marking a year is used only in case the given day repeatedly falls into the effectiveness period.

´The train operates on 10 XII 2008 – 12 XII, 19 – 23 II, 13, 14 V, 29 VI – 3 VII, 9 – 11 XII 2008.´

The example above also illustrates one of the options of text-generating, in case a suitable regularity has not been found, therefore the algorithm has failed. Then we deal with listing all of the valid caledar days. Such text, however, often happens to be long and not always possible to use. That is why there is another possiblity – generating an empty chain or alerting text, e. g. *´The calendar text description is not available´.*

4. Determining the calendar type

The above described way of regularities searching brings good results in case the train operates with certain regularity in the whole, most frequently 1-year period of the calendars validity. It corresponds to the validity period of the tested sample calendars. Nevertheless, as long as the train operates on Mondays and Fridays only during the school year, the high number of exceptions due to the holiday periods would, from the year-round point of view, refuse this regularity. This fact has brought us to the idea of tipping-out and consequently searching certain calendar types in the given bit map. The types should identify the period within which the train operates, and search and test the regularity within this period only. Consequently, an appropriate type text, complementing the information on the found regularity, will be generated for each calendar type.

We have decided to maintain the number of types low, for lucidity´s sake, however, at the same time we have to be able to use them to describe as many bit maps as possible. That is why we have also decided to resist the possibility of using recursive types searching, which is easy to implement from the viewpoint of programming. Our algorithm is currently using 5 types described further on. The assigned bit map is tested for its applicability to individual types, and it can also fall under more of them. As long as this happens, the type with the lowest number of exceptions from the found regularity gets selected. In case of an identical number of exceptions, as simple as possible type will be used. That corresponds to the sequence we have listed them in.

The basic type is so called *year-round calendar.* In this case, the text introduced in the previous part is generated.

The following type is the calendar with one ´operates´ period. It corresponds to the calendars according to which before and/or after the train operation-days period, i. e. at the beginning and/or at the end of the validity period the train does not operate at all. The minimal scope of the period in which the train (with consequently searched regularity) operates, is set as the algorithm parameter. For this type of calendar the texts are generated as follow:

´The train operates only from 1 XII to 31 VII on Mon, Thu, Fri´, ´The train operates only from 1 XII to 31 VII daily except (2), (6)´, or with exceptions *´The train operates only from 1 I to 31 VIII daily except Tue, Sat and does not operate on 9 IV, 14 V and 27 V. The train operates on 14 IV, 14 VII and 15 VII´.*

A developed version of the above type is the *calendar with two ´operates´ period.* In calendars like these, instead of one there are two ´the train operates´ periods, separated from each other by the period when the train does not operate at all. The generated texts look as follow:

´The train operates from 4 IV to 31 V and from 4 VII to 31 VII on (1), (5), (6), (7)´,

´The train operates from 2 V to 30 VI and from 1 IX to 31 X daily except (4)´,

´The train operates from 30 I to 30 IV and from 1 VII to 30 IX daily except 31 I and 15 VII´.

Another calendar type is the *calendar with a ´does not operate period´.* In this case the bit map serves to find one period within the validity period in which the train does not operate at all. The minimal range of the ´does not operate period´ is again given by an eligible parameter. The texts are generated as follow:

´The train, except the period from 1 IV to 2 VIII operates on Mon to Fri´, ´The train, except the period from 1 V to 31 VII operates daily except Mon, Sun.´ or *´The train except the period from 1 IV to 2 VIII*

Fig. 2 Example of one and two ´operates´ period calendar types

Fig. 3 Example of ´doesn´t operates´ and ´operates daily´ period calendar types

operates daily except Sat, Sun, and does not operate on 19 II, 11 III and 19 VIII. The train operates on 26 X, 8 XI and 22 XI´.

The last calendar type is the *calendar with ´operates daily´ period.* In this case the bit map needs to contain sufficiently long period in which the train operates daily. This period is introduced in an independent sentence at the end of the generated text.

´The train operates on (1), (3). In the period from 1 VI to 31 XII the train operates daily.´ ´The train operates on (2), (3), (6) except 23 I and 16 IX. The train operates on 30 X and 13 XI. In the period from 1 VII to 31 VIII the train operates daily´.

When testing a calendar type we came across a problem of specific extraordinary train operation days which disturbed the continuous periods when the train did not operate, and thus prevented the calendar from being allocated to otherwise unambigous type. They are e. g. ´holidays´ trains routings which are used as reinforcement before Christmas or Easter holidays. We call these days *isolated operation days* and we identify them as the days before and after which there is a sufficiently long period determined by the algorithm parameters in which the train does not operate. The days are identified and temporarily removed from the bit map even before the determination of the calendar type begins. Their maximal number is one of the algorithm parameters. The isolated days are listed in a special complementary sentence ´the train also operates on ...´at the end of the generated text.

´The train operates in the period from 4 IV to 28 VII on (1), (5), (6), (7). The train also operates on 6 II and 18 IX.´

5. Language usage

The content of the generated texts of course depends on the quality of algorithms which search the relevant calendar type, its regularity and exceptions to it. Last but not least, the language quality is important, too. We have already mentioned some of the principles, now we are going to talk about some language specifics concerning the text description of the train operation days.

First of all, the verb determining the sentence contents is important. It is convenient if the verb is able to help us distinguish between processes regularly repeating themselves and processes that happen once. In Slovak we consider using the verbs to go and to be running (to operate and to be operating) as optimal. We use the first one to describe the regularity, the second one to describe the train rides exceptions. As long as the calendar does not describe the train operation days, we recommend to select a different verb pair. Another important part of sentences is the subject. Hereby, we have been using the word train. It is easy to replace it with a more universal word (line, route) and we have created also the text version with silent subject. These are possible to use universally e. g. to describe the train rides calendar, and, at the same time, direct carriages. We can also talk about the service, its availability level etc.

Another thing we would like to point out is conscious usage of simple, unambigous and clear conjunctions and particles (except, also) the meanings of which users are able to understand and realise very well. In case of using a negative note, we avoid double negatives. Therefore, we do not introduce negative exceptions by using the word ´except´ for the second time but we use the text as follows: *´The train operates daily except (5) and does not operate on 19 V.´*

Generally, we try to generate only simple clauses or simple sentences. In the examples used hereby we have been using slightly longer text variants; it is possible to use more concise and shorter text forms, too.

One needs to realise that language formulation is an extention to the algorithms to search for the calendar type and regularity. It should present and at the same time not depreciate their results, which is a rather difficult but on the other hand manageable task. Modifying the formulations is supposed to enable us to use the algorithm core for the bit calendars text descriptions also from a different area than just the transport one, and also to use the algorithm in various languages mutations.

6. The algorithm implementation

The created algorithm works in accordance with the principles described in the previous chapters. First, isolated operation days are idendified in the bit map. Then they are saved into a separate data structure and removed from the bit map. Next, the bit map is tested for what calendar type it agrees with. Consequently we search the most suitable combination of sample calendars for the convenient types and the periods within which the train operates that result from them. Thus we determine the calendar regularities. On the basis of the found regularity, calendar type and isolated operation days algorithm generates a relevant text.

Implementing the algorithm we also concentrated on another important property. As we have already mentioned, transforming the calendar bit maps into its text representation is a problem which occurs practically in each information system for supporting the basic or operational planning. Thus, when designing the software architecture of the modul implementing this algorithm, we put great emphasis on universality and robustness. Our aim was to use the modul in various systems without the necessity of extensive alterations. The algorithm presupposes the existence of various binary (and, or, xor etc.) as well as unary (not) operators over the bit maps. Furthermore, the calendar bit map interface was defined as a bit values field together with the calednar validity attributes. To use the modul in arbitrary information system, the system just needs to implement the interface. The algorithm approaches the bit map through the interface, and doing that, it does not need to know the implementation details, e. g. the way of the bit map implementation into the system.

To be able to modify the bit map in a simple manner, the modul also contains a bit map editting component. It as well contains the buttons for mass introduction and changes of it departs / does not depart tokens of the whole day groups, such as week

days, month days. Last but not least it contains a place for writing out a bit map text description which is generated immediately with the bit map change. This behaviour is extremely user-friendly.

7. Testing and implementing the algorithm

We did the algorithm pilot testing on a set of calendars of MERITS – the whole-Europe passenger-trains database. Except the entire database we tested separately the calendar sets of the Slovak and Czech Railways trains.

Text generating is controlled by several mentioned parameters. The maximal determined number of exceptions for the testing was 15, and the maximal number of isolated days was 5. The minimal length of period to determine a calendar type was 14 days. In case the algorithm failed the generated text listed all of the calendar valid days in a form of positive exceptions.

The pilot testing brought good results. For the Slovak calendars, in 44 cases we also used the + calendar, and in 89 cases the X calendar. Regarding the similarities of the Slovak and Czech Republics, we used the + and X calendars in approximately identical numbers for the Czech Republic, too. The percentage of calendars which used the isolated days directly increased the algorithm success rate.

By means of setting the parametres in various ways it is possible to increase the algorithm success rate, as well as the quality of generated texts. The relation between the parameters values and the algorithm success rate are to be subject to thorough testing. The simplest way is increasing the number of exceptions but this, together with hightening the success rate, also makes the generated texts longer. Another possibility is to reduce the number of days necessary to accept a period when determining the calendar type. A detail analysis of bit maps which the algorithm has not been successful with will also be important. The analysis can prove it will be necessary to introduce few more calendar types. We suppose that increasing the algorithm success rate will, from a certain level on, require to consider some national specifics.

8. Conclusion and further development

The Slovak Railways use the algorithm basic version to generate text descriptions in the information system to search for an

Calendars of the Slovak Railways trains. 791 calendars altogether, 10 (1.26%) with isolated days.

Calendar type	Not successful	Round-a-year	´operates´ period	two ´operates´ period	´does not operate´ period	´operates daily´ period
Number / %	45 / 5.69	396 / 50.06	219 / 27.69	26 / 3.29	83 / 10.49	22 / 2.78
Aver. number of exceptions	171	5	0	2	4	5
Aver.number of letters in text	245	61	61	92	87	101
Min./max. number of letters	42 / 538	13 / 144	45 / 121	66 / 131	48 / 154	67 / 164

Calendars of the Czech Railways trains. 1352 calendars altogether, 14 (1.04%) with isolated days:

Calendar type	Not successful	Round-a-year	´operates´ period	two ´operates´ period	´does not operate´ period	´operates daily´ period
Number / %	155 / 11.46	631 / 46.67	341 / 25.22	65 / 4.81	96 / 7.10	64 / 4.73
Aver.number of exceptions	183	5	1	2	3	6
Aver.number of letters in text	295	59	68	92	85	116
Min./max. number of letters	45 / 585	13 / 146	45 / 155	65 / 159	48 / 142	63 / 174

Calendars of the complete MERITS database. 21 389 calendars altogether, 642 of them (3.00%) calendars with isolated days:

Calendar type	Not successful	Round-a-year	´operates´ period	two ´operates´ period	´does not operate´ period	´operates daily´ period
Number / %	2282 / 10.67	7157 / 33.46	7240 / 33.85	3606 / 16.86	794 / 3.71	310 /1.45
Aver.number of exceptions	161	6	3	2	5	7
Aver.number of letters in text	223	61	80	92	95	113
Min./max. number of letters	36 / 770	13 / 158	45 / 176	62 / 196	46 / 174	62 / 204

optimal conection. The system called VIS has also been created by our department. The algorithm results have been satisfactory, and having implemented it, we gain priceless notices and impulses.

Now we are preparing new and full-scale version of the algorithm for VIS system. Further on, we suppose the algorithm implementation mainly with the ZONA and SENA information systems for timetable construction, to generate the calendar description texts for passenger timetables.

We are rather surprised that, with respect to importance of the problem, we don´t know about any similar algorithm, result or research. Thus we will welcome any contact and possibility to meet another approach and compare the results.

Acknowledgement

This work was supported by project "Centre of Excellence for Systems and Services of Intelligent Transport" ITMS 26220120028.

References

[1] RUZBARSKY, J.: *Algoritmization of timetable creation,* INFOTRANS 2002, ISBN 80-7194-419-X

[2] GABOR, M.: *Data stream and modification in the timetable construction and other aids of the train traffic diagram,* INFOTRANS 2005, ISBN 80-7194-792-X

[3] SOTEK, K., BACHRATY, H.: *New trends in creation of railway timetable,* Conference EURNEX-ZEL 2008, ISBN 978-80-8070-853-5.

Michal Kohani – Peter Marton *

METHODS AND TECHNIQUES FOR DESIGN OF EFFECTIVE AND COMPETITIVE SINGLE WAGON LOAD TRANSPORTATION

European freight railway operators are losing money with their single wagon load transportation activities. Closing down this business segment does not appear to be a valid option, since single wagon load transportation is connected with full train load business and intermodal traffic. This paper presents methods and techniques that were chosen to solve different problems of single wagon load transportation.

Key words: single wagon load transportation, location-allocation problem, computer simulation

1. Introduction

Single wagon load traffic has been one of the basic offerings of railway companies ever since the railways were created. Over more than a century and a half, this product has undergone several major phases. For almost 100 years there was a constant increase in the volume of wagon loads transported. As road haulage began to grow in the 1960s, the number of single wagon loads carried started to decline. In the 1990s, the predominant view was that single wagon load traffic had no future and would be completely superseded by combined transport. The last few years, however, have seen a turning point, with interest in this service growing again, including among private railway operators [5].

The client will choose single wagon load transportation when he wants to dispatch one or several wagons at the time but does not have enough quantity to fill a full train.

Logistically the system of single wagon load transportation is comparable with a "hub and spoke system" (a system where all goods are brought into a central point – the hub – for sorting and are distributed out from the centre in all directions). It is a network system which consists of customer sidings, stations and marshalling yards [7]:

- If the customer has railway tracks, the operator will send a feeder service to collect the wagons (and give the customer empty wagons to fill). These are then hauled or pulled to a marshalling yard (assembly point for the goods to compromise a wagon load).
- If the client does not have railway track access, he will transport the goods to a terminal by truck where the goods are loaded onto a railway wagon and then brought to the marshalling yard.
- In the marshalling yard further wagons (from other customers) are added and the train is built up for departure to the next hub / marshalling yard in the network. All departures within the network are scheduled and depart at predefined times.
- The wagons are transported from one hub/ marshalling yard to another and wagons are added and taken away at each stop.
- Once the wagon has reached the hub nearest to its destination, it is taken off the train and is transported either by truck or by track to the final destination.

2. Specification of problems to be solved in system of single wagon load transportation

As mentioned above, there are two main processes that exist in the system of single wagon load transportation:

- Wagon transportation among the origin station, marshalling yards and destination station,
- Sorting of wagons in marshalling yards.

The management and control of these processes have significant influence on the client satisfaction and service efficiency. These two processes cannot be organized independently because of their interaction. Different problems can be detected during management and planning of this system, because of complexity of both processes:

- Location of "hubs" in the network,
- Allocation of railway stations (origin or destination) or private sidings of a client to the hubs,
- Specifying a number of trains among marshalling yards (called as direct long-distance trains) and trains that pick up the wagons from origin stations and distribute the wagons to the destination stations (called local freight trains),
- Specifying the composition of direct long-distance trains – one-block trains vs. block trains,
- Specifying routes of local freight trains,
- Specifying the composition of local freight trains,
- Organization of wagon sorting in marshalling yards,
- Developing sorting schemes for secondary sorting in connection to the formation of local freight trains.

* **Michal Kohani, Peter Marton**
Department of Transportation Networks, Faculty of Management Science and Informatics, University of Zilina, Slovakia,
Email: Michal.Kohani@fri.uniza.sk

Different methods of operation research can be used to solve the mentioned problems.

2.1 Location of "hubs" in the network and allocation of origin/destination stations

A single wagon load transportation system is a transportation system, which has approximately the same number of primary sources as number of customers and provides transport of carriages between railway stations. Such a system is called "hub and spoke system". In this case, demands of customers form a matrix of yearly flows of carriages from source railway stations to destination railway stations. We denote this matrix as $B = \{bsj\}$, for $s \in S$ and $j \in J$, where S is a set of source railway stations and J denotes a set of destination railway stations. The fact that the unit cost of transportation is smaller when bigger bulks of items are transported, approves concentration of flows between different pairs of source and customer to stronger flows at least on a part of their way. This flow concentration needs marshalling yards, in which transshipment of transported items is performed and bigger bulks are formed or, on the other hand, where bulks (direct trains) are split into smaller groups (manipulating trains) designated to different destination railway stations.

Contrary to the classical distribution systems, in which big bulks leave a primary source, another situation emerges in this distribution system. Primary sources send relatively small bulks of items and it is useful to concentrate them to bigger bulks in the marshalling yards located near the sources and then to send these bigger bulks to remote marshalling yards and to split them there (see Fig. 1).

We restrict here to the system, in which a railway station is assigned to only one marshalling yard and an exchange of the consignments between this station and other stations is done via the assigned marshalling yard, as it is shown in Fig. 1. Furthermore,

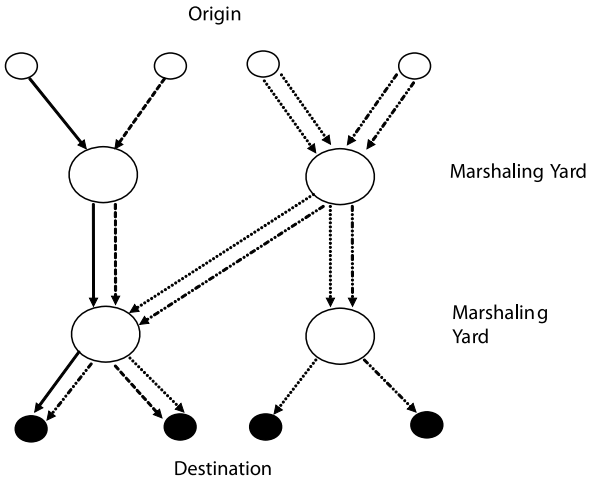


Fig. 1 Scheme of hub and spoke system

we consider the general case, in which any origin station is simultaneously a destination station.

We do not make any difference between an origin station and a destination station hereafter and we introduce in general the set J' of railway stations.

The matrix B gives by coefficients b_{sj} the yearly volume of the carriages, which are sent from the object s to the object j and it gives by coefficients b_{js} the total yearly volume sent from the object j to the object s. In the next section we try to model a symmetrical many-to-many distribution system with a unique assignment of customers to terminals.

Let us consider a case with a linear cost estimation function with the unit cost e_0 for transport of one item along the unit distance on the way from an origin railway station to a marshalling yard or from a marshalling yard to a destination railway station. Let us consider the unit cost e_1 for transport of one item along the unit distance on the way from one to other marshalling yards. Furthermore we denote a set of possible terminal locations by the symbol I, where each place $i \in I$ is associated with the yearly fixed charge f_i for building and performance of a terminal at the location i and with the unit cost g_i for transshipment of one unit in the terminal. In accordance to the previous definition, we denote J' the set of objects which mutually send carriages with the yearly total amounts bsj from $s \in J'$ to $j \in J'$. The symbol dij denotes the distance between the locations i and j. Our objective is to assign each sending or receipting object to exactly one terminal so that the total yearly cost of the designed system is minimal. If we denote by $y_i \in \{0, 1\}$ for $i \in I$ the bivalent variable, which corresponds to the decision if a marshalling yard will or will not be built at the place i and if we introduce the variable $z_{ij} \in \{0, 1\}$ for $i \in I$ a $j \in J'$, which says if the station j will or will not be assigned to the place i, than we can formulate the following mathematical programming model of problem:

Minimise

$$\sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m}\sum_{j=1}^{n} (e_0 d_{ij} + g_i)\left(\sum_{s=1}^{n} b_{js} + \sum_{s=1}^{n} b_{sj}\right) z_{ij} + \\ + \sum_{i=1}^{m}\sum_{k=1}^{m} e_1 d_{ik} \sum_{j=1}^{n}\sum_{s=1}^{n} b_{sj} z_{ij} z_{ks}. \qquad (1)$$

Subject to

$$\sum_{i=1}^{m} z_{ij} = 1 \quad \text{for } j = 1, ..., n, \qquad (2)$$

$$z_{ij} \le y_i \quad \text{for } i = 1, ..., m, j = 1, ..., n, \qquad (3)$$

$$y_i \in \{0, 1\} \quad \text{for } i = 1, ..., m, \qquad (4)$$

$$z_{ij} \in \{0, 1\} \quad \text{for } i = 1, ..., m, j = 1, ..., n. \qquad (5)$$

The model belongs to discrete quadratic programmes due the third term of (1).

2.2 Application of computer simulation

By organization of wagon sorting and development of sorting schemes for secondary sorting the usage of exact mathematical techniques is considerably restricted because of the system complexity and stochastic behavior. On the other hand, the classic planning techniques use very simplified operation models, which do not take into account stochastic behavior of the system and are not able to provide the possibility of observation of independent dynamic technological processes. Average values, standards and expert's knowledge are used instead. An only practicable way how to reach the required level of results plausibility is to use the computer simulation. The computer simulation can be used by checking the ability of a marshalling yard to handle all inbound and outbound trains that were scheduled to terminate or originate there according to the result gathered from solution of previous problems too.

The choice of a suitable simulation tool is very important in this case. By simulation of processes in a marshalling yard it is necessary to meet a certain detail of the model. We recommend the use of microscopic simulation for this case. Thanks to this it will be possible to simulate and consider all the trains, engines and wagons movements, sorting of wagons on hump or classification lead, "on-line" occupation of all the infrastructure parts and work of all personnel. Detailed animation of the processes modeled is very important in case of marshalling yard simulation too.

Another criterion for choice of a simulation tool is its proposition of simulation run evaluation. Many types of railway-specific statistics and protocols are needed by decision-making about ability or non-ability of a marshalling yard to handle the processes planed, e.g. use of switches, filling of classification tracks.

3. Realized parts of research

At present we are working on the solving of location-allocation problem and preparation of simulation models of marshalling yards.

3.1 Tool for decision-making about hubs location

The tool for decision making about hubs location (see Fig. 2) allows the user to compare various possibilities of hubs location, which are based on a different set of input parameters.

This tool works on the network of Slovakian cargo railways. The input parameters, which can't be changed by the user are the distance matrix and the matrix of yearly flows of carriages from source railway stations to destination railway stations. These parameters were obtained from the project [2].

Input parameters which can be changed by the user are fixed cost for each marshaling yard candidate, cost for transshipment of one wagon in terminal, locations of marshaling yards candidates, unit transportation cost e_0 and e_1. These parameters values are pre-set with the data from the project [2], but the user is allowed to change it and obtain various solutions which depend on various possible situations and scenarios. All adjusted parameters can be saved to the text file and can be used for afterwards application.

Having set the parameters the problem can be solved. The solving method based on the approximate linearization of the model (1)-(5) and the *Beta_is* adaptation of the model is used [3]. This solving method can provide the user with the solution in a very short time and the quality of the solution is very high.

Having solved the problem the software allows the user to see the solution on the map, where all the selected marshaling yards are shown. After selecting the marshaling yard from the list or on the map, all the stations allocated to the marshaling yard are shown.

The solution can be saved in the Excel file format and can be used for further application or comparison of various alternatives.

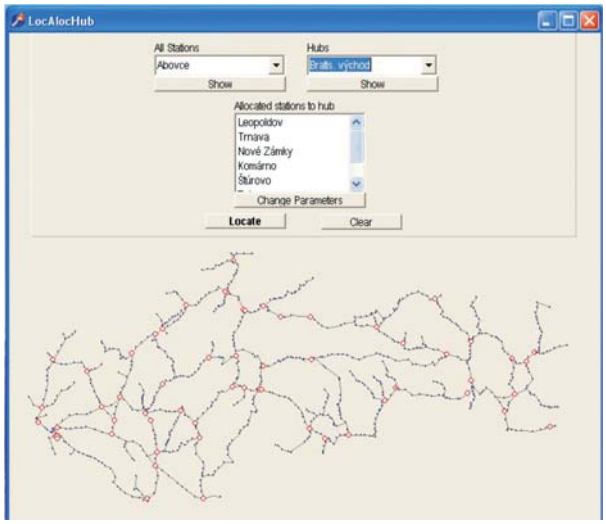


Fig. 2 Screenshot of tool for hubs location

3.2 Application of described methods and techniques – model of single wagon load transportation in Slovakia

Considering the specific requirements described in part 2.2, we recommend to use the Villon simulation tool for this purpose. Up to now the simulation models of many marshalling yards have been built with the help of Villon. We can mention especially the following marshalling yards: Wien Zvbf, Linz Ost Vbf, Hamburg Alte Süderelbe, Basel SBB RB I and Lausanne Triage. Experts from Austria, Germany [6], Switzerland [4] and China have already measured the qualities and properties of Villon. The Villon simulation tool was developed in cooperation of the University of Zilina, Faculty of Management Science and Informatics and SIMCON

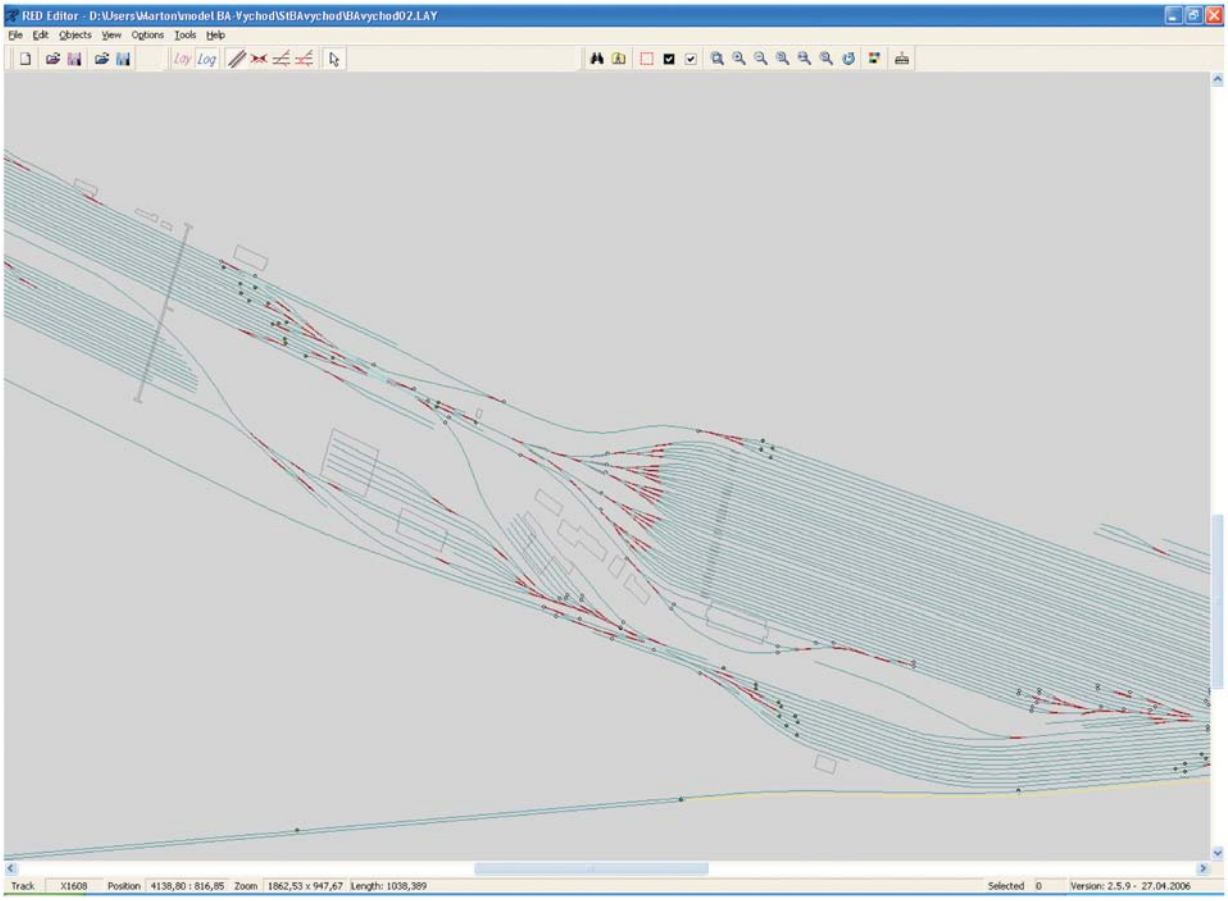


Fig. 3 Detail of infrastructure model of Bratislava vychod marshalling yard

s.r.o. Zilina. More about the properties of Villon simulation tool can be found in [1].

The construction of a simulation model of Bratislava východ marshalling yard has already started. Bratislava vychod marshalling yard is the most important marshalling yard in the Slovak railway network. It is located in Bratislava railway junction, close to Hungarian and Austrian borders. This marshalling yard can be used as a hub for about 12 000 km^2 and about 900 km of lines.

The work on the model development can be divided to several phases:

1. defining the input data – collecting, processing and analyzing the station data,
2. creation of an infrastructure model,
3. creation of a dynamic operation model,
4. experimenting with the simulation model,
5. analysis of the experiment results

Phases 1 and 2 have already been finished. Fig. 3 shows the infrastructure model of Bratislava vychod marshalling yard. Phase 3 has started but the work does not continue at present. Until now all the important data about personnel and engines have been defined in the dynamic simulation model. It is necessary to wait for the results from the design of hubs networks and determination of numbers and routes of freight trains. Thanks to these result we will have important information about freight trains timetable, primary sorting schemes and secondary sorting schemes. Then, phase 3 can continue.

4. Conclusion

Single wagon load transportation is a very flexible system which gives the customer full adaptability in terms of dispatch volatility. Basically the client can choose how many wagons he wants to dispatch. From one day to another the quantity of dispatched wagons can vary. He can decide when to load the wagons, which is a major benefit to the trucks which very often use a time slot loading system with penalties if they cannot dock at the right time. As the routes are fixed in advance, the customer can as soon as he needs to, add wagons to a train.

With an annual freight volume of around 100 billion tkm, single wagon load transportation accounts for approximately 50% of Europe's total rail market. This kind of transport is a crucial supply

chain element for Europe's predominantly midsized and geographically dispersed industry and agriculture [7].

We described the tools that can be used and that are developing to reach acceptable results in field of effective and competitive single wagon load transportation. The software developed for solving the location-allocation problem will be used not only for designing hubs networks and determining numbers and routes of freight trains. We assume that it would be possible to use it for the evaluation of different models of the infrastructure charging system and their influence to the costs and revenues of single wagon load service.

Acknowledgement

This work has been supported by the Slovak Grant Foundation under grant No. 1/4057/07 "Agent oriented models of service systems".

References

[1] ADAMKO, N., MARTON, P.: *Villon – a tool for simulation of operation of transportation terminals,* Communications – Scientific Letters of the University of Zilina, Vol. 10, 2/2008, p. 10–14, ISSN 1335-4205

[2] JANACEK, J. et all: *Report of solving of initial study of 5.3 task "Software solution for optimization of transport work control" (in Slovak)*, Fakulta riadenia a informatiky, Zilinska univerzita, Zilina, 2001

[3] KOHANI, M.: *Approximative methods for "many-to-many" distribution system design,* Quantitative Methods in Economics (Multiple criteria decision making XIII): Proc. of the 13th International Conference: Bratislava, University of Economics, ISBN 80-8078-129-X. - P. 77–83, 2006

[4] KONIG, H.: *VirtuOS – Simulieren von Bahnbetrieb* Eisenbahntechnische Rundschau Nr. 1/2, Hestra-Verlag, Hamburg, 2001.

[5] MARTON, P.: *Einzelwagenverkehr – gegenwärtige Situation in Europa,* EI – Der Eisenbahningenieur. Jahrg. 59, 2008, p. 43–47. ISSN 0013-2810

[6] TALKE, W.: *Effizienzsteigerung in Zugbildungsanlagen.* Eisenbahntechnische Rundschau, Nr. 11 & 12, Hestra-Verlag, Hamburg, 1998.

[7] UIC – Rail Freight Portal – http://www.uic.asso.fr/uic/spip.php?rubrique1166

COMMUNICATIONS – Scientific Letters of the University of Zilina
Writer's Guidelines

1. Submissions for publication must be unpublished and not be a multiple submission.
2. Manuscripts written **in English language** must include **abstract** also written in English. The submission should not exceed **10 pages** with figures and tables (format A4, Times Roman size 12). The **abstract** should not exceed 10 lines.
3. Submissions should be sent: **by e-mail** (as attachment in application MS WORD) to one of the following addresses: *komunikacie@uniza.sk* or *holesa@uniza.sk* or *vrablova@uniza.sk* or *polednak@fsi.uniza.sk* **with a hard copy** (to be assessed by the editorial board) **or on a CD** with a hard copy to the following address: Zilinska univerzita, OVaV, Univerzitná 1, SK-010 26 Zilina, Slovakia.
4. Abbreviations, which are not common, must be used in full when mentioned for the first time.
5. Figures, graphs and diagrams, if not processed by Microsoft WORD, must be sent in electronic form (as GIF, JPG, TIFF, BMP files) or drawn in contrast on white paper, one copy enclosed. Photographs for publication must be either contrastive or on a slide.
6. References are to be marked either in the text or as footnotes numbered respectively. Numbers must be in square brackets. The list of references should follow the paper (according to **ISO 690**).
7. The author's exact **mailing address of the organisation where the author works, full names, e-mail address or fax or telephone number,** must be enclosed.
8. The editorial board will assess the submission in its following session. In the case that the article is accepted for future volumes, the board submits the manuscript to the editors for review and language correction. After reviewing and incorporating the editor's remarks, the final draft (before printing) will be sent to authors for final review and adjustment.
9. The deadlines for submissions are as follows: September 30, December 31, March 31 and June 30.

COMMUNICATIONS

SCIENTIFIC LETTERS OF THE UNIVERSITY OF ZILINA
VOLUME 11

Address of the editorial office:
Zilinská univerzita
Office for Science and Research
(OVaV)
Univerzitna 1
SK 010 26 Zilina
Slovakia
E-mail: komunikacie@nic.uniza.sk,
polednak@fsi.uniza.sk

Each paper was reviewed by two reviewers.

Journal is excerpted in Compendex and Scopus

It is published by the University of Zilina in
EDIS – Publishing Institution of Zilina University
Registered No: EV 3672/09
ISSN 1335-4205

Published quarterly

Single issues of the journal can be found on:
http://www.uniza.sk/komunikacie