JYVÄSKYLÄ STUDIES IN COMPUTING
92

Anne Setämaa-Kärkkäinen

Network Connection Selection-Solving a New Multiobjective Optimization Problem

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 92

Anne Setämaa-Kärkkäinen

Network Connection Selection – Solving a New Multiobjective Optimization Problem

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksessa (Ag Aud. 2)
elokuun 23. päivänä 2008 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in the Building Agora (Ag Aud. 2), on August 23, 2008 at 12 o'clock noon.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2008

Network Connection Selection – Solving a New Multiobjective Optimization Problem

JYVÄSKYLÄ STUDIES IN COMPUTING 92

Anne Setämaa-Kärkkäinen

Network Connection Selection – Solving a New Multiobjective Optimization Problem

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2008

Editors
Timo Männikkö
Department of Mathematical Information Technology, University of Jyväskylä
Pekka Olsbo, Marja-Leena Tynkkynen
Publishing Unit, University Library of Jyväskylä

URN:ISBN:978-951-39-8047-4
ISBN 978-951-39-8047-4 (PDF)
ISSN 1456-5390

ISBN 978-951-39-3301-2
ISSN 1456-5390

Jyväskylä University Printing House, Jyväskylä 2008

ABSTRACT

Setämaa-Kärkkäinen, Anne
Network Connection Selection - Solving a New Multiobjective Optimization Problem
Jyväskylä: University of Jyväskylä, 2008, 52 p.(+included articles)
(Jyväskylä Studies in Computing
ISSN 1456-5390; 92)
ISBN 978-951-39-3301-2
Finnish summary
Diss.

Mobile terminals are nowadays able to use several different kinds of wireless network connections to transfer data. The users of mobile terminals can select the network connection that is the most suitable for their purposes. In this thesis, we consider a more advanced case in which a mobile terminal can use several different connections simultaneously to transfer data. When the data is composed of separate components, the components can be transferred using different connections. Then the user needs to make a selection which network connection to use for each component. This kind of selection may be very difficult to make and most of the users are not interested in making such selections. Therefore, an automatic selection method is needed.

The network connection selection is modelled as a multiobjective optimization problem in which the aim is to minimize the time used in transferring and the costs of the transfer. To solve the problem, we have developed a fast heuristic solution method that uses only little computational capacity so that it is possible to implement it in a mobile terminal. In addition, the method takes into account possibly changing network environment during the transmission and improves, when possible, the transmission schedule formed by selecting a network connection for each component of the data.

Keywords: Multiobjective optimization, Combinatorial optimization, Heuristics, Parallel machines scheduling, Telecommunications, Wireless networks

Author

Anne Setämaa-Kärkkäinen
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Supervisors

Professor Kaisa Miettinen
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Doctor Jarkko Vuori
EVTEK University of Applied Sciences
Finland

Reviewers

Associate Professor Vincent T'kindt
Laboratory of Computer Science
University of Tours
France

Professor G. Anandalingam
University of Maryland
United States

Opponent

Professor Xavier Gandibleux
University of Nantes
France

ACKNOWLEDGEMENTS

First, I want to thank my supervisors Doctor Jarkko Vuori for introducing this research topic to me and Professor Kaisa Miettinen for many fruitful discussions and guidance during this work. Doctor Jani Kurhinen was of valuable assistance with issues related to wireless networks, and Professor Marko M. Mäkelä shared his opinions in the beginning of this research. I also want to thank Associate Professor Vincent T'kindt and Professor G. Anandalingam for reviewing this thesis.

For arranging practical things, especially during my visit in Lund, Sweden, I thank Marja Leppäharju from the GETA graduate school and Päivi Jämsen from the Department of Mathematical Information Technology at the University of Jyväskylä.

I wish to thank everybody with whom I have worked during this research. I have had a priviledge to get to know many delightful people, of which I would especially like to thank Katja Kaario and Sanna Mönkölä. Working side by side with them has always been a pleasure.

I thank my parents for everything they have done for me as well as my parents-in-law. My brother Sami I want to thank for being my personal computer support as well as encouraging me. Finally, I thank my husband Kimmo for his patience, love and support and our beautiful children Eeva and Miika for teaching me what is really important in life.

This work was financed by the GETA graduate school and the Agora Center. Financial support was also provided by Emil Aaltonen foundation, Tekniikan edistämissäätiö foundation and Ellen and Artturi Nyyssönen foundation.

Jyväskylä, June 2008
Anne Setämaa-Kärkkäinen

LIST OF FIGURES

CONTENTS

LIST OF INCLUDED ARTICLES

PI Anne Setämaa-Kärkkäinen, Kaisa Miettinen and Jarkko Vuori. Best Compromise Solution for a New Multiobjective Scheduling Problem. *Computers & Operations Research 33(8): 2353–2368*, 2006.

PII Anne Setämaa-Kärkkäinen, Kaisa Miettinen and Jarkko Vuori. Heuristic for a New Multiobjective Scheduling Problem. *Optimization Letters 1(3): 213–225*, 2007.

PIII Anne Setämaa-Kärkkäinen and Jani Kurhinen. Optimal Usage of Multiple Network Connections. *In Proceedings of the 1st International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications, MOBILWARE 2008, ed. by Jiang (Linda) Xie and Tuna Tugcu*, 2008.

PIV Anne Setämaa-Kärkkäinen. Heuristic for Selecting Network Connections in a Dynamic Environment. *Reports of the Department of Mathematical Information Technology, Series B. Scientific Computing, No. B 9/2008*, 2008.

The articles **PI**-**PIII** are mainly written by the author. The multiobjective optimization methods tested for solving the problem in the article **PI** were proposed by Professor Kaisa Miettinen. The heuristics presented in the articles **PII**-**PIV** are based on the author's ideas. The author has implemented the heuristics and run all the computational tests during this research. The simulator used in the article **PIV** has also been implemented by the author. The pricing models used in the test instances were developed by Doctor Jarkko Vuori, who also provided the network connections used in the test instances. Valuable advice (considering especially multiobjective optimization) have been given by Professor Kaisa Miettinen during this research. Doctor Jarkko Vuori and Doctor Jani Kurhinen have been consulted for issues concerning telecommunications.

1 INTRODUCTION

Mobile terminals are nowadays able to use different kinds of wireless network connections. In the future, fourth generation (4G) networks enable global roaming across many wireless networks [34]. This means that the user of a mobile terminal can choose which network connection to use and change the connection when needed. This possibility is called the concept of being always best connected (ABC) [11]. The ABC concept has been under a lot of study recently. For example, in [8] a generic model of the ABC concept is introduced and in [39] an architecture supporting the ABC concept is presented for the 4G networks. The architecture presented includes also an optimization method for selecting a network connection to be used among all the connections available.

In this thesis, we take a step further and consider a case in which mobile terminals are able to use several different kinds of network connections simultaneously. Then the selection which network connection to use for transferring data is extended to which network connection to use for transferring each separate component of the data. In other words, the selection has to be made for each data component that can be transferred independently of the other components. Using several network connections simultaneously to transfer data makes the transmission faster. In addition, when the users can select connections from a wider range of different connections and change the connections when wanted, the operators of networks face a more rigorous competition which can lower the prices of using the connections.

Not only the users of mobile terminals but also service operators can encounter this problem in the future. *Service operators* do not own networks but use networks of other operators. Currently, service operators use networks mainly from one operator but in the future a service operator may have contracts with several operators. Then the customers of the service operator will have a wider range of networks available for data transmission and some kind of a selection method is needed to decide which network connections to use for each data transmission.

We call this problem of selecting a network connection for each data component the *network connection selection problem* (NCS). We have modelled the prob-

lem as a multiobjective optimization problem in which the objective is to minimize both the time used in transferring the data and the costs of the transfer. The NCS problem has not been studied earlier and we are not aware of any studies on similar problems of which solution methods could be applied to the NCS problem. Thus, we have started our research by modelling the problem, then we have studied different existing methods for solving the problem, and finally we have developed a new solution method. Solving the NCS problem is not straightforward since we have two conflicting objectives: Instead of an optimal solution we have several mathematically equivalent solutions among which we need to find the best compromise solution in which the time used in transferring is small and the costs are reasonable.

Since the users of mobile terminals are rarely interested in this kind of selection problems, the aim of this research is to develop an automatic solution method that is transparent to the user of a mobile terminal. In addition to that, the solution method should be fast and it should be possible to implement it in mobile terminals. This means that the solution method should use as little computational capacity as possible since the capacity available in mobile terminals is very limited. Since wireless networks are dynamic, the method should also take into account possibly changing network conditions during the transfer of the data. In other words, if there occur changes in the network conditions that make the current solution infeasible or poor in some other way, the solution should be updated in order to finish the transfer in a short time and with reasonable costs.

As already mentioned, we have modelled the NCS problem as a multiobjective optimization problem. To be more precise, the problem has been formulated as a biobjective integer programming problem. The mathematical model is presented in Chapter 2. Before we consider solving the NCS problem, we introduce in Chapter 3 some concepts of multiobjective optimization and briefly present multiobjective optimization methods that have been studied for solving the NCS problem. Since the NCS problem can be considered as a multiobjective scheduling problem, we also define some scheduling terminology in Chapter 4. In Chapter 5, we discuss solving the NCS problem. In Chapter 6, we consider how to deal with dynamically changing network conditions and discuss test instances used in the computational tests. In this research, we have assumed that the components of the data to be transferred are independent of each other, that is, it is possible to transfer them in any order. This kind of an assumption is not always valid and, therefore, in Chapter 7 we discuss a case of the NCS problem in which there are precedence constraints defining that some components have to be transferred before some other components. Finally, in Chapter 8 we present conclusions and define future research plans.

The main results of this research have been presented in the four included articles. The articles, as well as the whole process of solving the NCS problem, follow the three phases introduced in [31] for solving a general multiobjective scheduling problem. Each phase consists of solving a subproblem. The subproblems are modelling the problem, taking into account the objectives and scheduling. In the first phase, modelling the problem, the objectives and the constraints

of the problem are formulated. The second phase of taking into account the objectives consists of selecting a multiobjective optimization method for solving the problem. In the last phase the scheduling problem defined by the first two phases is solved. Thus, the actual scheduling is done in the final phase. In the first article **PI**, we have modelled the NCS problem and decided how the objectives will be taken into account. For that purpose, we have studied and compared different multiobjective optimization methods. In the second article **PII**, we have considered solving the actual scheduling problem obtained in **PI**, and we have presented a heuristic that solves the problem fast. In the first two articles, which consider the NCS problem from the optimization point of view, we have assumed a static environment in which there are no changes in the network environment during the transfer. This assumption was made because the NCS problem is very challenging with a lot of restrictions on the solution method (it should be automatic, fast and use only little computational capacity). However, the assumption is valid only when the transfer time is very short and, therefore, we have considered a more realistic dynamic network environment in **PIII** and **PIV**. In **PIII**, the NCS problem is considered from the telecommunications point of view, and a method for dealing with the possibly changing network conditions is proposed. The method is a modification of the heuristic presented in **PII**, and we call it the dynamic heuristic. The dynamic heuristic is presented in more detail in **PIV**, in which results of simulations used to test different versions of the dynamic heuristic are also shown.

2 PROBLEM FORMULATION

The network connection selection problem consists of deciding which network connections to use for transferring data. The data to be transferred is composed of a set of components that are assumed to be independent of each other. Then the components can be transferred separately using different network connections available. The network connections that are available may have very diverse properties. We assume that some kind of selection is made in advance so that the network connections we consider have the properties required for the transmission. The required properties depend on the mobile terminal in question as well as on the data to be transferred. (This kind of a selection can be done, for example, using a method similar to the one presented in [1].) Thus, we consider only connections that are such that it is possible to transfer the data using any of them.

The selection which network connection to use for each component is based on the transmission rates and prices of the available connections. In other words, we want to minimize both the time needed for transferring and the costs of the transfer. In order to solve the NCS problem, we need the following information: the rate and price of each connection and the sizes of the data components. It should be noted that we consider a general case in which the sizes of the components are not necessarily equal, and the components considered in this research are not packets that are used when transferring digital data but larger entities that together form the data. For example, a component can be an image or a small text part of a web page that is to be transferred.

The sizes of the components are easily obtained when the data to be transferred is known. The rates and especially the prices are not that easy to obtain. We assume that the operators of the network connections provide this information in some form to the customers. How the information on the rates and prices can be obtained and in what kind of a form it is given is out of the scope of this thesis. We only assume that this information is given in such a way that the costs as well as the time used in transferring can be calculated for each component. These costs and time used when solving the NCS problem can be estimates as long as the estimates are similarly formed for each network connection.

It should be noted that we do not define the actual routing of the data components in the networks but select a network for each data component to be used for transferring the component. Inside each network, routing algorithms handle the actual physical routing of the data components from the source to the destination. In other words, the NCS problem is an assignment problem, not a routing problem. We next formulate the problem mathematically.

2.1 Mathematical model

Let us assume that there are n components to be transferred using m network connections. The rates and prices of the network connections as well as the sizes of the components are known. Let x_{ij} be a binary variable such that $x_{ij} = 1$, when component j is transferred using connection i, otherwise $x_{ij} = 0$, where $i = 1, \ldots, m$, $j = 1, \ldots, n$. The time needed for transferring component j using connection i is denoted by d_{ij} and the costs of that transfer are denoted by c_{ij}. The network connection selection problem can now be stated as a biobjective integer programming problem:

$$\begin{aligned}
\text{minimize} \quad & f_1(x) = \max_{i=1,\ldots,m} \sum_{j=1}^{n} d_{ij} x_{ij} \quad \text{and} \quad f_2(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{i=1}^{m} x_{ij} = 1, \quad \text{for all} \quad j = 1, \ldots, n, \\
& x_{ij} \in \{0, 1\}, \quad \text{for all} \quad i = 1, \ldots, m \quad \text{and} \quad j = 1, \ldots, n.
\end{aligned}$$

The objective function f_1 presents the time when all the components have been transferred and the objective function f_2 presents the total costs of transferring the components. The constraints make sure that each component is transferred using exactly one connection.

The objective function f_1 is nonlinear but we can reformulate the problem to avoid that. We use an additional variable y which denotes the time when all the components are transferred. To define the time, we add m new constraints:

$$\sum_{j=1}^{n} d_{ij} x_{ij} \leq y, \quad \text{for all} \quad i = 1, \ldots, m,$$

and define the first objective function (to be minimized) to be y. Thus, an equivalent linear model of the network connection selection problem is

$$
\begin{aligned}
\text{minimize} \quad & f_1(x) = y \quad \text{and} \quad f_2(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{i=1}^{m} x_{ij} = 1, \quad \text{for all} \quad j = 1, \ldots, n, \\
& \sum_{j=1}^{n} d_{ij} x_{ij} \leq y, \quad \text{for all} \quad i = 1, \ldots, m, \\
& x_{ij} \in \{0, 1\}, \quad \text{for all} \quad i = 1, \ldots, m \quad \text{and} \quad j = 1, \ldots, n, \\
& y \geq 0.
\end{aligned}
$$

The solution to this problem tells which network connection is used for transferring each component. It does not consider the order in which the components are transferred on each network connection because neither the time used in transfers nor the total costs depend on the order of the components. Therefore, the components can be ordered on each connection after the model is solved. This simplifies the model and makes it faster to solve it. It should however be noted that the order of the components is needed when a transmission schedule is formed. A *transmission schedule* tells when each component is transferred on each connection. The transmission schedule is used when the components are transferred. Note also that here we assume that the components are independent of each other, that is, it is possible to transfer the components in any order. This is not always possible, and the case in which the components are not independent but there are precedence relations between some components is considered in Chapter 7.

3 MULTIOBJECTIVE OPTIMIZATION

We have modelled the NCS problem as a multiobjective optimization problem. Solving an optimization problem with multiple objectives is not as straightforward as solving a single objective problem. Therefore, we next briefly explain some concept of multiobjective optimization following the terminology used in [20]. Since the NCS problem contains only two objective functions we consider only biobjective problems in this section.

3.1 Concepts

Let us consider a general biobjective optimization problem:

$$\begin{array}{ll} \text{minimize} & f_1(x) \quad \text{and} \quad f_2(x) \\ \text{subject to} & x \in S. \end{array}$$

The vector of decision variables $x = (x_1, x_2, \ldots, x_n)^T$ is called a *decision vector* and the vector of objective functions $F(x) = (f_1(x), f_2(x))^T$ is called an *objective vector*. The set $S \subset \mathbf{R}$ is the *feasible region* that is formed by constraint functions. The image of the feasible region $Z = F(S)$ is called the feasible objective region. We want to minimize both objective functions f_1 and f_2 simultaneously. Usually, it is not possible to find a solution in which both of the objective functions attain minimal values. In addition, we cannot compare all the solutions mathematically and, therefore, we cannot find an optimal solution in a similar way as in single objective optimization. In multiobjective optimization, a concept called Pareto optimality is used instead of optimality. A decision vector $x^\star \in S$ and the corresponding objective vector $F(x^\star)$ are *Pareto optimal* if there does not exist another decision vector $x \in S$ such that $f_i(x) \leq f_i(x^\star)$ for $i = 1, 2$ and $f_j(x) < f_j(x^\star)$ for at least one j. Similarly, a decision vector $x^\star \in S$ and the corresponding objective vector $F(x^\star)$ are *weakly Pareto optimal* if there does not exist another decision vector $x \in S$ such that $f_i(x) < f_i(x^\star)$ for $i = 1, 2$. A *properly Pareto optimal solution* is a Pareto optimal solution with bounded trade-offs.

All the Pareto optimal solutions are mathematically equivalent. Which of the Pareto optimal solutions is selected as the final solution to the problem depends usually on a decision maker. A *decision maker* is a person who knows the problem well and can express preferences between the objectives. The decision maker can, for example, give aspiration levels for the objectives. The *aspiration levels* are objective function values that are satisfying to the decision maker. A vector containing the aspiration levels is called a *reference point*. It should be noted that it is not always possible to use a decision maker in which case the selection of the final solution has to be based on other information on the problem.

It is useful to know the ranges of the objective functions in the Pareto optimal set. The best, that is, the minimal values of the objective functions in the Pareto optimal set are contained in an *ideal objective vector* z^*. The ideal objective vector can be calculated as follows: Component z_i^*, $i = 1, 2$, is obtained by minimizing the ith objective function subject to the constraint $x \in S$. The ideal objective vector is typically infeasible. Otherwise, it would be the optimal solution to the problem and the objectives would not be conflicting, which is not usually the case. Another infeasible objective vector used in some multiobjective optimization methods is called a *utopian objective vector* z^{**}. It can be formed from the ideal objective vector by subtracting a relatively small scalar $\epsilon_i > 0$ from each component z_i^* of the ideal objective vector. Thus, we have $z_i^{**} = z_i^* - \epsilon_i$ for $i = 1, 2$.

The worst, that is, the largest values of the objective functions in the Pareto optimal set are collected into a *nadir objective vector* z^{nad}. In the case of two objective functions, the components of the nadir objective vector can be calculated at the same time the ideal objective vector is formed [7]: The first component is the value of the first objective function $f_1(x_2)$ in the point x_2 in which the second objective function attains its minimal value. Similarly, the second component is the value of the second objective function $f_2(x_1)$ in the point x_1 in which the first objective function attains its minimal value. It should be noted that when there are more than two objective functions, the nadir objective vector can only be estimated and the estimate can be poor, see for example [35].

3.2 Methods

Multiobjective optimization problems are usually solved by scalarization [20, 29]. *Scalarization* means converting the problem with multiple objectives into a single objective optimization problem or a family of such problems. The objective function of the single objective problem is called a *scalarizing function*. The scalarized problem can be solved using methods developed for single objective optimization.

Multiobjective optimization methods can be divided into four classes according to the role of the decision maker [20]. In no-preference methods, the decision maker is not involved in the solution procedure and no preference in-

formation is available. In a priori methods, the decision maker gives preference information before the solution procedure. In a posteriori methods, the Pareto optimal set, or a part of it, is generated and the decision maker selects the final solution from the set. In interactive methods, the decision maker is involved in an iterative solution process. The decision maker can express preferences and guide the search for the most preferred solution while different solutions are formed and shown to the decision maker.

The aim of this research is to develop a fast automatic solution method to solve the NCS problem. Therefore, we cannot ask the user of a mobile terminal to act as a decision maker in the solution procedure and, thus, we have concentrated on the no-preference methods of multiobjective optimization.

Because we do not have a decision maker, we need to know the nature of the problem before we can select a method that produces a good compromise between the conflicting objectives. For that reason we have started our research by using a posteriori methods to form approximations of Pareto optimal sets of some problem instances [25]. Then, we studied different scalarizing functions in order to find a method for obtaining a good compromise between the conflicting objectives. We used the method of weighted metrics with three different metrics to form approximations of Pareto optimal sets in [25]. A method producing a neutral compromise solution and an achievement scalarizing function were then considered as the final method for dealing with the multiple objectives. The main results are presented in **PI** whereas in [25] a larger description of the results is presented. We next briefly present the above mentioned methods.

Method of weighted metrics

The method of weighted metrics produces Pareto optimal solutions by minimizing the distance between some reference point and the feasible objective region [20]. We have used the ideal objective vector z^* as the reference point, and the distance has been measured using a weighted L_p-metric with values 1, 2, and ∞ for p. For $1 \leq p < \infty$, the *weighted L_p-problem* to be minimized in the method is

$$\begin{aligned} &\text{minimize} && \left(\sum_{i=1}^{2} w_i |f_i(x) - z_i^*|^p \right)^{1/p} \\ &\text{subject to} && x \in S, \end{aligned} \tag{1}$$

and for $p = \infty$, that is, the Tchebycheff metric, the *weighted Tchebycheff problem* is

$$\begin{aligned} &\text{minimize} && \max_{i=1,2} \{ w_i |f_i(x) - z_i^*| \} \\ &\text{subject to} && x \in S. \end{aligned} \tag{2}$$

The weighting coefficients w_1 and w_2 must be greater than or equal to zero and sum up to one, that is, $w_1 + w_2 = 1$. If the real ideal objective vector is known, the absolute value signs can be left out. In addition, the exponent $1/p$ can be dropped in problem (1).

By altering the weighting coefficients w_1 and w_2 we get different solutions. The solution to the weighted L_p-problem (1) is Pareto optimal if either the solution is unique or all the weighting coefficients are positive [20]. The solution to the weighted Tchebycheff problem (2), on the other hand, is weakly Pareto optimal if all the weighting coefficients are positive [20].

When the weighted Tchebycheff metric is used with a utopian objective vector z^{**} as the reference point, the method of weighted metrics can find any Pareto optimal solution. Unfortunately, also weakly Pareto optimal solutions may be produced. This can be avoided by using an augmented weighted Tchebycheff metric to measure the distance between the utopian objective vector and the feasible objective region. The *augmented weighted Tchebycheff problem* to be minimized is

$$\begin{array}{ll} \text{minimize} & \max\limits_{i=1,2}\{w_i|f_i(x) - z_i^{**}|\} + \rho \sum\limits_{i=1}^{2} w_i|f_i(x) - z_i^{**}| \\ \text{subject to} & x \in S. \end{array} \tag{3}$$

where the augmentation coefficient $\rho > 0$ is a sufficiently small scalar. In (3), the weighted Tchebycheff problem (2) is modified by adding an augmentation term to the scalarizing function.

Neutral compromise solution

A method for generating a so-called neutral compromise solution is described in [38]. A *neutral compromise solution* is a solution that is located somewhere in the middle of the Pareto optimal set. It can be a good candidate for the final solution to a multiobjective optimization problem when there are no preferences between the conflicting objectives available. A neutral compromise solution is obtained by solving the problem

$$\begin{array}{ll} \text{minimize} & \max\limits_{i=1,2} \dfrac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^{**}} + \rho \sum\limits_{i=1}^{2} \dfrac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^{**}} \\ \text{subject to} & x \in S, \end{array} \tag{4}$$

where the reference point z^{mid} is located in the middle of the ranges of the objective functions in the Pareto optimal set, that is,

$$z_i^{mid} = \frac{z_i^{nad} + z_i^{*}}{2}, \; i = 1, 2 \tag{5}$$

and the augmentation coefficient ρ is a small positive scalar.

The problem is a modification of the augmented weighted Tchebycheff problem (3), and its idea resembles the idea of problem (3). However, the reference point is now different and the distance measure is more general guaranteeing that the solution is Pareto optimal regardless of the location of the reference point [20, 38]. The Pareto optimality is guaranteed by the augmentation term. Note also that the objective functions are scaled to be of comparable magnitude in (4).

Achievement scalarizing function

The following achievement scalarizing function to be minimized [36, 37] is a modification of the method that produces a neutral compromise solution [38]:

$$\begin{array}{ll} \text{minimize} & \max\limits_{i=1,2} w_i \dfrac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^{**}} + \rho \sum\limits_{i=1}^{2} w_i \dfrac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^{**}} \\ \text{subject to} & x \in S. \end{array} \tag{6}$$

The reference point is the same middle point (5) as in the method producing a neutral compromise solution, and ρ is again a small positive scalar. The ratio of the positive weighting coefficients w_1 and w_2 represents the rate at which the user is willing to trade off values of the objective functions. It should again be noted that the objective functions are scaled to be of comparable magnitude in (6). The solution to the achievement scalarizing function (6) is Pareto optimal since the scalarizing function is strongly increasing [6], and the method can find any properly Pareto optimal solution [20].

We next present a proof of Pareto optimality for the solution to the achievement scalarizing function (6). The proof is presented here because the achievement scalarizing function (6) was found suitable for solving the NCS problem in **PI** and it is used also in the heuristic solution method developed for solving the problem in **PII**.

Proposition. The solution to the achievement scalarizing function (6) is Pareto optimal.

Proof. Let $x^s \in S$ be a solution to the achievement scalarizing function (6). Let us suppose that x^s is not Pareto optimal. Then, there exists another solution $x \in S$ such that $f_i(x) \leq f_i(x^s)$ for $i = 1, 2$ and $f_j(x) < f_j(x^s)$ for at least one j, $j = 1, 2$. This means that

$$\frac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^{**}} \leq \frac{f_i(x^s) - z_i^{mid}}{z_i^{nad} - z_i^{**}}$$

for both $i = 1, 2$ and

$$\frac{f_j(x) - z_j^{mid}}{z_j^{nad} - z_j^{**}} < \frac{f_j(x^s) - z_j^{mid}}{z_j^{nad} - z_j^{**}}$$

for at least one j, $j = 1, 2$. Since we have $w_i > 0$ for both $i = 1, 2$ and $\rho > 0$, the following inequalities are valid:

$$\max_{i=1,2} w_i \frac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^{**}} \leq \max_{i=1,2} w_i \frac{f_i(x^s) - z_i^{mid}}{z_i^{nad} - z_i^{**}}$$

and

$$\rho \sum_{i=1}^{2} w_i \frac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^{**}} < \rho \sum_{i=1}^{2} w_i \frac{f_i(x^s) - z_i^{mid}}{z_i^{nad} - z_i^{**}}.$$

This implies that

$$\max_{i=1,2} w_i \frac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^{**}} + \rho \sum_{i=1}^{2} w_i \frac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^{**}} <$$

$$\max_{i=1,2} w_i \frac{f_i(x^s) - z_i^{mid}}{z_i^{nad} - z_i^{**}} + \rho \sum_{i=1}^{2} w_i \frac{f_i(x^s) - z_i^{mid}}{z_i^{nad} - z_i^{**}}.$$

Thus, x^s cannot be a solution to the achievement scalarizing function (6). □

4 SCHEDULING

In this chapter, we discuss scheduling problems since the NCS problem can be seen as a scheduling problem. Scheduling is a general term used to refer to many different kinds of problems in which the objective is to allocate resources to tasks over a time period [2, 18]. We consider here machine scheduling problems in which the common feature is that a set of jobs is to be assigned to a set of machines for processing. A solution to a machine scheduling problem is a schedule telling when a job is processed and on which machine. A job consists of one or several operations which may have to be processed in a certain order and on a certain machine. Processing of an operation consumes resources and uses a certain amount of time on a machine. The objective in scheduling problems is usually related to time but also the use of resources can be optimized. In multiobjective scheduling problems, there are multiple objectives conserning time and/or resources.

In order to classify scheduling problems, a three field notation $\alpha|\beta|\gamma$ introduced in [9] is used. The first field α denotes the scheduling environment, the field β contains information about the constraints of the problem and the field γ describes the objective of the problem. The notation was originally developed for single objective optimization problems and it is extended for multiobjective problems in [30] and [31]. We next formulate the NCS problem as a scheduling problem and then we discuss scheduling methods that could be applied to the NCS problem.

4.1 NCS as a scheduling problem

The NCS problem can be seen as a scheduling problem in which machines represent network connections and each component forms a job that has to be processed on one of the machines. The machines are in this case *uniform parallel machines*, which means that the machines are identical, that is, they perform the same operation, but they have different speeds that do not depend on the job.

(Thus, the network connections have different rates, and the rate of a connection is not dependent on the component that is transferred using it.) Each job is to be processed on one machine, and the goal is to minimize the total costs of processing the jobs and the *makespan*, which is the time when all the jobs are processed. Using the three field notation described above, the NCS problem can be stated as $Q||C_{max}, \sum f_j$, where Q denotes uniform parallel machines, C_{max} is the makespan and f_j denotes the cost of processing a job j. The above mentioned extended three field notation for multiobjective scheduling problems can contain information about the multiobjective optimization method used for solving the problem. For example, $Q||F_l(C_{max}, \sum f_j)$ means that a linear convex combination of the objectives (the makespan and the costs) is minimized. We have not specified any method in the notation, since our aim has been to study different methods for solving the problem.

The NCS problem has not been studied before in the literature, and also the scheduling problem $Q||C_{max}, \sum f_j$ has not attracted much attention. The reason for that is most likely that the scheduling problems usually represent actual problems that arise in the field of production and manufacturing. In that field, there are usually no costs to be paid for using the machines after the machines are bought and the objectives of scheduling are related to the time used in processing and inventory costs. The NCS problem comes from a very different kind of an environment in which, in addition to other differences, the solution must be obtained very fast and without using much computational capacity. This makes it difficult to apply existing scheduling methods to the NCS problem.

The scheduling problem $Q||C_{max}$ is NP-hard since the problem $P||C_{max}$ is NP-hard [31]. This means that the problem $Q||C_{max}, \sum f_j$ and, thus, the NCS problem is NP-hard. When a problem is NP-hard, it is unlikely that there exists a polynomial-time algorithm to solve the problem. An algorithm is called *polynomial-time* when its complexity is polynomial in the size of the input. The *complexity* of an algorithm is the worst-case behaviour of the algorithm on any input [23], the input in the case of optimizing is an optimization problem to be solved, and the size of the input is the size of the problem. For example, in the case of the NCS problem, the size is defined by the number of components and the number of network connections.

NP-hardness of a problem means that it is time-consuming to solve large problem instances using exact methods. For some problems, it is very time-consuming to solve even moderate size problem instances. Since the exact optimal solution is not always needed and, in many cases, there is not much time available to be used in solving the problem, many heuristics have been developed for NP-hard problems. A *heuristic* solves a problem approximately without a guarantee that the solution is optimal, or even near-optimal. However, in many cases in which finding the exact optimal solution is not necessary heuristics perform well. Another reason for using heuristics, especially in the case of scheduling, is that it can be hard to formulate the problem mathematically, and many exact methods need a mathematical formulation of a problem in order to solve it.

It should be noted that, in the case of a multiobjective optimization prob-

lem, depending on the multiobjective optimization method selected for solving the problem the actual scheduling problem solved can be polynomially solvable [30]. For example, let us consider a biobjective optimization problem in which minimizing the first objective function is NP-hard and the second objective function can be minimized in a polynomial time. Then, we have an NP-hard problem in general. If we choose to minimize the objectives lexicographically so that the second objective function is minimized first and then the first objective function is minimized subject to that the second objective function attains its minimal value, it is possible that the problem can be solved in a polynomial time: Minimizing the second objective function can be done in a polynomial time, and minimizing the first objective function subject to the new constraint may be solvable in a polynomial time because of the new constraint.

4.2 Methods for scheduling

In this section, we discuss existing scheduling methods that could be applied to the NCS problem. Since the NCS problem has two objectives, we concentrate here on algorithms that are developed for multiobjective scheduling problems. More algorithms for multiobjective scheduling problems can be found in the surveys [15, 22, 30, 31]. Methods for single objective parallel machines scheduling can be found in [21].

In scheduling, multiple objectives have been considered mostly in the case of single machine problems, and the studies have rarely involved any multiobjective analysis, as emphasized in [30]. In these studies, the aim is usually either to find all the Pareto optimal solutions to the problem, or a good representation of them, (see for example [4, 19]) or minimize the objectives in a predefined lexicographic order of importance (see for example [10, 32]). Methods producing many Pareto optimal solutions are not applicable to the NCS problem since we aim at an automatic solution method and do not have a decision maker who could select the final solution from the set of Pareto optimal solutions produced. In addition, we cannot define which of the objectives is more important and, therefore, we cannot minimize the objectives in a lexicographic order.

An approach to minimize the makespan (the time) and the costs has been presented in [28]. The algorithm proposed is a polynomial-time approximation algorithm that requires upper bounds C and T to be given a priori for the costs and the makespan, respectively. The algorithm produces a solution with costs at most C and makespan at most $2T$, if there exists a schedule with costs C and makespan T. A drawback in this kind of an algorithm is the selection of the upper bounds: They should be large enough to get a feasible solution, and small enough to get a good solution. Another disadvantage is that the solution obtained may be unsatisfactory since the makespan may be twice its limit. In the case of the NCS problem, the selection of the limits should be done by a decision maker, that is, the user of the mobile terminal who could express how much he/she is prepared

to pay for the transmission and how long he/she is willing to wait. However, it would be unpleasant for the user to express this kind of information every time he/she is transferring data. In addition, the solution obtained might be far from the best possible.

The most often used method to solve parallel machines scheduling problems is list scheduling. In list scheduling, the jobs that are to be processed are ordered on a list according to some criterion [12]. Then, every time a machine becomes available, that is, has finished a job, the next job on the list is given to the machine for processing. This simple greedy method works well for parallel identical machines that have equal processing times. It can be modified also for the case of non-identical parallel machines. However, when there are multiple objectives to be minimized it is difficult to find such an ordering of the jobs that the resulting schedule produces a good compromise.

Metaheuristics are high-level solution methods that can be applied to a wide variety of optimization problems with minor modifications [3]. Metaheuristics are, for example, tabu search and genetic algorithms. Metaheuristics are popular solution methods for NP-hard optimization problems with one or several objectives. A review of multiobjective metaheuristics used for scheduling problems is presented in [17]. Metaheuristics could be applied to the NCS problem. However, metaheuristics usually use a lot of computational capacity or need many iterations, which makes them slow. Therefore, metaheuristics are not suitable for our purposes. (Remember that we are searching for a method that is fast and uses as little computational capacity as possible.)

Since the NCS problem has not been studied earlier in the literature and we have not found any methods that could be directly applied to the NCS problem, we have developed a heuristic solution method for the problem. The solution method is general and can be applied to other multiobjective (scheduling) problems as well. In the next chapter, we discuss the development of the solution method, which was started by studying the nature of the problem in order to know what kind of a solution is a good compromise between the conflicting objectives.

5 SOLVING THE NCS PROBLEM

The first step in solving the NCS problem is to decide how to deal with the conflicting objectives. Since the problem has not been studied previously we have started the development of a solution method by getting to know the problem settings. The NCS problem has only two objective functions and, therefore, we can study the problem settings by forming approximations of the Pareto optimal set of some problem instances and drawing a graphical presentation of the Pareto optimal set. The aim is to get an idea of the properties of a good compromise between the conflicting objectives and how this good compromise can be obtained.

It should be remembered that the aim of this research is to develop an automatic solution method and, therefore, we cannot ask a decision maker to give preferences between the objectives. That is why we have concentrated in no-preference methods of multiobjective optimization that do not use preferences of a decision maker when searching for a solution. The methods tested in this research were briefly presented in Chapter 3. It should also be born in mind that the solution method developed should be fast and use only little computational capacity so that it is possible to implement it in a mobile terminal.

In this chapter, we discuss solving the NCS problem. We summarize the results of different multiobjective optimization methods used for solving the problem (Section 5.1). The results imply a need for a heuristic solution method and, therefore, we discuss the development of a heuristic for solving the NCS problem (Section 5.3). We also show an example of the NCS problem in order to illustrate solving the problem (Section 5.2).

5.1 Dealing with the conflicting objectives

The development of a solution method for the NCS problem was started by studying how to deal with the conflicting objectives. The results of this phase are presented in **PI**. We next describe the research done for finding a method that produces a good compromise.

In order to get to know the nature of the NCS problem, we formed approximations of the Pareto optimal sets of some problem instances (of which three instances are presented in **PI**) using the method of weighted metrics with three different metrics. An approximation is obtained by varying the weighting coefficients in the method. The main interest in these approximations has been on getting some understanding about the form of the Pareto optimal set and especially the 'middle' part of the Pareto optimal set, not the extreme Pareto optimal solutions, since we are searching for a method for producing a good compromise between the objetives.

In the approximations obtained using the method of weighted metrics, there were large gaps between some Pareto optimal points (see **PI**). This is natural since the NCS problem has integer variables. However, the gaps could also be due to the method, if the method was unable to find some Pareto optimal solutions, for example, because of the selection of the weighting coefficients. Therefore, the scalarizing function of the GUESS method [20] was used to ensure that the gaps in the approximations produced by the method of weighted metrics belong to the Pareto optimal sets. The approximation produced by the scalarizing function of the GUESS method was similar to the approximation obtained by the method of weighted metrics with the Tchebycheff metric.

When we studied Pareto optimal sets of different problem instances we found that, in all of the instances, there exists a compromise solution (or several near-by solutions) that can be considered as a good candidate for the final solution. These candidates were, however, produced with different weighting coefficient values in the method of weighted metrics and, therefore, another method for producing the good compromise solution was needed. The method should naturally be robust so that it can produce a good solution for any example and preferable not have any artificial parameters. The methods studied in this phase were the method of weighted metrics with L_1 and L_2 metrics, the method producing a neutral compromise solution (4) and the achievement scalarizing function (6). We also used another formulation of problem (4) in which the scaling of the objective functions was left out. These methods were selected because their ideas are different and we wanted to test different kinds of methods in order to find the most suitable for the NCS problem.

The method of weighted metrics with the L_2-metric produced good compromise solutions only with a very large value of the first weighting coefficient w_1 in some problem instances. We studied different methods for scaling the objective functions to be of equal magnitude in this method, but unfortunately scaling did not change the need for a large value for the weighting coefficient w_1. This kind of behaviour when the L_2-metric was used was not expected. It seems that this method with the L_2-metric cannot handle objective functions that have very different ranges, though the functions are scaled to be of equal magnitude. Therefore, this method was not considered to be appropriate for our purposes.

The method of weighted metrics with the L_1-metric, on the other hand, produced good compromise solutions with a large range of the weighting coefficient values. However, there is a drawback in the L_1-metric: if the Pareto optimal set

is nonconvex, the method cannot find solutions located in the nonconvex part of the set [5]. In addition, though the Pareto optimal set would be convex, the values of the weighting coefficients are difficult to choose beforehand because the ranges in which the good compromise solutions are obtained were different in each problem instance studied though we studied only few examples. Also the scaling of the objective functions may affect the ranges, as was noticed when the L_2-metric was used. Therefore, we could conclude that the method of weighted metrics cannot be considered to be robust enough to be used in an automatic solution process.

It should be noted that the method of weighted metrics with the L_1-metric is the same as the well-known weighting method when $(0,0)^T$ is used as the reference point. We studied also this version of the method since the weighting method is usually the first thing that comes to mind when a multiobjective optimization problem is to be solved. The weighting method has also the above mentioned drawback that if the Pareto optimal set is nonconvex, the method cannot find all the solutions in the Pareto optimal set. The good compromise solutions of the examples studied were found with a large range of the weighting coefficient values. However, the values were different for each example and, therefore, it is difficult to select such weighting coefficient values that the method would produce a good compromise for any problem instance. In addition, it is not clear how the weighting coefficient values affect the solution: a small change in the values can make the solution very different [20]. Therefore, the weighting method is not suitable for solving the NCS problem.

As was mentioned earlier, we have used different methods for scaling the objective functions to be of similar magnitude in the method of weighted metrics. The methods for scaling are according to [29]: normalization, use of ten raised to an appropriate power and range equalization factors. Normalization means dividing the objective vector by its norm. Instead of normalization, we used another similar scaling: we divided each objective function f_i by its range in the Pareto optimal set, that is, by $z_i^{nad} - z_i^*$. The scaling that uses range equalization factors multiplies each objective function by its *range equalization factor*

$$\pi_i = \frac{1}{R_i}\left[\sum_{j=1}^{k}\frac{1}{R_j}\right]^{-1},$$

where R_i is the *range width* of the ith objective function over the Pareto optimal set [29]. The range width can be calculated as follows

$$R_i = z_i^{nad} - z_i^{**}.$$

The different scaling methods used affected the ranges of the weighting coefficient values with which the good compromise solutions were obtained. However, none of the methods was superior to the others when we consider the ranges of the weighting coefficient values.

In the method producing a neutral compromise solution (4) as well as in the achievement scalarizing function (6) the objective functions are already scaled

to be of similar magnitude using the ideal and nadir objective vectors. We had large hopes that the neutral compromise solution would be a good compromise in the problem instances studied. Unfortunately, this was not the case in all the instances (see **PI**). Therefore, we needed to adjust the method and it was done by adding weighting coefficients which produced the achievement scalarizing function (6). We had hoped that we need not to define artificial parameters such as weighting coefficients in the method selected since the selection of, for example, weighting coefficient is not necessarily easy. However, the achievement scalarizing function (6) with values 1 and 2 for the weighting coefficients w_1 and w_2, respectively, worked well in all the problem instances studied.

The selection of the weighting coefficient values was done by testing different values and observing how the solution is affected. The values 1 and 2 for the coefficients w_1 and w_2, respectively, gave the best results. The objective functions are already scaled to be of similar magnitude and the ratio of the weighting coefficients represents the rate at which the user is willing to trade off values of the objective functions. The user of a mobile terminal could be given the possibility to change the values of the weighting coefficients. This would be done in the settings of the mobile terminal when wanted and the values 1 and 2 would be the default values. Thus, the user would not need to consider the weighting coefficients unless he/she wants to.

It is important to note that the achievement scalarizing function (6) is not as sensitive to the changes in the weighting coefficient values as the method of weighted metrics because the scalarizing function has also other parameters that affect the solution, such as the reference point, which is the middle point in the Pareto optimal set in this case. Therefore, the selection of the weighting coefficients is not as crucial in this method as in the method of weighted metrics. The strengths of the achievement scalarizing function (6) are that it produces Pareto optimal solutions and that it can find any (properly) Pareto optimal solution. Note also that when we later studied some additional problem instances (of which one example is presented in the next section) forming approximations of their Pareto optimal sets, the achievement scalarizing function produced good compromise solutions also for these new problem instances.

Before minimizing the achievement scalarizing function (6), the ideal and nadir objective vectors have to be calculated. This means additional computations slowing down the solution method considerably. This can be avoided by using their approximations. This is possible since the vectors are used mainly for scaling the objective functions and the method is not very sensitive to changes in the values of the vectors. Therefore, we have selected to use vector $(0,0)^T$ to approximate the ideal objective vector and to estimate the nadir objective vector from the problem data as follows. The first component of the vector is the objective function value $f_1(x)$ related to a solution x where every component is transferred using the slowest connection. The second component of the vector is the objective function value $f_2(x)$ related to a solution x where every component is transferred using the most expensive connection. These estimates are larger than or equal to the components of the real nadir objective vector and, unfortu-

nately, they may be very rough. However, the estimates are sufficient for our purposes as noticed in **PI**.

Scalarization converts the original problem into a combinatorial single objective optimization problem. We have used the CPLEX 8.0 software package to solve the single objective problems. CPLEX can solve linearly constrained optimization problems in which the objective function is linear or quadratic [16]. The variables may be continuous or integer-valued. CPLEX uses a branch-and-cut algorithm [33] to solve (mixed) integer programming problems. Solving the NCS problem using exact methods such as the branch-and-cut algorithm that CPLEX uses is time-consuming. The computational results in **PI** showed that the solution to the achievement scalarizing function (6) is a good compromise between the conflicting objectives but solving problem (6) is time-consuming using exact methods. In addition, many exact methods need a lot of computational capacity when solving the NCS problem. Therefore, a heuristic solution method giving a solution near the optimal solution to the achievement scalarizing function (6) is needed in our case. We have developed a fast heuristic for solving the NCS problem. The heuristic is discussed in Section 5.3. Before that, in the next section, we show an example of the NCS problem to clarify the situation.

5.2 Example

Next we present an example of the network connection selection problem in order to illustrate the problem settings and the compromise solution produced by minimizing the achievement scalarizing function (6). In this instance, there are 13 data components that are to be transferred using a set of eight network connections. The components are obtained from the Internet and they form a real web page. The prices and rates of the connections are estimates for network connections of the near future. More information on how these estimated were obtained, especially the prices, can be found in Section 6.2, in which we discuss the test instances used.

In this example, the cost of transferring component j using network connection i is calculated using formula

$$c_{ij} = k_0 + k_1\sqrt{r_i} \cdot s_j \tag{7}$$

where r_i is the rate of connection i (Kbits per second), s_j is the size of component j in bytes, and k_0 and k_1 are connection-specific coefficients. The connections form three groups that have different parameter values in the pricing model. In the first group, the values for parameters k_0 and k_1 are 300 and 1.1, respectively. In the second group the values are 100 and 2.3, and in the third group 30 and 0.8, respectively. The rates of the connections are 14.4, 59.2, and 384 Kbits per second in the first group, 14.4, 59.2, and 115 Kbits per second in the second group, and in the last group 14.4 and 59.2 Kbits per second. The price given by formula (7) is multiplied by 10^{-5} in order to have it in euros.

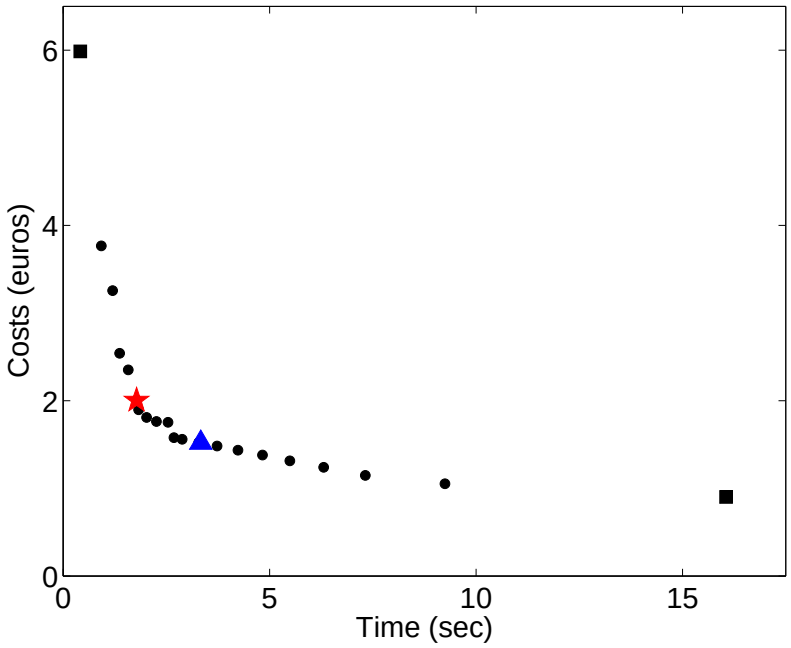


FIGURE 1 Pareto optimal solutions: star represents the solution to problem (6), triangle the solution to problem (4), and the solutions depicted by squares minimize time or costs.

An approximation of the Pareto optimal set for this problem instance is presented in Figure 1. The approximation contains some (not necessarily all) Pareto optimal solutions and they show the form of the Pareto optimal set. The axis labelled time presents the value of the objective function f_1 and the axis labelled costs shows the value of the objective function f_2. The approximation has been formed by solving the augmented weighted Tchebycheff problem (3) with different values of the weighting coefficients. The augmentation coefficient ρ was given a value 0.001, and the problem was solved using CPLEX 8.0. In the approximation, the solutions obtained by minimizing only one of the objective functions are depicted by squares. The solutions obtained by solving problems (4) and (6) (the method producing a neutral compromise solution and the achievement scalarizing function, respectively) are also presented in the figure. A triangle depicts the neutral compromise solution (solution to problem (4)) and a star represents the solution to the achievement scalarizing function (6).

It should be noted that because we are dealing with an integer programming problem, the Pareto optimal set is a collection of points where there may be even large gaps between the points. Therefore, there do not necessarily exist solutions between the solution points depicted in Figure 1. In addition, it is important to note that producing an approximation of a Pareto optimal set is time-consuming. Therefore, it is not possible to form an approximation during the actual solution procedure that has to be fast when this kind of a real-time problem is solved. However, approximations of Pareto optimal sets can be used to evaluate solutions when developing a solution method, as is done in **PI**.

As can be seen in Figure 1, the solution to problem (6) is a good compromise between the two objectives: The time used in transferring is small (1.78 seconds) and the costs are reasonable (2.00 euros). In this problem instance, the neutral compromise solution is also a good compromise with time of 3.33 seconds and costs of 1.52 euros. However, in some test instances the neutral compromise solution was far from a good compromise (see **PI**).

5.3 Heuristic

Since the NCS problem is NP-hard, it is unlikely that there exists a polynomial-time algorithm to solve the problem. In other words, solving the NCS problem using exact methods is computationally demanding, and it takes a lot of time to solve large problem instances. In addition, exact methods require a lot of computational capacity which is limited in mobile terminals. Therefore, we have developed a heuristic to solve the NCS problem. The ideas of the heuristic were first introduced in [27]. The ideas were developed further using computational tests [26] and then the heuristic was presented in **PII**.

The basic idea of the heuristic is simple and absolutely not new: We form an initial solution by assigning one component at a time to some connection. The initial solution is then improved by local search using simple moves called 1-exchanges. In a 1-exchange, a component is moved to another connection in the solution if the solution quality improves. (The solution is, in this case, the assignment of the components to the network connections.) What is new in the heuristic is that the solution quality is measured using the achievement scalarizing function (6). Usually in heuristics developed for scheduling problems, the objectives are considered separately. For example, the objectives are minimized one objective at a time or an upper bound is given to one objective and then the other objective is minimized subject to that the upper bound is not exceeded.

Using a scalarizing function to measure the solution quality has the advantage that we can consider as many objectives as wanted at the same time, and if new objectives arise we do not have to develop a new solution method. This is very appealing, especially in this case in which the problem is new and have not being studied previously, because new objectives may arise when we learn more about the problem. It should be noted that adding a new objective requires naturally more research and tests but the same heuristic is still applicable to the problem.

During the development of the heuristic, the basic idea of improving an initial solution using 1-exchanges was kept unchanged. However, how the initial solution should be formed was under a lot of study and many different versions were considered. We studied, for example, the following methods:

- assigning the components to the connections in such a way that the number of components on each connection is the same,

- minimizing the time used in transferring,
- giving an upper bound on the time used in transferring and then assigning as many components as possible to each connection starting from the cheapest connection in such a way that the upper bound is not exceeded. (Different values for the upper bound were naturally tested.)

The reason for studying this kind of methods for forming the initial solution was that we hoped to form an initial solution which is easy to improve and in which the components are assigned to the connections rather evenly. In those methods in which the components are assigned one by one to the connections, ordering the components before assigning them was studied: In addition to a random order, descending and increasing orders of the components' sizes were studied. These different orders were studied also in the case of using the 1-exchanges to each component in turn. In addition to different initial solutions, different functions to measure the solution quality when using 1-exchanges were studied. However, the achievement scalarizing function (6) was found the best for measuring the solution quality during the entire heuristic.

In the final version of the heuristic, the initial solution is formed as follows: The components are assigned one by one to the connection that gives the lowest value of the achievement scalarizing function (6) to the current partial problem. The current partial problem consists of all the connections available and the components that are already assigned (with fixed assignments) in addition to the component that is to be assigned at the moment. The order in which the components are assigned to the connections is random, since the other orders studied did not improve the performance of the heuristic. After the initial solution is formed, the solution is improved using 1-exchanges to each component in turn. In the 1-exchanges, the solution quality is measure using the achievement scalarizing function (6). The 1-exchanges are applied to the components until a stopping criterion is satisfied. The stopping criterion can be, for example, a certain number of times 1-exchanges applied. Based on the computational results presented in **PII**, we have applied 1-exchanges to each component in turn twice. After this improvement phase, the components are assigned to the connections but the order of the components on each connection is still to be defined. The order has no effect on the objective functions, that is, the time used in transferring and the costs, as mentioned in Chapter 2, but it is needed when the components are transferred. We have used a random order of the components, but the components could be ordered, for example, according to their sizes.

The heuristic can be briefly summed up as follows.

1. *Initial solution*. For each component, find the network connection that gives the lowest value of the scalarizing function (6) for the current partial problem, and assign the component to that connection. The current partial problem consists of all the connections, the component to be assigned at the moment and the components that are already assigned (with fixed assignments).

2. Set the initial solution formed in the previous step as the current solution.

3. *Improvements*. Repeat twice: For each component, use a 1-exchange to improve the current solution: Consider transferring the component using another connection than the one it is currently assigned to. Select the connection that gives the lowest value of the scalarizing function (6) when the component is transferred using it and all the other assignments are fixed. If the value of the scalarizing function is smaller than in the current solution, set the component to the selected connection and set this solution as the current solution.

4. *Ordering*. Order the components assigned to each network connection according to some criteria. The components will be transferred in that order.

For more details of the heuristic, see **PII**.

The heuristic was computationally tested and the results of the tests are presented in **PII**. The performance of the heuristic was compared to solving the problem exactly using CPLEX 8.0. The solutions produced were compared with respect to the values of the objective functions, that is, the time used in transferring and the costs. In addition, the difference between the heuristic and optimal solutions was measured using the achievement scalarizing function (6). The computational results showed that the heuristic performs well and finds a solution near the optimal solution to the problem. We also compared the time used in solving, that is, the time needed for finding a solution. The heuristic is fast and the time needed for solving is short even for larger problem instances, whereas the time needed for solving by CPLEX grows exponentially when the size of the problem instance grows. CPLEX needs also a lot of computational capacity to solve the problem, while the heuristic was developed avoiding that in order to make it possible to implement it in a mobile terminal.

In the case of the problem instance presented in Section 5.2, the heuristic produces a solution near the solution to the achievement scalarizing function (6). The heuristic solution uses 2.02 seconds to transfer the data while the costs are 1.97 euros. (The corresponding values of the optimal solution to the achievement scalarizing function (6) are 1.78 seconds and 2.00 euros.) The difference in the value of the objective function (6) is 0.0135 between these two solutions. The heuristic uses only little computational capacity and is fast. For this instance, CPLEX uses 0.11 seconds of CPU time to solve problem (6) while the heuristic uses 0.01 seconds.

One might ask why to use such a simple heuristic and not to develop the heuristic further. First of all, based on the computational results, the heuristic performs well: It finds a solution near the optimal solution to the achievement scalarizing function (6). Secondly, the method developed should be fast and use only little computational capacity since it is to be implemented in a mobile terminal. In addition, so far we have considered a static environment in which there are no changes in the network conditions during a transfer. This assumption is not always valid since wireless networks are highly dynamic. Therefore, the heuristic needs to be modified to take into account changing network conditions during

the transfer. In this case, the heuristic should not use a lot of time in finding a solution since the solution may become infeasible and then a new solution needs to be found. In the next chapter we will discuss dynamic network environments and how the heuristic is modified to take into account the changing conditions.

6 DYNAMIC ENVIRONMENT

Wireless networks are dynamic and conditions in the networks may change very rapidly. For example, congestion can slow down a network connection notably. In addition, pricing of network connections can vary in time: Dynamic network pricing has been considered as a promising economical approach for solving congestion problems that have been diminishing the benefits of the wireless Internet [24]. Naturally, also the movement of a mobile terminal can cause changes in the wireless environment: New networks may appear and network connections may be lost. Therefore, we need to take into account the dynamic nature of the wireless networks. The heuristic presented in **PII** assumes a static environment in which the conditions do not change during the transmission. This kind of an assumption is valid only when the transmission time is very short and, even then, we cannot assure that the conditions do not change. In this chapter, we discuss an improved version of the heuristic that takes into account changing network conditions during the transfer (Section 6.1). We also briefly discuss the computational tests run using a simulator and the test instances used in the tests during this research (Section 6.2).

6.1 Dynamic heuristic

As stated in [14], a schedule is optimal only when the real world behaves as expected when the schedule is executed. Therefore, it is not wise to use a lot of time and effort to find an optimal schedule (or a solution to a dynamically changing problem) when we cannot predict the real world behaviour. Keeping this in mind, we have studied the case in which the network conditions may change during the transmission. We have concentrated on developing a computationally light method that reacts to changing network conditions. The method is called the *dynamic heuristic* and it is based on the same ideas as the heuristic presented in the previous chapter and in **PII**, which is called the *static heuristic*. The idea of the dynamic heuristic is shortly introduced in **PIII**. In more detail, the dynamic

heuristic is presented in **PIV**, in which simulation results of using it are also presented.

When the dynamic heuristic was developed as well as in the simulations run for testing the heuristic, we have considered four kinds of changes in the wireless network environment: a network connection currently available is lost, a new network connection becomes available, the transmission rate of a connection changes and the price of a connection changes. The availability of a connection can naturally change due to the movement of a mobile terminal. It is also natural that transmission rates change during a transmission because of, for example, congestion on networks. However, it is not clear if the price of a connection can change during a transmission. The user of a mobile terminal is a customer of the operator charging for the use of a connection. The customers must know how much the transmission will cost and, therefore, the prices cannot change without the customers knowing it. Though it might be complicated, we have considered also this case in which the prices can change during a transmission. The idea behind this scenario is that the users pay only for the rate they use. In other words, the price is dependent on the transmission rate used at the moment and when the rate changes also the price changes. This kind of a pricing would be fair to the customer but it is not simple to define how the price information could be given in real-time so that the customer would be aware of the price all the time. In addition, the operators might not want to use such a pricing method. This issue is, however, out of the scope of this thesis.

The dynamic heuristic uses the static heuristic to form the transmission schedule before the transmission is begun. The transmission schedule that is followed during the transmission is changed during the transmission if a better schedule is available. The quality of the schedule is measured as in the static heuristic using the scalarizing function (6). In other words, a new schedule is better than the old one if the value of the scalarizing function for the new schedule is lower than the value of the old schedule. The new improved schedule is formed by using 1-exchanges which are also used in the static heuristic. This improvement phase of the dynamic heuristic is kept simple and fast in order to make it possible to use the method during the transmission without slowing down the transmission. The improvement phase considers a partial problem that consists of all the currently available connections and those components that are waiting to be transferred. The components that are currently being transferred are not considered here and their transmissions can continue during the calculations of the new improved schedule. The transmission of the other components is not allowed until a new schedule is found. Note that if a network connection is lost during the transfer of a component, the component has to be transferred again using another connection. Then, this component is also considered in the partial problem. If the new schedule formed is better than the current one (measured using the scalarizing function (6)) the rest of the components are transferred using the new schedule.

It is not wise to try to improve the schedule constantly during the transmission, because the transmission of the components that are not yet been transferred

is not allowed until a new schedule is formed, even though the schedule might not be changed. In addition, the computational capacity of mobile terminals is limited, and it should not be waisted in unnecessary calculations. Therefore, we have limited running the improvement phase of the dynamic heuristic to three situation: when a network connection is lost, a new connection has appeared or a change in the transmission rates or prices is large enough. In the case of a lost connection, it is important to run the improvement phase. Before the improvement phase using the 1-exchanges is run, the components that are assigned to the lost connection and have not yet been transferred have to be assigned to other connections. (If a component was being transferred when the connection was lost, the component has to be transferred again and also this component is assigned to a new connection.) The assignment of the components to new connections is done similarly as in the static heuristic when the initial solution is formed: Each component is assigned to the network connection available that gives the lowest value of the scalarizing function (6) to the partial problem consisting of the currently available connections and the components that are already assigned in addition to the component being assigned. In the case in which a new connection has appeared the schedule is still feasible, but an improved schedule may be obtained by using the new connection. In this case, the improvement phase can be run without any additional preparations. Similarly, when there has happened a change in the transmission rates and prices, an improved schedule may be available. If the change is small, it is unlikely that there exists a better schedule. Therefore, we have set a limit on running the improvement phase in the case of a change, and require that the change is larger than a predetermined bound B. How the dynamic heuristic responds to a change in the network conditions is summarized in Figure 2, in which *rescheduling* refers to running the improvement phase of the dynamic heuristic in order to find a new improved schedule.

The dynamic heuristic can be summarized as follows.

1. *Initialization*. Use the static heuristic to form a transmission schedule that is followed during the transmission.

2. *Change in the network conditions*. When a change in the network conditions occurs, control which connections are currently available and which components are waiting to be transferred using the connections available. Remember the current assignment of the components to the connections.

   Depending on the type of the change, do the following.

   - *Lost connection*. If a connection was lost, remove components that are waiting to be transferred using the lost connection from the transmission schedule. If the transfer of a component was interrupted when the connection was lost, remove also this component from the transmission schedule. Assign these components to the available connections: For each component, assign the component to the connection that gives the lowest value of the scalarizing function (6) to the partial problem consisting of all the available connections, the components waiting to

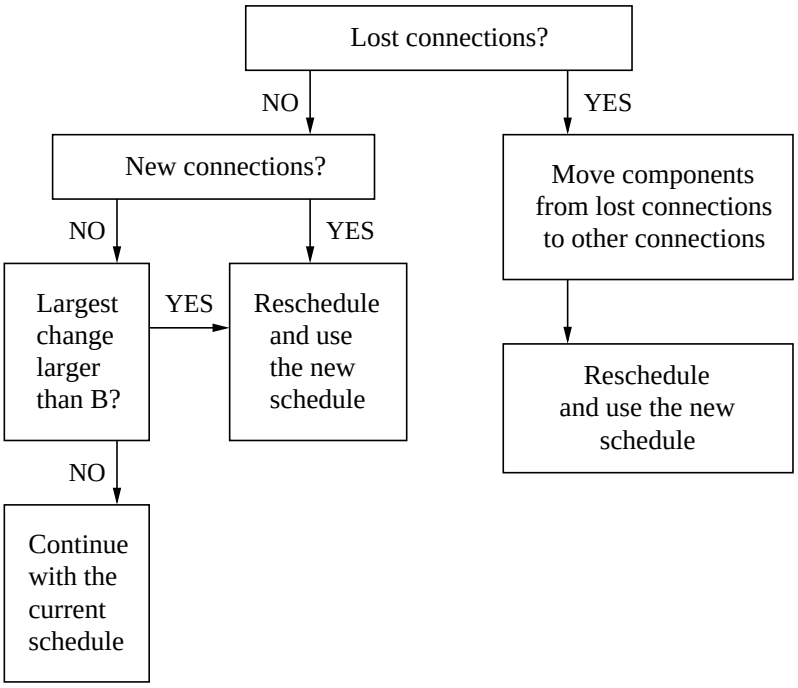


FIGURE 2 How the dynamic heuristic responds to a change in the network conditions.

be transferred using these connections (with fixed assignments) and the component currently being assigned. Go to Step 3 and use this assignment as the initial solution.

- *New connection*. If a new connection has appeared, go to Step 3 and use the current transmission schedule as the initial solution.
- *Change in the rates and prices*. If the largest change is larger than B, go to Step 3 and use the current transmission schedule as the initial solution. Otherwise continue with the current transmission schedule.

3. *Rescheduling*. Set the initial solution as the current solution. Use rescheduling to each component in turn: Consider a set of solutions where the component is assigned to another connection instead of the connection to which it is assigned in the current solution. Select among the solutions the one having the lowest value of the scalarizing function (6) to be a *rescheduling solution*. If the rescheduling solution has a lower value of the scalarizing function (6) than the current solution, set the rescheduling solution as the current solution.

4. *New transmission schedule*. Set the current solution formed in the previous step as the final solution. If the final solution is different from the current transmission schedule, form a new transmission schedule from the final solution by ordering the components on each connection according to some criteria, and use this new schedule to transfer the components.

(For more details of the dynamic heuristic, see **PIV**.)

How to define when a change is large enough is not an easy question. We have run simulations and based on the results we have concluded in **PIV** that a default value for the limit of a large change is 40% of the previous rate or price. In other words, if a change in the transmission rates or prices is larger than 40% the change is large enough. The results of the simulations presented in **PIV** were not easy to interpret, because different limits for the amount of a large change gave results that were difficult to compare without a decision maker. Thus, the results showed that it would be advantageous to know some preferences of the user of the mobile terminal. We have concentrated on developing an automatic solution method in which the user is not involved, and an automatic method for selecting the connections used for transferring is clearly needed. However, some kind of preference information given a priori would be helpful. The preference information could be given in the settings of the mobile terminal when wanted, and default values would be used otherwise. This preference information could be given, for example, in a user profile, and it would help in deciding when the schedule should be improved during the transmission. More discussion on this issue can be found in Chapter 8.

6.2 Computational tests

In this section, we briefly discuss the computational tests run when developing the dynamic heuristic and the test instances used in the computational tests run during this research.

In order to test the dynamic heuristic a simulator was needed. We implemented a simulator that generates three kinds of changes in the network environment: a network connection is lost, a new connection appears or the rate of a connection changes. The simulator was written in programming language C++ and it was run on a UNIX environment in a computer with a 550 MHz PA-RISC processor. In the tests run using the simulator, the performance of the dynamic heuristic was compared to the performance of the static heuristic. This was done by transferring data using two different strategies in the simulations: In the first strategy, the transmission schedule given by the static heuristic was followed during the entire transmission. The second strategy used the dynamic heuristic and changed the transmission schedule when a better schedule was available. Since the first strategy used fixed assignments of the components, the transmission was stopped on a connection if the connection was lost, until the connection was available again. The other strategy, on the other hand, moved the components to other connections if a connection was lost.

We have run 2000 simulations with 25 different test instances, and the results are presented in **PIV**. In the simulations, we have used both fixed pricing and dynamic pricing for the use of the network connections. In the fixed pricing, the price of using a network connection is constant during the transmission though the transmission rate may change. In the dynamic pricing, on the other

hand, the price is dependent on the transmission rate and changes accordingly during the transmission. A change in the transmission rates is observed immediately and the new rate is valid from that moment on. However, the new price based on the new transmission rate is valid from the transmission of the next component. This way it is possible to stop using the connection if its price is not acceptable. For more information on the simulations and the results, see **PIV**.

We next discuss the test instances used in this research with a special emphasis on the pricing models used.

Test instances

We have used different test instances in the different phases of our research. Each test instance consists of a set of network connections and a set of data components that are to be transferred using the network connections. The components of each test instance are obtained from the Internet and they each form a real web page. The web pages were selected randomly from the known web pages. In the first phase of testing different multiobjective optimization methods, we had a selection criterion for the web pages: The number of components had to be quite small in order to make it possible to calculate approximations of the Pareto optimal set using different methods in a reasonable time using CPLEX. In the other phases, there was no limitations or criteria for selecting a web page to be used in the test instances.

The network connections used in the test instances were formed in such a way that they represent possible wireless connections available in the near future. Since we cannot predict the pricing of the future network connections, we have used different pricing models. The parameters of the pricing models were given values based on fitting current data transmission prices for GSM and UMTS networks in Finland into the pricing models using the method of least squares. We next present the different pricing models used.

In the test instances, the cost of transferring component j through network connection i (denoted by c_{ij}) is calculated using formula

$$c_{ij} = g(B_i) \cdot S_j + k, \tag{8}$$

where S_j is the size of component j in bits, B_i is the transmission rate of connection i (bits/s), and $k \geq 0$ represents fixed costs of the transfer. Function g is used to calculate variable costs of the transfer related to the size of the component. The variable costs depend also on the transmission rate of the connection. We have used three different functions g_1, g_2, and g_3 to model the variable costs.

The first function g_1 models the price as linearly dependent on the transmission rate B:

$$g_1(B) = k_1 B,$$

where k_1 is a connection-specific coefficient. The higher the transmission rate is, the higher the price per bit is. The second function g_2 defines an exponential relation between the price and the transmission rate:

$$g_2(B) = k_1 \exp(k_2 B),$$

where k_1 and k_2 are connection-specific coefficients. This function models a situation where high-speed connections are very expensive. The reason for this may be, for example, an attempt to prevent congestion in networks. The third function g_3 is the most often used model for network pricing [24]. It is a function in which the exponent is less than one. In this case, the exponent is 0.5. Thus, we have

$$g_3(B) = k_1\sqrt{B},$$

where k_1 is again a connection-specific coefficient. In this function, the price is less strongly dependent on the transmission rate than in functions g_1 and g_2.

Using these pricing models five sets of network connections were formed, each set containing three to eight network connections. In the sets having more than three network connections, there are connections that have the same rate but different prices. These connections represent competing operators that have similar networks to offer for the customers but use different pricing for the use of the network. The sets of network connections were randomly combined with different web pages that form the data to be transferred. This way we obtained the test instances used in the computational tests run. It should be noted that different test instances were used in the different computational tests run during this research. Thus, the test instances used when developing the static heuristic were not used in the simulations run in order to test the dynamic heuristic.

7 PRECEDENCE CONSTRAINTS

So far, we have assumed that the data components that are transferred are independent of each other. This has made it possible to transfer them in any order. However, in some cases, the data may contain some components that have to be transferred before some other components. For example, if the data to be transferred contains a compression program and data components that have been compressed using the program, it is natural to transfer the program first so that the data components can be decompressed as soon as they arrive at the destination. When there are such priorities between the components we need to take into account the order already in the mathematical model as well as when the components are assigned to the connections. This complicates the problem.

In this chapter, we consider a case of the NCS problem in which there are precedence relations between components requiring that some components have to arrive at the destination earlier than some others. We give a mathematical model of the problem and discuss solving it. This problem case with precedence relations has not been considered in the included articles. When we began studying the NCS problem we decided not to consider precedence relations in the early phases of the research for two reason. First, the NCS problem with precedence relations is much more complicated to solve than the NCS problem considered so far which already has many challenges. Second, if it is not absolutely essential that all the precedence relations are followed, the precedence relations can be taken into account in the heuristic presented in Section 5.3 in such a way that at least some precedence relations are followed. We discuss this issue in more detail after we have presented a mathematical model for the NCS problem with precedence relations.

7.1 Model

Let us assume that there are n components to be transferred using m network connections. A *precedence relation* marked by $i \prec j$ claims that component i has

to arrive at the destination before component j is transferred. For a mathematical model of the NCS problem with precedence relations between some components, the following variables are needed: Variable C_i, $i = 0, \ldots, n$, denotes the time component i arrives at the destination. Variable y_{jk}, $j = 1, \ldots, n$, $k = 1, \ldots, m$, is given value 1 when component j is transferred using network connection k, otherwise it is zero. Variable x_{ijk}, $i = 0, \ldots, n$, $j = 1, \ldots, n$, $k = 1, \ldots, m$, equals 1 when component j is transferred immediately after component i on connection k, otherwise it is zero. In addition, a dummy component 0 is needed in order to be able to calculate the arrival time for the first component transferred on each connection. The dummy component is transferred in null time, that is, the arrival time of the dummy component C_0 is always zero. Now, the NCS problem with precedence relations can be formulated as follows.

$$\text{minimize} \quad f_1(x) = \max_{j=1,\ldots,n} C_j \quad \text{and} \quad f_2(x) = \sum_{j=1}^{n} \sum_{k=1}^{m} c_{jk} y_{jk} \tag{9}$$

$$\text{subject to} \quad \sum_{i=0, i\neq j}^{n} \sum_{k=1}^{m} x_{ijk} = 1 \quad \text{for all} \quad j = 1, \ldots, n, \tag{10}$$

$$\sum_{i=0, i\neq j}^{n} x_{ijk} = y_{jk} \quad \text{for all} \quad j = 1, \ldots, n, \quad k = 1, \ldots, m \tag{11}$$

$$\sum_{j=1, j\neq i}^{n} x_{ijk} \leq y_{ik} \quad \text{for all} \quad i = 1, \ldots, n, \quad k = 1, \ldots, m \tag{12}$$

$$\sum_{j=1}^{n} x_{0jk} \leq 1 \quad \text{for all} \quad k = 1, \ldots, m \tag{13}$$

$$C_j \geq \sum_{k=1}^{m} (C_i + d_{jk}) x_{ijk} \quad \text{for all} \quad j = 1, \ldots, n, \quad i = 0, \ldots, n, \quad i \neq j \tag{14}$$

$$C_j \geq C_i + \sum_{k=1}^{m} d_{jk} y_{jk} \quad \text{for all} \quad i, j = 1, \ldots, n, \quad i \prec j \tag{15}$$

$$x_{ijk} \in \{0, 1\} \quad \text{for all} \quad i = 0, \ldots, n, j = 1, \ldots, n, k = 1, \ldots, m \tag{16}$$

$$y_{jk} \in \{0, 1\} \quad \text{for all} \quad j = 1, \ldots, n, \quad k = 1, \ldots, m \tag{17}$$

$$C_i \geq 0 \quad \text{for all} \quad i = 1, \ldots, n \tag{18}$$

$$C_0 = 0 \tag{19}$$

The first objective function to be minimized is the time used in transferring and the second objective function denotes the total costs of the transfers. Constraints (10) require that each component is transferred exactly once in a specific time slot using exactly one network connection. Constraints (11) define that a component follows immediately its preceding component on the network connection that is used for transferring them. Constraints (12) define in turn that each component is immediately followed by at most one component. Constraints

(13) require that on each connection there is only one component to be transferred first. The arrival times of the components are given by constraints (14) and precedence constraints (15) make sure that the precedence relations are followed in the solution. The objective function f_1 is nonlinear. However, we can make the model linear by using an additional variable as was done in Chapter 2 for the original model without the precedence constraints.

Note that a set of constraints used in the original model of the NCS problem presented in Chapter 2,

$$\sum_{k=1}^{m} y_{jk} = 1, \qquad j = 1, \ldots, n,$$

is not needed in this model. These constraints are always valid because of constraints (10) and (11). In some cases, we might need another kind of a precedence relation, marked again by $i \prec j$, that defines that component i has to arrive at the destination before component j arrives. This kind of a precedence relation would be more natural in the case of the NCS problem. It can be achieved in the above model by replacing the precedence constraints (15) with the following precedence constraints:

$$C_j \geq C_i \quad \text{for all} \quad i, j = 1, \ldots, n, \quad i \prec j. \tag{20}$$

The total amount of variables in this model is $n^2m + 2nm + n$ (disregarding the dummy variable C_0). The number of constraints without the precedence constraints (15) and the constraints giving the ranges of the variables is $n^2 + 2nm + n + m$. In the original model without the precedence constraints, the number of variables is nm, and the number of constraints without the constraints giving the ranges of the variables is n. Thus, solving this model is much more difficult than solving the original model. Let us now consider one possible solution approach to the problem with precedence relations.

7.2 Solution approach

The above presented mathematical model of the NCS problem with precedence relations contains many variables and constraints. Therefore, it is difficult to solve the problem. The heuristic presented in Section 5.3 can be modified to solve also this case of the NCS problem. However, the solution quality may not be the best possible. Let us consider two cases of the NCS problem with precedence relations and how the heuristic could be modified to solve these cases.

If it is not essential that the transmission schedule obeys all the precedence contraints, these constraints can be considered as soft constraints. *Soft constraints* differ from regular constraints (that are called hard constraints in this case) in that a solution that does not obey the soft constraints is still feasible, whereas a solution that does not obey regular (hard) constraints is not a feasible solution to

the problem. If the precedence constraints can be considered as soft constraints, the heuristic presented in Section 5.3 can be applied also to this problem: After the components have been assigned to the connections, the components are ordered on each connection in such a way that the precedence constraints are valid on each connection. Naturally, some constraints are not necessarily valid in the transmission schedule obtained this way. The ordering of the components on each connection could also be done in such a way that the components on the other connections are taken into account. Thus, the components could be ordered on all the connections at the same time with special consideration given to those components that are involved in the precedence relations. However, we cannot guarantee that all the precedence constraints are valid in the schedule formed this way either.

If it is necessary to follow the precedence relations, that is, the precedence constraints are not soft, the constraints should be taken into account already when the components are assigned to the connections. This complicates the problem notably. In [13], three heuristic algorithms for solving a parallel machines scheduling problem with precedence constraints are presented. The objective in the problem considered is to minimize the makespan, so the heuristics presented are not directly applicable to the NCS problem, in which we want to minimize also the costs. Two of the heuristics presented first assign the jobs (components in the case of the NCS problem) to the machines (connections) and then order them according to the precedence constraints. The number of precedence constraints is assumed to be small so that it is possible to assign the components first and order them afterwards. A similar heuristic could be developed to the NCS problem with the precedence relations by using the heuristic presented in Section 5.3 in the assignment phase and then ordering the components on each connection in such a way that the precedence constraints are satisfied. Since the precedence constraints are not soft in this case, a simple ordering might not result in a feasible solution. Then, idle time needs to be inserted between transmissions of some components, which means that after the transmission of a component is finished there is a pause before the transmission of the next component on the same connection is started. The idle time inserted can naturally make the solution worse. However, if there are many components and only few precedence constraints, it is more likely that no idle time needs to be inserted or the idle time inserted is short.

This issue needs further study and, thus, the NCS problem with precedence relations will be considered in the future among other future research topics, which are summarized in the next chapter.

8 CONCLUSIONS AND FUTURE WORK

In this thesis, we have studied a problem called the network connection selection (NCS) problem. The problem consists of deciding which network connection to use for each component of data that is to be transferred using a set of different kinds of network connections available. The NCS problem has been formulated as a multiobjective integer programming problem in which the objectives are to minimize both the time used in transferring and the costs of the transfer. The aim of this study has been to develop an automatic solution method that can be implemented in mobile terminals. The NCS problem has not been studied in the literature earlier, and we have not found any existing methods that could be directly applied to the NCS problem. Therefore, we have developed a solution method by ourselves.

Before we started developing the solution method, we formulated the NCS problem mathematically and studied the problem settings with the help of some problem instances in **PI**. The aim of this phase was to learn about the properties of the problem considered, for example, the shape of the Pareto optimal set and then to find a suitable multiobjective optimization method that produces a good compromise between the conflicting objectives. In **PI**, different multiobjective optimization methods were tested and the most suitable for our purposes was selected.

Solving the NCS problem is difficult since the problem is NP-hard. In addition, to make it possible to implement the solution method in mobile terminals, the solution method should be fast and use only little computational capacity. Therefore, we needed to develop a heuristic to solve the problem. The heuristic is presented in **PII**, and it is a simple local search heuristic that solves the problem fast. Though the heuristic is simple, it is efficient: the computational results in **PII** showed that the heuristic reliably finds a solution near the optimal solution.

The heuristic presented in **PII** was developed under an assumption that the network environment is static, that is, there happens no changes in the network conditions during the transfer. This kind of an assumption is valid only when the transmission time is very short. Therefore, we have considered a dynamic network environment in **PIII**. The heuristic presented in **PII** was improved to

take into account possibly changing network condition in **PIII**. This version of the heuristic is called the dynamic heuristic. The dynamic heuristic was shortly presented in **PIII**. In **PIV**, the dynamic heuristic was presented in more detail, and the results of the simulations run for testing the dynamic heuristic were also presented.

The solution method developed for the NCS problem is general and it can be applied to other multiobjective integer programming problems as well. The NCS problem contains only two objective functions but the method is also applicable to problems with more than two objective functions. This is a good property since there are many issues involving the NCS problem that have not yet been considered. For example, we can consider a case of the problem in which the data components can be compressed before the transmission in such a way that they will be transferred faster. The time used in compressing a component has to be taken into account when considering the total time used in transferring so that it is not necessarily wise to compress every component. In addition, compressing a component makes energy consumption of the mobile terminal larger and also this issue should be considered when making the decisions. The energy consumption could be considered as the third objective function in addition to the time used in transferring and the costs.

The solution method developed is automatic, that is, we do not ask the user of a mobile terminal for any preferences. However, the simulation results of testing the dynamic heuristic implied that some kind of preference information from the user would be helpful in the solution method. In the future, we will consider this issue of how the preference information could be given in advance in such a way that the user would not be involved in the actual solution procedure. Naturally, not all the users are interested in giving preference information but the users could be given the possibility to give it if wanted. This could be done, for example, in the settings of a mobile terminal by introducing a user profile. In the profile, the user could, for example, deny the use of a network connection if he/she feels that the connection is too expensive or otherwise not satisfying. The user could also express how important the costs are to him/her, as well as the time used in transferring. The user profile could be updated as often as wanted, and if some values were not given, default values would be used.

In the future, we will also study the NCS problem in which there are precedence relations between components. In Chapter 7, we already presented a mathematical formulation of the problem and discussed one way to deal with the precedence relations. The solution approach proposed is not necessarily the best possible, and it requires further study and computational tests.

The NCS problem has arisen in the field of telecommunications, but it can also be used to model another kind of a transmission problem, namely the transmission of goods using different vehicles. For example, a warehouse needs to transfer goods in containers of various sizes. The containers can be transferred using trucks, trains, airplanes and ships, depending naturally on the case, and all of these vehicles have different rates at which the containers are transferred as well as different prices for the transfer. In addition, there are many companies

offering their services in the field of logistics, and the selection between these companies and different transmission methods could be done using the methods provided for the NCS problem. It should, however, be noted that there is more time available for making the selections in the case of goods than in the original NCS problem and, therefore, the heuristic is not necessarily needed for solving the problem as in the case of wireless networks. Also other application areas of the NCS problem should be studied in the future.

YHTEENVETO (FINNISH SUMMARY)

Tulevaisuuden langattomissa verkoissa päätelaitteen, kuten esimerkiksi matkapuhelimen, käyttäjä ei ole rajoitettu käyttämään vain tiettyä tiedonsiirtokanavaa, vaan hän voi vapaasti valita useista eri operaattorien tarjoamista kanavista. Tässä väitöskirjassa tutkitaan tilannetta, jossa päätelaite voi käyttää useita tiedonsiirtokanavia samanaikaisesti tiedon eri osien lähettämiseen ja myös vaihtaa käytettäviä tiedonsiirtokanavia lähetyksen aikana. Tällöin käyttäjä voi periaatteessa itse valita mitä kanavia hän käyttää kulloisenkin informaation eri osien lähettämiseen tai vastaanottamiseen. Koska vaihtoehtoja on paljon, voi valitseminen olla hidasta ja vaikeaa, eivätkä käyttäjät välttämättä ole halukkaita kuluttamaan aikaa siihen. Sen sijaan väitöskirjassa esitelty valinnan automatisointi mahdollistaa käyttäjille helpon tavan hyödyntää useaa tiedonsiirtokanavaa yhtä aikaa.

Tässä työssä mallinnettu usean tiedonsiirtokanavan käyttö samanaikaisesti nopeuttaa tiedonsiirtoa. Lisäksi vapaammat valinnanmahdollisuudet eri palveluntarjoajien välillä voivat laskea tiedonsiirron hintoja. Tiedonsiirron nopeus ja hinta ovatkin usein päätelaitteen käyttäjän kannalta tärkeimpiä ominaisuuksia tiedonsiirtokanavaa valittaessa. Tässä tutkimuksessa valinnan kriteereinä käytetäänkin tiedonsiirron kestoa ja kustannuksia. Keston ja kustannusten minimointi ovat kuitenkin ristiriitaisia tavoitteita, sillä nopein tiedonsiirtotapa on harvoin myös halvin. Tiedonsiirtokanavien valinta mallinnetaankin monitavoitteiseksi optimointiongelmaksi, jossa kaikkia tavoitteita ei voida yhtäaikaa saavuttaa vaan etsitään hyvää kompromissia niiden väliltä.

Väitöskirjan päätuloksena esitellään mallinnetulle valintaongelmalle automaattinen heuristinen optimointimenetelmä, joka etsii hyvän kompromissiratkaisun nopeasti. Menetelmä käyttää vain vähän laskentakapasiteettia, jotta algoritmi voidaan ajaa päätelaitteissa, joissa ei ole juurikaan ylimääräistä kapasiteettia käytettävänä. Menetelmä myös ottaa huomioon jatkuvasti muuttuvan verkkoympäristön ja tarvittaessa päivittää ratkaisua tiedonsiirron aikana.

REFERENCES

[1] F. Bari and V. C. M. Leung. Automated network selection in a heterogeneous wireless network environment. *IEEE Network*, 21(1):34–40, 2007.

[2] J. Błażewicz. Selected topics in scheduling theory. In S. Martello, G. Laporte, M. Minoux, and C. Ribeiro, editors, *Surveys in Combinatorial Optimization*, volume 31 of *Annals of Discrete Mathematics*, pages 1–59. Elsevier Science Publishers, 1987.

[3] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

[4] J. K. Cochran, S.-M. Horng, and J. W. Fowler. A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers & Operations Research*, 30(7):1087–1102, 2003.

[5] I. Das and J. E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization. *Structural Optimization*, 14(1):63–69, 1997.

[6] M. Ehrgott. A discussion of scalarization techniques for multiple objective integer programming. *Annals of Operations Research*, 147(1):343–360, 2006.

[7] M. Ehrgott and D. Tenfelde-Podehl. Computation of ideal and Nadir values and implications for their use in MCDM methods. *European Journal of Operational Research*, 151(1):119–139, 2003.

[8] V. Gazis, N. Alonistioti, and L. Merakos. Toward a generic "always best connected" capability in integrated WLAN/UMTS cellular mobile networks (and beyond). *IEEE Wireless Communications*, 12(3):20–29, 2005.

[9] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P. L. Hammer, E. L. Johnson, and B. H. Korte, editors, *Discrete optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287–326. North-Holland Publishing Company, 1979.

[10] J. N. D. Gupta, K. Hennig, and F. Werner. Local search heuristics for two-stage flow shop problems with secondary criterion. *Computers & Operations Research*, 29(2):123–149, 2002.

[11] E. Gustafsson and A. Jonsson. Always best connected. *IEEE Wireless Communications*, 10(1):49–55, 2003.

[12] L. A. Hall. Approximation algorithms for scheduling. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, 1997.

[13] J. Herrmann, J.-M. Proth, and N. Sauer. Heuristics for unrelated machine scheduling with precedence constraints. *European Journal of Operational Research*, 102(3):528–537, 1997.

[14] D. W. Hildum. *Flexibility in a knowledge-based system for solving dynamic resource-constrained scheduling problems*. PhD thesis, Department of Computer Science, University of Massachusetts, 1994.

[15] H. Hoogeveen. Multicriteria scheduling. *European Journal of Operational Research*, 167(3):592–623, 2005.

[16] *ILOG CPLEX 8.0 User's Manual*. ILOG, 2002.

[17] J. D. Landa-Silva, E. K. Burke, and S. Petrovic. An introduction to multiobjective metaheuristics for scheduling and timetabling. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, chapter 4, pages 91–129. Springer, 2004.

[18] J. Y.-T. Leung, editor. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, 2004.

[19] S. T. McCormick and M. L. Pinedo. Scheduling n independent jobs on m uniform machines with both flowtime and makespan objectives: A parametric analysis. *ORSA Journal on Computing*, 7(1):63–77, 1995.

[20] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[21] E. Mokotoff. Parallel machine scheduling problems: A survey. *Asia-Pacific Journal of Operational Research*, 18(2):193–242, 2001.

[22] A. Nagar, J. Haddock, and S. Heragu. Multiple and bicriteria scheduling: A literature survey. *European Journal of Operational Research*, 81(1):88–104, 1995.

[23] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.

[24] I. Ch. Paschalidis and J. N. Tsitsiklis. Congestion-dependent pricing of network services. *IEEE/ACM Transactions on Networking*, 8(2):171–184, 2000.

[25] A. Setämaa, K. Miettinen, M. M. Mäkelä, and J. Vuori. Multiobjective optimization of a transmission link selection problem. Reports of the Department of Mathematical Information Technology, Series B. Scientific Computing, No. B 10/2003. University of Jyväskylä, 2003.

[26] A. Setämaa-Kärkkäinen, K. Miettinen, and J. Vuori. Heuristic for a transmission link selection problem. Reports of the Department of Mathematical Information Technology, Series B. Scientific Computing, No. B 3/2004. University of Jyväskylä, 2004.

[27] A. Setämaa-Kärkkäinen, K. Miettinen, and J. Vuori. New heuristic approach to multiobjective scheduling. In P. Neittaanmäki, T. Rossi, S. Korotov, E. Onate, J. Periaux, and D. Knörzer, editors, *Proceedings of the 4th European Congress on Computational Methods in Applied Sciences and Engineering*, volume II, Jyväskylä, Finland, 2004.

[28] D. B. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62(3):461–474, 1993.

[29] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Application*. John Wiley & Sons, 1986.

[30] V. T'kindt and J.-C. Billaut. Multicriteria scheduling problems. In M. Ehrgott and X. Gandibleux, editors, *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, pages 445–491. Kluwer Academic Publishers, 2002.

[31] V. T'kindt and J.-C. Billaut. *Multicriteria Scheduling: Theory, Models and Algorithms*. Springer-Verlag, 2002.

[32] V. T'kindt, J. N. D. Gupta, and J.-C. Billaut. Two-machine flowshop scheduling with a secondary criterion. *Computers & Operations Research*, 30(4):505–526, 2003.

[33] P. Toth. Optimization engineering techniques for the exact solution of NP-hard combinatorial optimization problems. *European Journal of Operational Research*, 125(2):222–238, 2000.

[34] U. Varshney and R. Jain. Issues in emerging 4G wireless networks. *IEEE Computer*, 34(6):94–96, 2001.

[35] H. R. Weistroffer. Careful usage of pessimistic values is needed in multiple objectives optimization. *Operations Research Letters*, 4(1):23–25, 1985.

[36] A. P. Wierzbicki. The use of reference objectives in multiobjective optimization. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making, Theory and Applications*, pages 468–486. Springer-Verlag, 1980.

[37] A. P. Wierzbicki. A mathematical basis for satisficing decision making. *Mathematical Modelling*, 3(5):391–405, 1982.

[38] A. P. Wierzbicki. Reference point approaches. In T. Gal, T. J. Stewart, and T. Hanne, editors, *Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory, and Applications*, pages 9–1–9–39. Kluwer Academic Publishers, 1999.

[39] C. Yiping and Y. Yuhang. A new 4G architecture providing multimode terminals always best connected services. *IEEE Wireless Communications*, 14(2):36–41, 2007.

ORIGINAL PAPERS

PI

BEST COMPROMISE SOLUTION FOR A NEW MULTIOBJECTIVE SCHEDULING PROBLEM

by

Anne Setämaa-Kärkkäinen, Kaisa Miettinen and Jarkko Vuori 2006

Computers & Operations Research 33(8): 2353–2368

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 33 (2006) 2353–2368
computers & operations research
www.elsevier.com/locate/cor

Best compromise solution for a new multiobjective scheduling problem

Anne Setämaa-Kärkkäinen[a,*], Kaisa Miettinen[b], Jarkko Vuori[a]

[a] *Department of Mathematical Information Technology, PO Box 35 (Agora), FIN-40014, University of Jyväskylä, Finland*
[b] *Helsinki School of Economics, PO Box 1210, FIN-00101 Helsinki, Finland*

Available online 17 March 2005

Abstract

In future wireless networks, a mobile terminal will be able to communicate with a service provider using several network connections. These connections to networks will have different properties and they will be priced separately. In order to minimize the total communication time and the total transmission costs, an automatic method for selecting the network connections is needed. Here, we describe the network connection selection problem and formulate it mathematically. We discuss solving the problem and analyse different multiobjective optimization approaches for it.

Keywords: Multiobjective optimization; Biobjective optimization; Assignment problem; Parallel machine scheduling; Telecommunications; Wireless networks

1. Introduction

Future fourth-generation (4G) wireless networks will offer global roaming across many wireless networks [1]. This means that mobile terminals will be able to use multiple wireless connections to networks simultaneously. For example, data can be sent using a Bluetooth connection while communicating through a GPRS connection. In addition, it will be possible to select a network connection among those offered

* Corresponding author. Tel.: +358 14 260 4905; fax: +358 14 260 2771.
E-mail addresses: annseta@mit.jyu.fi (A. Setämaa-Kärkkäinen), miettine@hkkk.fi (K. Miettinen), jarkko.vuori@jyu.fi (J. Vuori).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.02.006

by different network operators. Thus, a mobile terminal will be able to use any available connection to a network to transfer data [2]. The data may consist of several distinct components, for example, a web page is composed of pictures and text parts, and also short video streams can be compressed into multiple data components. These different components can be transferred using separate network connections. The connections may have very different characteristics, like price and transmission rate, which makes the selection of connections difficult for the user of the mobile terminal. Therefore, an automatic network connection selection method is needed.

Not only the users of mobile terminals but also service operators that use networks of other operators may need automated network selection. Currently, service operators work mainly with only one network operator, but it is possible that in the future a service operator will be able to make agreements with several network operators. Service operators could then provide mobile terminals with a program that would request bids from network operators to transfer data. Based on the answers, the program would select network operators for the mobile terminal to be used in the data transmission.

In many cases, for example, when a web page is to be transferred, the amount of data to be transferred (that is, the size and the number of the components) is known in advance. In these cases, data transmission can be considered as transaction-based communication. The quality of service (QoS) in this kind of transaction-based transmission describes the time it takes to complete a transaction and the packet loss rate [1]. We consider here only the time, because our transactions are time-sensitive and the packet loss rate can be considered to be negligible.

The users of mobile terminals, as well as the service operators, naturally want the best possible quality of service with the lowest possible price. In other words, they want the data to be transferred as fast as possible while the transfer costs are as low as possible. Thus, the objectives of the network connection selection problem that we consider in this paper are to minimize both the time and the costs of the transfer. Minimizing the time and the costs are conflicting objectives since usually the faster the transfer, the more expensive it is. Therefore, we need a compromise solution that guarantees that the transfer is fast enough and the costs are not too high. In this paper, we analyse different multiobjective solution approaches in order to find a method that produces a good compromise solution for our network connection selection problem.

2. Literature review

The *network connection selection* (NCS) problem can be briefly described as follows. A set of available network connections is given with known properties, such as transmission rate and price. Data consisting of distinct components are to be transferred using the network connections. The problem is to select a network connection for each component of the data in such a way that both the time used in transferring and the costs are minimized. The goal is to develop a fast automatic solution method that gives a good compromise between the objectives without any human interaction.

The NCS problem has not been studied before in the literature. It is not a routing problem, where the objective is to find a route or a path from a source to a destination [3], but an assignment problem. In other words, we do not consider the actual routing of the data, but the selection of the network connections to be used in transmission. Inside each network, the routing algorithms handle the actual physical routing of the data from the source to the destination.

The NCS problem resembles three problem types studied previously in the literature. The problems are parallel machine scheduling [4], load balancing [5] and the generalized assignment problem [6]. Parallel machine scheduling is actually a group of problems, whereas the two other problems are more specific and, though they are treated as separate problems, they can also be regarded as parallel machine scheduling problems. We next discuss briefly the differences between these problems.

As already mentioned, *parallel machine scheduling* problems include different kinds of problems. The common feature is that in these problems, a set of jobs has to be processed on a set of machines. The NCS can be seen as a scheduling problem, where machines represent different network connections and each component forms a job that has to be processed on one of the machines. The machines are in this case *uniform parallel machines*, which means that the machines are identical, that is, they perform the same operation, but they have different speeds that do not depend on the job. Each job is processed on one machine, and the goal is to minimize the total costs of processing the jobs and the *makespan*, which is the time when all the jobs are processed.[1]

The *load balancing* problem is a special case of the parallel machine scheduling problem where the jobs arrive on-line, that is, one by one [5]. The traditional objective in load balancing is to minimize the maximum load on the machines. The difference between our NCS problem and the load balancing problem is that in the NCS all the jobs are known in advance, that is, they do not arrive one at a time, and we consider two objectives: in addition to maximum load, which can be interpreted as the time needed to process the jobs, we are also interested in minimizing the costs.

In the *generalized assignment problem*, there is also a set of machines that process jobs but the machines have finite capacities that restrict the assignment of jobs. In other words, we cannot assign more jobs to a machine than the capacity of the machine allows. The objective in the generalized assignment problem is to minimize the cost of the assignments. The main difference between the generalized assignment problem and the parallel machine scheduling problems is that in the latter also the order of the jobs on each machine usually has to be determined. It should be noted that there can be capacity restrictions also in scheduling problems. The difference between the NCS problem and the generalized assignment problem is that in the NCS problem there are no capacity limits on machines (that is, no limits on how many data components can be assigned to each network connection), and in addition to the costs we want to minimize also the time.

In [6], a multiobjective model of the generalized assignment problem is presented where the capacity restrictions are not used as constraints but the aim is to minimize, in addition to the costs, also the amount of capacity each machine needs. The capacity needed for processing a job can be thought of as the time that it takes to process a job. In that case, the NCS problem can be seen as a generalized assignment problem where instead of having a time limit as the constraint we minimize both the costs and the time that it takes to process all the jobs. Though the multiobjective generalized assignment problem has been shortly introduced in [6], we are not aware of any further study regarding it. The algorithms suggested in [6] for the multiobjective generalized assignment problem are an interactive method requiring a human

[1] Using the three field notation introduced in [7] for scheduling problems, the problem can be stated as $Q\|C_{\max}, \sum f_j$ where Q denotes uniform parallel machines, $C_{\max}$ is the makespan and f_j denotes the cost of processing job j.

decision maker and a method producing many Pareto optimal solutions among which the final solution is selected by a decision maker. Because our aim is to develop an automatic solution method, these algorithms cannot be used to solve the NCS problem.

In scheduling, multiple objectives have been mostly considered in the case of single machine problems, and the studies have rarely involved any multiobjective analysis, as emphasized in [8]. In the few papers dealing with multiobjective scheduling, the aim has been either to find all the Pareto optimal solutions to the problem, or a good representation of them (see for example [9,10]) or to minimize the objectives in a predefined lexicographic order of importance (see for example [11,12]). Since our aim is to develop an automatic solution method that produces a good compromise, a method producing many solutions is not useful. In addition, we cannot a priori define which of the objectives is absolutely more important. Therefore, these methods are not applicable to our problem.

Another approach to minimize the makespan (the time) and the costs has been presented in [13]. The algorithm presented is a polynomial-time approximation algorithm that requires upper bounds C and T to be given a priori for the costs and the makespan, respectively. The algorithm gives a solution with costs C and makespan of at most $2T$, if there exists a schedule with costs C and makespan T. A drawback in this kind of an algorithm is the selection of the upper bounds: they should be large enough to get a solution, and small enough to get a good solution. Another disadvantage is that the solution obtained may be unsatisfactory since the makespan may be twice its limit.

In [14], solving a general multiobjective scheduling problem has been divided into three subproblems: modelling the problem, taking into account the objectives and scheduling. The first phase, modelling the problem, means defining the scheduling problem and the objectives to be taken into account. In the second phase, the problem is to decide how the conflicting objectives will be taken into account, that is, how a compromise solution between the objectives can be obtained. In the last phase, the actual scheduling problem is solved and a solution to the multiobjective scheduling problem is obtained.

In this paper, we concentrate on modelling the problem and taking into account the objectives. Thus, we are not developing an actual algorithm for solving the network connection selection problem, but we aim at finding a multiobjective optimization method that produces a good compromise between the conflicting objectives. The ultimate goal of our study is to develop a fast automatic solution method that is transparent to the user of the mobile terminal. This means that the method solves the problem without any human interaction. This goal should be borne in mind while selecting the multiobjective optimization method. In order to be able to select an appropriate method, we have to know the nature of the problem. For that purpose, we have studied different examples, of which we present three here. The three examples represent different kinds of problem instances that may occur.

In the next section, we formulate our problem mathematically. In Section 4, we discuss multiobjective optimization and define some concepts. In Section 5, we present three examples to illustrate the problem settings. In the following section, we discuss three different multiobjective optimization methods and show the results of solving the examples using them. Finally, we draw conclusions in Section 7.

3. Model

As mentioned in the introduction, our model to be considered in this paper is designed for transaction-based data communication. Transactions are considered to be time-sensitive and the content of the

transaction is predefined. Before the transaction, the mobile terminal requests the properties of the available network connections, such as the transmission rate and the price. The time used in the transaction is considered to be short. Therefore, the properties of the network connections can be considered to be fixed during the transaction. Lengthy transactions have to be divided into smaller parts if the properties vary much in time. In other words, we assume that the transaction consists of fixed components that are transferred separately, and the properties of the available network connections are known and fixed in the model.

Let us now assume that there are m connections to networks available and n components of a transaction have to be transferred using the connections. Let x_{ij} be a binary variable such that $x_{ij}=1$, when component j is transferred using connection i, otherwise $x_{ij}=0$, $i=1,\ldots,m$, $j=1,\ldots,n$. A coefficient, *duration*, d_{ij} denotes the time needed to transfer component j through connection i. The duration is calculated using the minimum guaranteed rate of the connection given by the operator of the connection. This guarantees that, although we cannot define the exact transmission rate, the actual time used in the transfer is never longer than the duration d_{ij}. Another coefficient c_{ij} denotes the cost of transferring component j through connection i.

In order to maximize the quality of service of the time-sensitive transaction, we want to minimize both the time used in transferring the components and the costs. The time used in transferring components through connection i is $\sum_{j=1}^{n} d_{ij}x_{ij}$. Then, the total time used in transferring all the components is the maximum of the times used on each connection, that is $\max_{i=1,\ldots,m}\sum_{j=1}^{n} d_{ij}x_{ij}$. The cost of transferring component j can be written as a sum $\sum_{i=1}^{m} c_{ij}x_{ij}$, and then the total costs can be summed up over all the components, $\sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}x_{ij}$.

Now, the network connection selection problem can be mathematically formulated as follows:

$$\begin{aligned}
\min \quad & \max_{i=1,\ldots,m} \sum_{j=1}^{n} d_{ij}x_{ij} \quad \text{and} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{i=1}^{m} x_{ij} = 1 \quad \text{for all } j=1,\ldots,n, \\
& x_{ij} \in \{0,1\} \quad \text{for all } i=1,\ldots,m \text{ and } j=1,\ldots,n.
\end{aligned} \tag{1}$$

Here, constraints (1) require that each component is transferred using exactly one connection.

When the problem is solved, the solution tells which network connection is used to transfer each component. It does not pay attention to the order in which the components are transferred on each network connection because neither the time used in transfers nor the total costs depend on the order of the components. The components can be ordered, for example, in the increasing order of the component sizes on each network connection.

In order to avoid nonlinearity in the first objective function, we reformulate the problem with the help of an additional variable y. We define m new constraints:

$$\sum_{j=1}^{n} d_{ij}x_{ij} \leqslant y \quad \text{for all } i=1,\ldots,m,$$

and define the first objective function to be y. Thus, an equivalent model of the problem with linear objective functions and constraints is

$$\begin{aligned}
\min \quad & f_1(x) = y \quad \text{and} \quad f_2(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{i=1}^{m} x_{ij} = 1 \quad \text{for all } j = 1, \ldots, n, \\
& \sum_{j=1}^{n} d_{ij} x_{ij} \leqslant y \quad \text{for all } i = 1, \ldots, m, \\
& x_{ij} \in \{0, 1\} \quad \text{for all } i = 1, \ldots, m \text{ and } j = 1, \ldots, n.
\end{aligned}$$

It should be noted that in these models we have not specified how the costs are formed. The cost c_{ij} may be linearly dependent on the size of the component or it may consist of a fixed cost and variable costs.

4. Multiobjective optimization

We consider a general optimization problem with two objectives, where we want to minimize functions $f_1(x)$ and $f_2(x)$ subject to a constraint $x \in S$. We denote the vector of objective functions by $F(x) = (f_1(x), f_2(x))^{\mathrm{T}}$. The vector $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ is called a *decision vector* and $S \subset \mathbb{R}^n$ is the *feasible region*. The feasible region is formed by *constraint functions*. The image of the feasible region, $Z = F(S)$, is called the *feasible objective region*. Vectors belonging to the feasible objective region Z are called *objective vectors* and they are denoted by $F(x) \in \mathbb{R}^2$.

We want to minimize simultaneously both objective functions. Generally, it is not possible to find a solution in which both objective functions attain minimum values. This means that the objective functions are conflicting. Besides, the feasible objective region Z is only partially ordered. In other words, we cannot compare all the objective vectors mathematically. For example, we cannot distinguish which is a better objective vector, $(1, 5)^{\mathrm{T}}$ or $(5, 1)^{\mathrm{T}}$. However, we can say that $(1, 5)^{\mathrm{T}}$ is better than $(2, 5)^{\mathrm{T}}$ or $(2, 7)^{\mathrm{T}}$. This leads us to the concept of Pareto optimality. A decision vector $x^* \in S$ and the corresponding objective vector $F(x^*)$ are *Pareto optimal* if there does not exist another decision vector $x \in S$ such that $f_i(x) \leqslant f_i(x^*)$ for $i = 1, 2$ and $f_j(x) < f_j(x^*)$ for at least one j [15]. A decision vector $x^* \in S$ and the corresponding objective vector $F(x^*)$ are *weakly Pareto optimal* if there does not exist another decision vector $x \in S$ such that $f_i(x) < f_i(x^*)$ for $i = 1, 2$. A set containing all the Pareto optimal solutions of the problem is called the *Pareto optimal set*.

Now the solution we are looking for is a Pareto optimal solution. This guarantees that we cannot improve any of the objective function values of the solution without deteriorating the other objective function value. Which of the Pareto optimal solutions is the best, that is, is chosen to be the final solution to the problem, depends usually on a decision maker. A *decision maker* is a person who knows the problem well and can express preference relations between the conflicting objectives. The decision maker can, for example, tell objective function values that are satisfying to her/him. These objective function values are called *aspiration levels*. A vector containing the aspiration levels is called a *reference point*. However, it

is not always possible to use a decision maker, as in our case, when the goal is to automate the solution process in a way that it is transparent to the user. Then, we need a multiobjective optimization method that gives a good compromise between the objectives without any information from a decision maker.

Let us define a few more concepts involving multiobjective optimization. An objective vector containing the minimum value of each objective function is called an *ideal objective vector*, z^*. The ideal objective vector is usually infeasible. Otherwise, it would be the optimal solution to the multiobjective optimization problem. The components z_i^* of the ideal objective vector are obtained by minimizing each of the objective functions f_i separately subject to the original constraints. A *utopian objective vector* z^{**} is an infeasible objective vector that can be formed from the ideal objective vector by subtracting a relatively small scalar $\varepsilon_i > 0$ from each component z_i^* of the ideal objective vector. Thus, we have

$$z_i^{**} = z_i^* - \varepsilon_i \quad \text{for } i = 1, 2.$$

From the ideal objective vector we get the lower bound of the Pareto optimal set for each objective function. The upper bounds of the Pareto optimal set are the components of a *nadir objective vector* z^{nad}. In the case of two objective functions, the nadir objective vector can be obtained as follows. The first component of the nadir objective vector is the value of the first objective function $f_1(x_1)$ in the point $x_1 \in S$ where the second objective function attains its minimal value, and the second component is the value of the second objective function $f_2(x_2)$ in the point $x_2 \in S$ where the first objective function attains its minimal value [16].

Multiobjective optimization problems are usually solved by scalarization [15]. *Scalarization* means converting the problem with multiple objectives into a single objective optimization problem or a family of single objective optimization problems. When a multiobjective optimization problem has been scalarized, methods developed for single objective optimization can be used for solving the problem. The objective function of the single objective problem is called a *scalarizing function*.

We use CPLEX 8.0 to solve the single objective problems after scalarization. CPLEX is software that can solve linearly constrained optimization problems where the objective function is linear or quadratic [17]. The variables may be continuous or integer-valued. CPLEX uses a branch-and-cut algorithm [18] to solve (mixed) integer programming problems. It should be noted that CPLEX finds solutions that are within a certain percentage of the optimal solution, not necessarily the optimal solution. This may cause, for example, weakly Pareto optimal solutions to appear even in cases where there theoretically should not be weakly Pareto optimal solutions.

5. Examples

We have studied the network connection selection problem using a variety of examples. In this section, we discuss three examples that are representatives of different kinds of problem instances that may occur. The examples are used to illustrate the settings of the network connection selection problem. All the examples are reality based since the transmission rates and prices are estimates for future rates and prices and the components form in each example a real web page. Since we cannot predict exactly how the operators will price the network connections, we have used different kinds of pricing models. In the first two examples, the pricing is linear in the size of the component, and in the third example the price is calculated using a more elaborate formula.

5.1. Brief descriptions of the examples

The first example contains three network connections and ten components. It is a small instance when compared to reality. The connections to be considered are a GPRS connection with a rate of 14 400 bits per second, an EDGE connection with a rate of 115 200 bits per second and a UMTS connection with a rate of 384 000 bits per second. The prices are 8×10^{-8}, 7×10^{-7} and 3×10^{-6} euros per bit, respectively. The sizes of the components are 100, 1200, 400, 36 000, 23 000, 600, 300, 3400, 5600 and 9800 bytes.

The second example contains five network connections and 15 components. Three of the connections are the same as in the first example and two connections are new. The first new connection has the same rate as the first connection in the first example, but the price is a bit higher. The second new connection has the same rate as the second connection of the first example, but the price is a bit lower. These pairs of connections represent competing operators. Thus, the connections of the example are the following: two GPRS connections with a rate of 14 400 bits per second, two EDGE connections with a rate of 115 200 bits per second and a UMTS connection with a rate of 384 000 bits per second. The prices of the connections are 8×10^{-8}, 8.2×10^{-8}, 7×10^{-7}, 6.8×10^{-7} and 3×10^{-6} euros per bit, respectively. The sizes of the components are 110, 300, 480, 5300, 6100, 950, 960, 2300, 2500, 3100, 48 000, 42 000, 30 000, 500 and 12 000 bytes.

The third example includes five network connections and 14 components. The connections are three GSM connections with rates of 14 400, 32 000 and 56 000 bits per second, and two UMTS connections with rates of 56 000 and 384 000 bits per second. The cost of transferring a component is now calculated using the formula

$$c_{ij} = K_0 + K_1 \exp(K_2 \cdot \text{rate}(i)) \cdot \text{size}(j), \tag{2}$$

where rate(i) is the rate of connection i (bits per second), size(j) is the size of component j in bytes and K_0, K_1 and K_2 are connection-specific coefficients. The coefficients K_0, K_1 and K_2 are 9.0, 0.31 and 0.000043, respectively, for the GSM connections, and for the UMTS connections the coefficients are 6.0, 8.8 and 0.0000017, respectively. The cost given by formula (2) has to be multiplied by 10^{-5} in order to get the cost in euros. The sizes of the components are 62 030, 102, 524, 804, 320, 700, 685, 564, 4841, 53, 44, 725, 1251 and 1063 bytes.

As already mentioned, these three examples are representatives of different kinds of problem instances. The examples were selected to be presented here because their Pareto optimal sets are different. The other problem instances studied have similar forms of the Pareto optimal set as these examples. Before we present the Pareto optimal sets of the examples, we briefly discuss scaling the objective functions, which we need later when we consider solving the problem.

We can scale the objective functions to be of equal magnitude using the ideal and the nadir objective vectors. The following scaling converts the objective function values between 0 and 1: instead of $f_i(x)$ we use

$$\frac{f_i(x) - z_i^*}{z_i^{\text{nad}} - z_i^*}. \tag{3}$$

The ideal and the nadir objective vectors are $(1.3, 0.051456)^{\text{T}}$ and $(44.66667, 1.58749)^{\text{T}}$, respectively, in the first example, $(2.083333, 0.098944)^{\text{T}}$ and $(85.88889, 2.54768)^{\text{T}}$ in the second example and $(1.29229, 0.425664)^{\text{T}}$ and $(41.0, 10.7225)^{\text{T}}$ in the third example.

Calculating ideal and nadir objective vectors is time-consuming, since we need to solve two single objective optimization problems for each problem instance. Therefore, we would prefer a method for scaling that would not need much computation. (Remember that we aim at developing a fast automatic solution method.) That is why we have also used approximations of the ideal and the nadir objective vectors in the scaling. The ideal objective vector is approximated by a utopian objective vector $(0, 0)^{\mathrm{T}}$ in each example. This vector is guaranteed to be infeasible because no data can be sent without any costs and without using any time. The nadir objective vector can be estimated for each example as follows. The first component of the vector is the objective function value $f_1(x)$ related to the solution x where the slowest connection is used to transfer all the components. The second component of the vector is the objective function value $f_2(x)$ related to the solution x where the most expensive connection is used to transfer all the components. These estimates are larger than or equal to the components of the real nadir objective vector. The estimates of the nadir objective vector are $(44.6667, 1.9296)^{\mathrm{T}}$ for the first example, $(85.8889, 3.7104)^{\mathrm{T}}$ for the second example and $(40.9478, 12.4601)^{\mathrm{T}}$ for the third example. It should be noted that the estimate of the second component of the nadir objective vector may be very rough.

5.2. On Pareto optimal sets of the examples

Since the network connection selection problem has not been studied before, we first need to know the settings of the problem before we can decide what kind of a solution would be a good compromise between the objectives. In order to study the settings, we have formed different examples and calculated approximations of their Pareto optimal sets.

We show here approximations of the Pareto optimal sets of the examples presented in Section 5.1. The approximations are formed using the method of weighted metrics. The method of weighted metrics is a family of methods for generating Pareto optimal solutions [15]. In the method, the distance between some reference point and the feasible objective region is minimized. Here, the ideal objective vector z^* is used as the reference point. The distance can be measured using weighted L_p-metrics, of which we use the weighted Tchebycheff metric (L_∞). The *weighted Tchebycheff problem* to be solved in the method is the following when we have two objective functions:

$$\begin{aligned} \min \quad & \max_{i=1,2} \{w_i | f_i(x) - z_i^*|\} \\ \text{s.t.} \quad & x \in S. \end{aligned} \tag{4}$$

The weighting coefficients w_1 and w_2 must be greater than or equal to zero and sum up to one, that is, $w_1 + w_2 = 1$. When the real ideal objective vector is known, the absolute value signs can be left out.

The solution to the weighted Tchebycheff problem (4) is weakly Pareto optimal if all the weighting coefficients are positive, and the method can find every Pareto optimal solution [15]. If we measure the distance by an augmented weighted Tchebycheff metric, no weakly Pareto optimal solutions are produced [15]. The *augmented weighted Tchebycheff problem* is

$$\begin{aligned} \min \quad & \max_{i=1,2} \{w_i (f_i(x) - z_i^*)\} + \rho \sum_{i=1}^{2} (f_i(x) - z_i^*) \\ \text{s.t.} \quad & x \in S, \end{aligned} \tag{5}$$

where the *augmentation coefficient* $\rho > 0$ is a sufficiently small scalar. We use the value 0.001 for the augmentation coefficient.

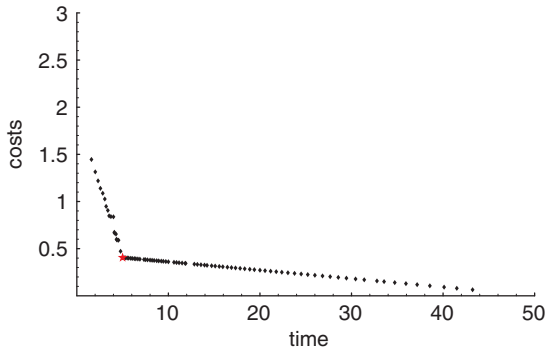


Fig. 1. Approximation of the Pareto optimal set of the first example.

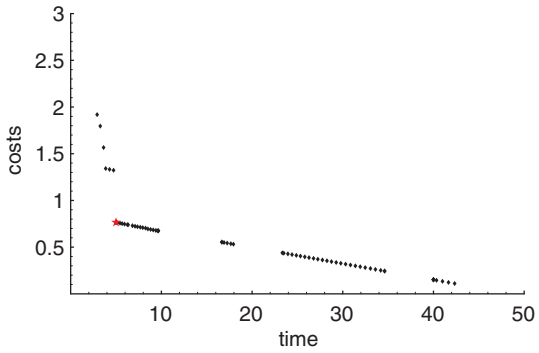


Fig. 2. Approximation of the Pareto optimal set of the second example.

By altering the weighting coefficients w_1 and w_2, we get different Pareto optimal solutions. Since there are only two coefficients that sum up to one, it is enough to alter only one coefficient w_1 and the other coefficient is then automatically set to $w_2 = 1 - w_1$.

In order to make the objective functions similar in magnitude, the objective functions are scaled using formula (3). It should be noted that the scaling has to be applied also to the ideal objective vector in (4) and (5). Thus, after scaling, the ideal objective vector is $(0, 0)^{\mathrm{T}}$.

The approximations of the Pareto optimal sets of the three examples presented in Section 5.1 are depicted in Figs. 1, 2, and 3, respectively. The coordinate axis *time* represents the value of the first objective function f_1 and the coordinate axis *costs* represents the value of the second objective function f_2. The solutions depicted in the figures were obtained by giving the weighting coefficient w_1 99 different values: from 0.01 to 0.99 with a step size 0.01. This amount of different values for the weighting coefficient is enough since we are only interested in the form of the Pareto optimal set. It should be noted that since we are dealing with integer variables, the Pareto optimal sets are not connected.

Fig. 1 shows that the Pareto optimal set of the first example is approximately piecewise linear. We can see a turning point in the Pareto optimal set. The solution in the turning point will be referred to as the *knee solution*, and it is depicted by a star in the figure. This knee solution is a logical choice for the

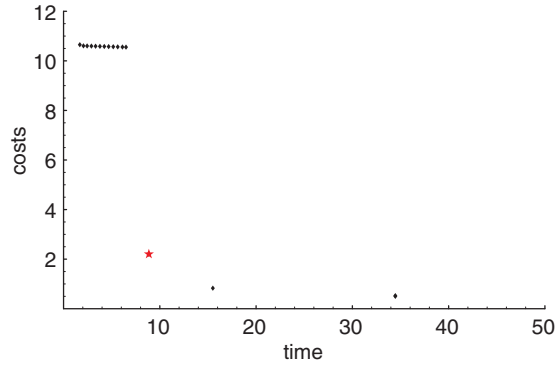


Fig. 3. Approximation of the Pareto optimal set of the third example.

final solution since the gain in costs would be relatively small when compared to the loss in time when we consider solutions to the right of the knee solution and vice versa when we consider solutions to the left of the knee solution. The knee solution is a solution with objective function values $f_1(x) = 5.0$ and $f_2(x) = 0.4056$.

The Pareto optimal set of the second example depicted in Fig. 2 looks a bit different when compared to the first example. There are large "jumps" in the approximation of the Pareto optimal set. However, there is still a natural choice for the final solution. The solution is represented by a star in the figure and its objective function values are $f_1(x) = 5.0$ and $f_2(x) = 0.765648$.

The approximation of the Pareto optimal set of the third example in Fig. 3 is more sparse. The sparsity may be caused by a dominant component in the example whose size is larger than the sum of the other components' sizes. Nevertheless, there is a good candidate for the final solution. The solution is again depicted by a star in the figure, and it has objective function values $f_1(x) = 8.86143$ and $f_2(x) = 2.2052$.

To sum up, in each of these examples there is a solution that would be a good compromise between the objectives. We refer to these solutions as the knee solutions. Computing the whole Pareto optimal set (or a part of it) is time-consuming and, thus, it cannot be done in the automatic method that we are developing for the problem. In other words, we cannot search for the knee solution by calculating many Pareto optimal solutions, but we have to find a multiobjective optimization method that gives one solution that is close enough to the knee solution to be accepted as a good compromise.

6. Solution methods

In this section, we describe three multiobjective optimization methods and analyse their performance on the examples presented in the previous section. We try to find a method that gives a solution near the knee solution in these examples, and also work similarly on other problem instances. Thus, the method should not be too sensitive to the selection of method-dependent parameters but it should give good results for any problem instance.

6.1. Weighting method

Often the first thing that comes to mind when a multiobjective optimization problem is encountered is to give weights to different objectives and then minimize the weighted sum of the objectives. This is

Table 1
Weighting coefficient values producing the knee solution

Example	Real ideal and nadir	Approximated ideal and nadir
1	0.25–0.85	0.2–0.85
2	0.65–0.9	0.55–0.9
3	0.45–0.8	0.45–0.75

a simple approach that has some drawbacks, such as only a part of the Pareto optimal solutions can be found with this method if the problem is nonconvex [15]. A positive aspect is that the solution is Pareto optimal if the weighting coefficients are positive [15].

In the weighting method, the following scalarizing function is minimized:

$$\sum_{i=1}^{2} w_i f_i(x), \tag{6}$$

where the weighting coefficients are greater than or equal to zero and sum up to one, that is $w_1 + w_2 = 1$. Again, since we have only two coefficients, we can alter the first coefficient w_1 and calculate the second as $w_2 = 1 - w_1$. The objective functions f_1 and f_2 are scaled to be of equal magnitude using formula (3).

The knee solutions are obtained with different values of weighting coefficients in each example. We used 19 different values for the coefficient w_1: from 0.05 to 0.95 with a step size 0.05. In the first example, the weighting coefficient values w_1 that produce the knee solution are located between 0.25 and 0.85. In the second example, the corresponding values are between 0.65 and 0.9, and in the third example, the values are between 0.45 and 0.8.

When we scale the objective functions using the approximated ideal and nadir objective vectors, the ranges of the weighting coefficient values do not change much. The ranges are 0.2–0.85 in the first example, 0.55–0.9 in the second example and 0.45–0.75 in the third example. The ranges are collected in Table 1.

Let us add that, in the second example, values between 0.45 and 0.6 produce a solution very close to the knee solution. Equally, when the approximated ideal and nadir objective vectors are used in the second example, values between 0.35 and 0.5 give the same solution near the knee solution.

As we can see, the ranges of the weighting coefficient values that produce the knee solution are different in each example. Therefore, it is difficult to choose an appropriate value for the coefficient in such a way that the method would give a good solution to every problem instance. Another disadvantage in this method is that it is not clear how the weighting coefficient values affect the solution: a small change in the coefficient value may make the solution very different [15]. Therefore, we do not consider this method to be appropriate for the network connection selection problem.

6.2. Neutral compromise solution

In [19], a method for generating a so-called neutral compromise solution is described. *A neutral compromise solution* is a solution that is located somewhere in the middle of the Pareto optimal set. When there are no preferences between conflicting objectives available, a neutral compromise solution may be a good candidate for the final solution. A neutral compromise solution is obtained by solving

Table 2
Neutral compromise solutions

Example	Real ideal and nadir	Approximated ideal and nadir	Knee solution
1	$(10.0, 0.36096)^{\mathrm{T}}$	$(8.61111, 0.37336)^{\mathrm{T}}$	$(5.0, 0.4056)^{\mathrm{T}}$
2	$(17.2556, 0.543441)^{\mathrm{T}}$	$(9.61111, 0.675221)^{\mathrm{T}}$	$(5.0, 0.765648)^{\mathrm{T}}$
3	$(8.86143, 2.2052)^{\mathrm{T}}$	$(8.86143, 2.2052)^{\mathrm{T}}$	$(8.86143, 2.2052)^{\mathrm{T}}$

the problem

$$\begin{aligned} \min \quad & \max_{i=1,2} \frac{f_i(x) - z_i^{\mathrm{mid}}}{z_i^{\mathrm{nad}} - z_i^*} + \rho \sum_{i=1}^{2} \frac{f_i(x) - z_i^{\mathrm{mid}}}{z_i^{\mathrm{nad}} - z_i^*} \\ \text{s.t.} \quad & x \in S, \end{aligned} \tag{7}$$

where the reference point z^{mid} is located in the middle of the ranges of the objective functions in the Pareto optimal set, that is,

$$z_i^{\mathrm{mid}} = \frac{z_i^{\mathrm{nad}} + z_i^*}{2}, \quad i = 1, 2, \tag{8}$$

and the augmentation coefficient ρ is a small positive scalar. The problem is a modification of the augmented weighted Tchebycheff problem (5), and its idea resembles the idea of problem (5). However, here the distance measure is more general and it guarantees that the solution is Pareto optimal regardless of the location of the reference point [15,19]. In addition, the reference point is now different, and we do not have to define any artificial parameters as weighting coefficients. It should also be noted that the objective functions are scaled to be of equal magnitude already in (7), so we do not use formula (3) to scale the objective functions. The augmentation term guarantees again that the solution is Pareto optimal.

The augmentation coefficient was given the value 0.001. The results for the examples are the following. In the first example, the neutral compromise solution is $(10.0, 0.36096)^{\mathrm{T}}$ when the real ideal and nadir objective vectors are used and $(8.61111, 0.37336)^{\mathrm{T}}$ when the approximated ideal and nadir objective vectors are used. The neutral compromise solutions of the second example are $(17.2556, 0.543441)^{\mathrm{T}}$ and $(9.61111, 0.675221)^{\mathrm{T}}$ when the real/approximated ideal and nadir objective vectors are used, respectively. In the third example, the neutral compromise solution is the knee solution $(8.86143, 2.2052)^{\mathrm{T}}$ in both the cases. These results and the knee solutions are also presented in Table 2.

The neutral compromise solution is the knee solution only in the third example. In addition, the neutral solutions are quite far from the knee solution in the other examples. Therefore, though this method has the advantage of not needing any problem-specific parameters, it cannot achieve a good compromise and cannot be used for solving the network connection selection problem in general.

6.3. Achievement scalarizing function

The following achievement scalarizing function to be minimized is a modification of the method that produces a neutral compromise solution:

$$\max_{i=1,2} w_i \frac{f_i(x) - z_i^{\mathrm{mid}}}{z_i^{\mathrm{nad}} - z_i^*} + \rho \sum_{i=1}^{2} w_i \frac{f_i(x) - z_i^{\mathrm{mid}}}{z_i^{\mathrm{nad}} - z_i^*}. \tag{9}$$

Table 3
Solutions to the achievement scalarizing function

Example	Real ideal and nadir	Approximated ideal and nadir	Knee solution
1	$(4.77778, 0.472128)^T$	$(4.77778, 0.472128)^T$	$(5.0, 0.4056)^T$
2	$(5.34444, 0.758617)^T$	$(5.0, 0.76639)^T$	$(5.0, 0.765648)^T$
3	$(8.86143, 2.2052)^T$	$(8.86143, 2.2052)^T$	$(8.86143, 2.2052)^T$

The reference point is the same middle point (8) as in the method producing a neutral compromise solution, and ρ is again a small positive scalar. The ratio of the positive weighting coefficients w_1 and w_2 represents here the rate at which the user is willing to trade off values of the objective functions. It should again be noted that the objective functions are scaled to be of equal magnitude in (9), so we do not use formula (3) to scale the objective functions. The solution to the achievement scalarizing function (9) is Pareto optimal [15].

The augmentation coefficient ρ was given the value 0.001 as before. We used the values $w_1 = 1$ and $w_2 = 2$ for the weighting coefficients. It could be possible to allow the user to change these coefficients in the setting of the mobile terminal. This adjustment would be separate from the actual optimization, so the user would not act as a decision maker. We return to this subject after we have discussed the results obtained using this method.

The results of using this method to solve the examples are the following. The solution of the first example is $(4.77778, 0.472128)^T$ when using both the real and the approximated ideal and nadir objective vectors. In the second example, the solutions are $(5.34444, 0.758617)^T$ and $(5.0, 0.76639)^T$ when the real/approximated ideal and nadir objective vectors are used, respectively. The solution of the third example is $(8.86143, 2.2052)^T$ in both the cases. These results and the knee solutions are also presented in Table 3.

The solution of the third example is the knee solution, and in the two other examples the solutions are close to the knee solutions. (Actually, in the second example, when the approximated ideal and nadir objective vectors are used, the solution is in practice the knee solution: the difference between the solutions is due to computational reasons, see the end of Section 4.) Similar results are also obtained for other instances of the network connection selection problem that we have studied. Therefore, this method seems suitable for the purpose of developing an automatic solution method for the network connection selection method.

The adjustment of the weighting coefficients may give rise to some arguments. If the user of the mobile terminal is allowed to adjust the weighting coefficients in the achievement scalarizing function (9), one might ask why not use the weighting method and let the user change the weighting coefficients. The reason is that the weighting method is very sensitive to the selection of the weighting coefficients, and the solution obtained may be very far from the solution that would actually be acceptable for the user. The achievement scalarizing function is not that sensitive to the weighting coefficients because the scalarizing function has also other parameters that affect the solution, such as the reference point, which is the middle point in this case. Therefore, the selection of the weighting coefficients is not that crucial in this method.

7. Conclusions

We have introduced a new problem called the network connection selection problem. The problem was formulated as a multiobjective scheduling problem where the objective is to minimize both the costs and the time. This kind of a problem has not been studied earlier in the literature.

In this paper, we have concentrated on selecting a multiobjective optimization method for the network connection selection problem. In order to be able to select the method, we need information on the nature of the problem. For that reason we have studied different examples, of which we have presented three here. The three examples represent different kinds of problem instances that may occur. In each of the three examples, there is a solution, the knee solution, that would be a natural choice for the final solution, that is, the best compromise between the objectives. The problem is how to find the knee solution fast. Our goal was to find a robust method that could find the knee solution for any example.

Based on the results obtained, we shall continue using the achievement scalarizing function presented in Section 6.3. However, we will also bear in mind the other results presented in this paper when we continue our research.

We have now resolved two of the three subproblems involved in solving a multiobjective problem: we have modelled the problem and decided how to take into account the objectives. In other words, we have selected a scalarizing function that converts the multiobjective optimization problem into a single objective optimization problem. The next task is to consider how the single objective problem should be solved. Because we aim at solving the problem in real time, the method has to be fast. This means that we have to develop a heuristic method that produces a good solution fast, though we might not be able to guarantee that the real optimum is attained. In a mobile terminal, there is not much computational capacity available that could be used in this kind of optimization, so the method also has to use only little memory. In other words, we will develop a heuristic that uses the achievement scalarizing function to obtain fast a good compromise solution to the problem.

Our problem setting relates to transaction-based data communication, where the transaction times are short. This allows us to assume that the environment is static during each transaction. However, if the transaction is lengthy, the properties of the network connections, such as the transmission rate, may change. In future, we will also consider longer transactions. Then the transaction needs to be divided into smaller parts or the possibly changing environment needs to be taken into account in another way.

So far, we have also assumed that the components can be transferred in any order. However, there may be precedence constraints between components that require some components to be transferred first. In future, we shall take the possible precedence constraints into consideration. In that case, we have to take into account the order of the components in the model.

Acknowledgements

This research was supported by Agora Center at the University of Jyväskylä and the GETA graduate school. The authors also wish to thank Dr. Marko M. Mäkelä for helpful comments.

References

[1] Varshney U, Jain R. Issues in emerging 4G wireless networks. IEEE Computer 2001;34(6):94–6.

[2] Gustafsson E, Jonsson A. Always best connected. IEEE Wireless Communications 2003;10(1):49–55.
[3] Huitema C. Routing in the internet. New York: Prentice-Hall; 1995.
[4] Leung JY-T. editor. Handbook of scheduling: algorithms, models, and performance analysis, Boca Raton, FL: CRC Press; 2004.
[5] Avidor A, Azar Y, Sgall J. Ancient and new algorithms for load balancing in the l_p norm. Algorithmica 2001;29(3): 422–41.
[6] Värbrand P. Generalized assignment type problems: models and solution procedures. Ph.D. thesis. Department of Mathematics, Linköping University, Sweden; 1988.
[7] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Hammer PL, Johnson EL, Korte BH, editors. Discrete optimization II, Annals of discrete mathematics, vol. 5. Amsterdam: North-Holland; 1979. p. 287–326.
[8] T'kindt V, Billaut J-C. Multicriteria scheduling problems. In: Ehrgott M, Gandibleux X, editors. Multiple criteria optimization: state of the art annotated bibliographic surveys. Dordrecht: Kluwer Academic Publishers; 2002.
[9] Cochran JK, Horng S-M, Fowler JW. A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. Computers & Operations Research 2003;30(7):1087–102.
[10] McCormick ST, Pinedo ML. Scheduling n independent jobs on m uniform machines with both flowtime and makespan objectives: a parametric analysis. ORSA Journal on Computing 1995;7(1):63–77.
[11] Gupta JND, Hennig K, Werner F. Local search heuristics for two-stage flow shop problems with secondary criterion. Computers & Operations Research 2002;29(2):123–49.
[12] T'kindt V, Gupta JND, Billaut J-C. Two-machine flowshop scheduling with a secondary criterion. Computers & Operations Research 2003;30(4):505–26.
[13] Shmoys DB, Tardos É. An approximation algorithm for the generalized assignment problem. Mathematical Programming 1993;62(3):461–74.
[14] T'kindt V, Billaut J-C. Multicriteria scheduling: theory, models and algorithms. Berlin: Springer; 2002.
[15] Miettinen K. Nonlinear multiobjective optimization. Dordrecht: Kluwer Academic Publishers; 1999.
[16] Ehrgott M, Tenfelde-Podehl D. Computation of ideal and Nadir values and implications for their use in MCDM methods. European Journal of Operational Research 2003;151(1):119–39.
[17] ILOG CPLEX 8.0 User's manual. ILOG; 2002.
[18] Nemhauser GL, Wolsey LA. Integer and combinatorial optimization. New York: Wiley; 1988.
[19] Wierzbicki AP. Reference point approaches. In: Gal T, Stewart TJ, Hanne T, editors. Multicriteria decision making: advances in MCDM models, algorithms, theory, and applications. Dordrecht: Kluwer Academic Publishers; 1999. p. 9-1–9-39.

PII

HEURISTIC FOR A NEW MULTIOBJECTIVE SCHEDULING PROBLEM

by

Anne Setämaa-Kärkkäinen, Kaisa Miettinen and Jarkko Vuori 2007

Optimization Letters 1(3): 213–225

Optimization Letters (2007) 1:213–225
DOI 10.1007/s11590-006-0020-7

ORIGINAL PAPER

Heuristic for a new multiobjective scheduling problem

Anne Setämaa-Kärkkäinen · Kaisa Miettinen · Jarkko Vuori

Received: 19 May 2006 / Accepted: 29 May 2006 /
Published online: 15 August 2006

Abstract We consider a telecommunication problem in which the objective is to schedule data transmission to be as fast and as cheap as possible. The main characteristic and restriction in solving this multiobjective optimization problem is the very limited computational capacity available. We describe a simple but efficient local search heuristic to solve this problem and provide some encouraging numerical test results. They demonstrate that we can develop a computationally inexpensive heuristic without sacrificing too much in the solution quality.

Keywords Heuristics · Parallel machine scheduling · Biobjective optimization · Combinatorial optimization · Telecommunications

1 Introduction

Future fourth generation (4G) wireless networks will enable global roaming across many wireless networks [18]. This means that mobile terminals will be able to use multiple wireless connections to networks simultaneously [5]. This

A. Setämaa-Kärkkäinen (✉)
Department of Mathematical Information Technology, University of Jyväskylä,
P.O. Box 35 (Agora), 40014 Jyväskylä, Finland
e-mail: annseta@mit.jyu.fi

K. Miettinen
Helsinki School of Economics, P.O. Box 1210, 00101 Helsinki, Finland
e-mail: miettine@hse.fi

J. Vuori
EVTEK University of Applied Sciences, Vanha maantie 6, 02650 Espoo Finland

provides valuable new potential because data to be transferred often consists of several distinct components. A web page, for example, is composed of multiple pictures and text parts. These different components can be transferred using distinct network connections. The connections may have very different characteristics, like price and transmission rate, which makes the wise selection of connections difficult for the user of the mobile terminal. Therefore, an automatic network connection selection method is needed.

The *network connection selection problem* [14] can be described as follows: A set of available network connections is given with known properties, such as transmission rate and price. Data consisting of distinct components is to be transferred using the network connections. The problem is to select a network connection for each component of the data in such a way that both the time used in transferring and the costs are minimized. This objective is chosen because the users of mobile terminals naturally want the data to be transferred as fast as possible while keeping the costs as low as possible. Usually, the faster the transfer is, the more expensive it is. Therefore, we need to find a compromise solution in which the transfer is fast enough while the costs are not too high.

Our goal is to develop a fast automatic solution method that gives a good compromise between the conflicting objectives. The method should be fast, because otherwise the usability of the mobile terminal impairs. In addition, the method should use little computational capacity because the capacity of mobile terminals is limited.

The network connection selection problem can be seen as a uniform parallel machine scheduling problem, where the goal is to minimize the makespan and the total costs of processing the jobs [14]. (Connections to other problem types are discussed in [14].) Using the three field notation introduced in [4], the problem can be stated as $Q||C_{\max}, \sum f_j$ where f_j denotes the cost of processing job j. The uniform parallel machine scheduling problem with the objective of minimizing the makespan is NP-hard [1]. Therefore, also the problem with the objectives of minimizing both the makespan and the costs is NP-hard. This means that it is unlikely that there exists a polynomial time algorithm capable of solving the problem. Because of the NP-hardness and the need for a fast solution method (also for large problem cases), we focus on developing a heuristic for the problem.

Some multiobjective scheduling problems have been considered in the literature, but the research has concentrated mostly on single machine problems [13,15]. We are not aware of any research on uniform parallel machine scheduling problems with the objectives of minimizing both the makespan and the costs. There are however some studies on unrelated parallel machine scheduling problems with these objectives [6,9]. The algorithms presented require that upper bounds T for the makespan and C for the costs are given. For example in [9], the algorithm forms a schedule with makespan at most $(1+\varepsilon)T$ and costs at most $(1+\varepsilon)C$ if there exists a schedule with makespan T and costs C. This kind of algorithms cannot be used for our network connection selection problem because we cannot determine values T and C a priori.

In [16], solving a general multiobjective scheduling problem has been divided into three subproblems: modelling the problem, taking into account the objectives and scheduling. The first phase, modelling the problem, means defining the scheduling problem and the objectives. In the second phase, the problem is to decide how the conflicting objectives will be taken into account, that is, how a compromise solution between the objectives can be obtained. In the last phase, the actual scheduling problem is solved and a solution to the multiobjective scheduling problem is obtained.

In our previous paper [14], we concentrated on modelling the network connection selection problem and taking into account the conflicting objectives. We compared different multiobjective optimization methods that do not need human interaction, studied the nature of the problem and found a method that produces a good compromise between the conflicting objectives. The method combines the two objectives into a scalarizing function that can be minimized in order to obtain a good compromise solution for the multiobjective optimization problem. Unfortunately, in practice, this method cannot be applied in solving the network connection selection problem because it is computationally far too demanding. Thus, we need new approaches.

In this paper, we continue our research on the network connection selection problem and consider the third phase: solving the problem. The goal of our study has been to develop a fast automatic solution method that uses only little computational capacity. Therefore, we have developed a heuristic solution method which we present here. The heuristic uses the scalarizing function found appropriate for solving the network connection selection problem in [14] to measure the solution quality. This idea of using a scalarizing function to measure the solution quality is new, since usually in heuristics developed for scheduling problems different objectives are considered separately. The heuristic is applicable also to other combinatorial multiobjective optimization problems.

The rest of this paper is organized as follows. In the next section,we give a mathematical model of the network connection selection problem. In Sect. 3, we define some concepts of multiobjective optimization and describe a scalarizing function. The scalarizing function is used in the heuristic we propose in Sect. 4. Computational results are given in Sect. 5, and finally, we conclude in Sect. 6.

2 Model

Let us assume that there are m network connections available and data consisting of n components is to be transferred using them. The connections available and their properties are known because the mobile terminal requests this information from network operators before the data transmission. Transmitting all the components is considered as a transaction that is time-sensitive. The time used in the transaction is assumed to be short and, therefore, the properties of the network connections can be assumed to be fixed during the transaction.

Let x_{ij} be a binary variable such that $x_{ij} = 1$, when component j is transferred using connection i, otherwise $x_{ij} = 0$, $i = 1, \ldots, m$, $j = 1, \ldots, n$. A coefficient, *duration*, d_{ij} represents the time used in transferring component j using connection i. Another coefficient c_{ij} represents the cost of using connection i to transfer component j.

Now, the network connection selection problem can be formulated as follows: minimize

$$f_1(x) = \max_{i=1,\ldots,m} \sum_{j=1}^{n} d_{ij}x_{ij} \quad \text{and} \quad f_2(x) = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to

$$\sum_{i=1}^{m} x_{ij} = 1, \quad \text{for all } j = 1, \ldots, n, \tag{1}$$

$$x_{ij} \in \{0, 1\}, \quad \text{for all } i = 1, \ldots, m \text{ and } j = 1, \ldots, n.$$

The objective function f_1 expresses the time used in transferring the components and the objective function f_2 denotes the costs of the transfers. The constraints (1) require that each component is transferred using exactly one network connection.

When the model is solved, the solution tells which connection is used for transferring each component. It does not pay attention to the order in which the components are transferred on each network connection because neither the time used in transfers nor the total costs depend on the order of the components. After solving the problem, the components can be ordered, for example, in the increasing order of the component sizes on each network connection.

It should be noted that since we cannot determine the exact transmission rate a priori, the minimum guaranteed rate of each connection given by the operator of the connection is used in calculating the durations. This ensures that the actual time used in each transmission is never longer than the duration d_{ij}.

3 Concepts of multiobjective optimization

Before we start solving the network connection selection problem, we briefly define some concepts of multiobjective optimization. Let us consider a problem where we want to minimize two conflicting objective functions $f_1(x)$ and $f_2(x)$ simultaneously subject to a general constraint $x \in S$. The vector of objective functions, called *objective vector*, is denoted by $F(x) = (f_1(x), f_2(x))^{\mathrm{T}}$, and the vector $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ is called a *decision vector*.

Generally, it is not possible to find a solution in which both objective functions attain minimal values. A decision vector $x^* \in S$ and the corresponding objective vector $F(x^*)$ are *Pareto optimal* if there does not exist another decision vector $x \in S$ such that $f_i(x) \leq f_i(x^*)$ for $i = 1, 2$ and $f_j(x) < f_j(x^*)$ for at least one j [11].

Pareto optimality guarantees that we cannot improve any objective function value of the solution without deteriorating the other objective function value. In other words, all the Pareto optimal solutions are mathematically equivalent. Which of them is the best and is selected to be the final solution depends on the problem settings.

An objective vector containing the minimal value of each objective function is called an *ideal objective vector*, z^* [11]. With conflicting objectives, the ideal objective vector is infeasible. The components z_i^* of the ideal objective vector are obtained by minimizing both objective functions f_i separately subject to the constraint $x \in S$.

From the ideal objective vector we get the lower bound of the Pareto optimal set for each objective function. The upper bounds of the Pareto optimal set are the components of a *nadir objective vector* z^{nad}. In the case of two objective functions, the nadir objective vector can be obtained at the same time the ideal objective vector is calculated. The first component of the nadir objective vector is the value of the first objective function in the point where the second objective function attains its minimal value, and the second component is the value of the second objective function in the point where the first objective function attains its minimal value [3].

Multiobjective optimization problems are usually solved by scalarization [11]. *Scalarization* means converting the problem with multiple objectives into a single objective optimization problem or a family of single objective optimization problems. When a multiobjective optimization problem has been scalarized, methods developed for single objective optimization can be used for solving the problem. The objective function of the single objective problem is called a *scalarizing function*.

In [14], we compared different scalarizing functions (including the well-known weighting method) for solving the network connection selection problem. We next describe one of them that was found the most suitable for our purposes. This scalarizing function is used in the heuristic to be presented in Sect. 4.

The following achievement scalarizing function is minimized subject to the constraint of the problem:

$$\max_{i=1,2} w_i \frac{f_i(x) - z_i^{\text{mid}}}{z_i^{\text{nad}} - z_i^*} + \rho \sum_{i=1}^{2} w_i \frac{f_i(x) - z_i^{\text{mid}}}{z_i^{\text{nad}} - z_i^*}, \tag{2}$$

where z^{mid} is a middle point located in the middle of the ranges of the objective functions in the Pareto optimal set, that is,

$$z_i^{\text{mid}} = \frac{z_i^{\text{nad}} + z_i^*}{2}$$

for $i = 1, 2$ and the augmentation coefficient ρ is a small positive scalar. The ratio of the positive weighting coefficients w_1 and w_2 represents here the rate

at which the user of the mobile terminal is willing to trade off values of the objective functions. In (2), denominators are used to scale the terms of the scalarizing function to be of a similar magnitude. This increases the computational efficiency of this type of scalarizing functions, as shown in [12]. The solution to the achievement scalarizing function (2) is Pareto optimal [11].

Calculating ideal and nadir objective vectors means additional computations. To avoid that, we use approximations of the ideal and the nadir objective vectors. We use vector $(0, 0)^{\mathrm{T}}$ to approximate the ideal objective vector. This vector is infeasible because no data can be transferred without any costs and without using any time. The nadir objective vector is estimated from the problem data as follows. The first component of the vector is the objective function value $f_1(x)$ related to a solution x where every component is transferred using the slowest network connection. The second component of the vector is the objective function value $f_2(x)$ related to a solution x where every component is transferred using the most expensive connection. These estimates are larger than or equal to the components of the real nadir objective vector. Though the estimate of the second component may be very rough, the estimates are sufficient for our purposes [14], and no optimization problems need to be solved to get them.

4 Heuristic

Because the capacity of mobile terminals is limited, our aim is to develop an algorithm that uses as little computational resources as possible. In other words, the heuristic should produce a good enough solution while being as simple as possible. In addition, the network connection selection problem requires that the solution is obtained fast. Otherwise, the user of the mobile terminal is not satisfied.

In [14], we studied the settings of the network connection selection problem using different problem instances. The instances represented different kinds of cases of the problem that may occur. In all the examples studied, we could identify a solution that is a good compromise between the objectives and, thus, a logical choice for the final solution. We studied different scalarizing functions in order to find a method that produces a solution near the good compromise solution. Our computational tests showed that the achievement scalarizing function (2) was the most suitable and robust for that purpose.

The problem has binary-valued variables which makes minimizing the scalarizing function using exact methods, such as branch-and-cut methods [10], time-consuming. Therefore, we need to develop a heuristic that gives a solution near the optimal solution to the scalarizing function (2). Our heuristic to be presented next is a simple local search method that uses the scalarizing function (2) to measure the solution quality.

Because we cannot use a lot of computational capacity in our heuristic, we use simple moves called 1-exchanges [17] for improving an initial solution. A 1-exchange means in this case that a component j that is assigned to a connection i is moved from connection i to another connection i'. This move

should improve the solution, otherwise it is not performed. We want to use 1-exchanges in a systematic way, and not to use random selection. The reason for this is that if we use random selection, 1-exchanges may be used for some component many times, whereas for some components the 1-exchanges are not applied at all, if there are only few iterations of 1-exchanges in the heuristic. Therefore, 1-exchanges are used for each component j in turn, and the potential new connection i' is chosen in the following way: The algorithm considers all the connections except connection i, and connection i' is the connection that gives the best solution when component j is transferred using it.

As already mentioned, we use the scalarizing function (2) to measure the solution quality. In other words, the lower the value of the scalarizing function is, the better the solution is. The parameters of the scalarizing function are given the following values: the augmentation coefficient ρ was set to 0.001 and the coefficients of the objective functions (time and costs) w_1 and w_2 were set to 1 and 2, respectively. These values were used also in [14]. This idea of using a scalarizing function to measure the solution quality is new: In heuristics developed for scheduling problems the objectives are usually considered separately.

The initial solution plays a very important role in getting a good final solution. It is desirable that the initial solution enables the improvements to lead the search near a good compromise solution. Therefore, we use the scalarizing function (2) also when forming the initial solution: We set each component in the initial solution to the network connection that gives the lowest scalarizing function value. This kind of an initial solution ensures that the scalarizing function measures the solution quality already when the first feasible solution is constructed.

To be more precise, the initial solution is formed as follows. A set C containing the components that have already been assigned to a network connection is initially empty. At each iteration, we consider a component i not yet in the set C, add it to the set and assign it to a connection. We define a *partial problem* as a problem consisting of all the connections and the components currently in C. The connection to which the component is assigned is the one that gives the lowest scalarizing function value for the partial problem. (Note that the assignments of the other components in C are fixed.) In order to be able to use the scalarizing function, the nadir objective vector of the partial problem has to be approximated. The vector is approximated as presented in Sect. 3.

Now we can sum up the heuristic algorithm as follows:

1. Initialize by setting $C = \emptyset$.
2. Construction of the initial solution. For each component $i = 1, \ldots, n$:
   (a) Add component i in the set C.
   (b) Approximate the nadir objective vector for the partial problem consisting of all the network connections and the components in C.
   (c) Assign component i to the network connection that gives the lowest value of the scalarizing function (2) for the current partial problem when the assignments of the other components in the partial problem

are fixed. The nadir objective vector approximated in the previous step is used in the scalarizing function.

3. Set the initial solution formed in the previous step as the current solution.
4. IMPROVEMENTS. For each component $i = 1, \ldots, n$, use 1-exchange:
   (a) Select the connection that gives the lowest value of the scalarizing function (2) when component i is transferred using it, instead of using the connection the component is assigned to in the current solution.
   (b) If the value of the scalarizing function is smaller than in the current solution, set component i to the selected connection and set this as the current solution.
5. Repeat Step 4 if the stopping criterion is not satisfied.

The stopping criterion can be a certain number of improvement rounds (Step 4) done or no improvement in the solution in the previous round of improvements. The stopping criterion can also be a combination of these criteria.

We have not yet specified in which order the components are dealt with in the heuristic. The 1-exchanges can be used for the components in a random order, which means, for example, the order in which the components of the data happen to be, or in the decreasing order of the component sizes. These orders can also be used in forming the initial solution. The decreasing order is intuitively appealing, because the larger the component is the more influence it has on the solution.

Without ordering the components, the computational complexity of the heuristic is $O(mn)$ (note that usually $m < n$). The ordering of the components can be done using quicksort, which has the worst case complexity of $O(n^2)$ and the mean complexity of $O(n \log n)$ [2]. Then, the computational complexity of the heuristic is $O(n^2)$ if the components are ordered.

We have tested the heuristic using both random order and the decreasing order. Thus, in the following computational results, there are two versions of the heuristic: in *heuristic1*, the components are dealt with in a random order, whereas in *heuristic2* the 1-exchanges are applied to the components in a decreasing order of their sizes. The heuristic is stopped in our computational tests when the solution has not improved during the previous round of improvements.

Finally, we want to remark that the scalarizing function used to measure the solution quality can be changed during the heuristic. In other words, we can use different scalarizing functions in forming the initial solution and in the improvemet phase, or we can use another scalarizing function after few rounds of improvements to refine the final solution.

5 Computational results

The two versions of the heuristic were tested with 16 reality-based test instances. By reality-based instances we mean that the components of each problem instance form a real web page and the properties of the network connections

(that is, price and transmission rate) are estimates for connections of the near future.

Because we cannot predict exactly what kind of a pricing model the operators will use in the future, we have used two different pricing models. In the first model, the pricing is linear in the size of the component. Thus, the price is, for example, $8.2 \cdot 10^{-8}$ euros per bit (and naturally the price is different for each connection). The second model is more elaborate: the price is calculated using formula

$$c_{ij} = K_0 + K_1 \exp(K_2 \cdot \text{rate}(i)) \cdot \text{size}(j), \tag{3}$$

where rate(i) is the rate of connection i (bits per second), *size*(j) is the size of component j in bytes and K_0, K_1, and K_2 are connection-specific coefficients. The coefficients K_0, K_1, and K_2 used for GSM connections are 9.0, 0.31, and 0.000043, respectively, and for UMTS connections the coefficients are 6.0, 8.8, and 0.0000017, respectively. The cost given by formula (3) has to be multiplied by 10^{-5} in order to get the cost in euros.

The test instances are briefly summarized in Table 1: For each instance, the number of components n and the number of network connections m as well as the total size of the components are given. The table reports also which of the pricing models is used in each test instance. We would like to note that the number of network connections m is small in the test instances because mobile terminals cannot use too many connections simultaneously.

As mentioned earlier, the solution obtained by minimizing the scalarizing function (2) is a good Pareto optimal compromise between the conflicting objectives. That is why we compare the performance of our heuristic to that solution. This optimal solution is calculated using CPLEX 8.0, which uses a branch-and-cut method to solve integer programming problems [7]. Both the heuristic and CPLEX were run in a computer with a 550 MHz HP PA-RISC processor. The relative optimality tolerance used in CPLEX was 10^{-8}, with the exception of

Table 1 Test instances (with n components and m connections)

Instance	n	m	Total size (bytes)	Pricing model
geu1	10	3	80,400	Linear
geu2	15	5	154,600	Linear
geu3	25	5	156,141	Linear
city	9	5	65,671	Linear
mit	11	5	70,776	Linear
jyu	12	5	22,795	Linear
hut	14	5	73,706	Linear
helsinki	15	5	61,168	Linear
wireless	15	5	34,028	Linear
yahoo	17	5	44,384	Linear
geizhals	18	5	24,003	Linear
bbc	23	5	27,554	Linear
gigantti	24	5	39,094	Linear
gsm-umts	14	5	73,706	Formula (3)
gsm-umts2	18	5	24,003	Formula (3)
gsm	17	3	44,384	Formula (3)

instance geu3 in which the tolerance was 10^{-4}. (A smaller tolerance 10^{-6} caused CPLEX to run 38 days of CPU time without establishing the final solution.) The relative optimality tolerance of 10^{-4} guarantees that the solution given by CPLEX is within 0.01% of the optimal solution value.

The computational results are presented in Tables 2 and 3. Table 2 contains (in the last column) the objective function values of the optimal solutions to the scalarizing function (2), to which the heuristic solutions are compared. In order to ease comparison, instead of reporting the solutions given by the two versions of the heuristic, the differences of the heuristic solutions to the optimal solution are given in the table. The differences in the objective function values (time, costs) are in parentheses, and they are calculated by subtracting the value of the optimal solution from the value of the heuristic solution. The differences in the value of the scalarizing function (2) between the heuristic and optimal solutions are also presented in Table 2. The average differences in the scalarizing function value are 0.0026 (heuristic1) and 0.0023 (heuristic2), which are very good results. It is also significant that in the cases of two test instances the solution given by both versions of the heuristic is the same as the optimal solution to the scalarizing function (2).

In Table 3, the CPU times used by the heuristics and CPLEX are given, as well as the number of improvement rounds done in each heuristic. The number of improvement rounds applied in the heuristics was small in all the problem instances tested. In one instance (geu3) four rounds were needed, whereas in the other cases at most two rounds were applied. Thus, if we had stopped the iteration in the heuristics after two rounds of improvements, the solutions would have remained the same except in the instance geu3 with the version

Table 2 Differences between heuristic solutions and optimal solutions and differences measured using the scalarizing function (2)

	heuristic1		heuristic2		Optimal solution
Instance	(time, costs)		(time, costs)		
geu1	(0.0, 0.0514)	0.0001	(0.0, 0.0514)	0.0001	(4.7778, 0.4721)
geu2	(0.0, 0.0047)	0.0000	(0.0, 0.0009)	0.0000	(5.0, 0.7664)
geu3	(0.1035, -0.0288)	0.0012	(0.1222, -0.0304)	0.0014	(4.4076, 1.0318)
city	(0.0, 0.0008)	0.0000	(0.0, 0.0008)	0.0000	(2.3224, 0.3441)
mit	(-0.0785, 0.0015)	0.0017	(-0.0785, 0.0015)	0.0017	(2.1137, 0.4712)
jyu	(0.0728, 0.0027)	0.0058	(0.0306, 0.0028)	0.0024	(0.7182, 0.1134)
hut	(0.0, 0.0)	0	(0.0, 0.0)	0	(4.3076, 0.3450)
helsinki	(0.0581, -0.0174)	0.0017	(0.0229, -0.0066)	0.0007	(1.7297, 0.4039)
wireless	(0.0472, -0.0045)	0.0025	(0.0472, -0.0045)	0.0025	(0.9641, 0.2247)
yahoo	(0.1538, -0.0258)	0.0062	(0.1538, -0.0258)	0.0062	(1.2594, 0.2934)
geizhals	(0.0253, 0.0016)	0.0019	(0.0253, 0.0016)	0.0019	(0.6848, 0.1570)
bbc	(0.0, 0.0243)	0.0001	(0.0, 0.0243)	0.0001	(0.8407, 0.1434)
gigantti	(0.0461, -0.0137)	0.0021	(0.0461, -0.0023)	0.0021	(1.1033, 0.2584)
gsm-umts	(0.0, 0.0)	0	(0.0, 0.0)	0	(8.8614, 2.2052)
gsm-umts2	(0.1033, -0.0016)	0.0078	(0.0559, 0.0191)	0.0069	(1.2099, 1.1936)
gsm	(0.0508, 0.0087)	0.0112	(0.0508, 0.0087)	0.0112	(6.6995, 0.5904)

Table 3 CPU-times and numbers of improvement rounds

	heuristic1		heuristic2		Optimal time(s)
Instance	time(s)	rounds	time(s)	rounds	
geu1	<0.01	1	0.01	1	0.05
geu2	0.02	1	0.01	1	0.08
geu3	0.04	4	0.04	2	79.64
city	0.01	1	0.01	1	0.07
mit	0.01	1	0.01	2	0.08
jyu	0.01	1	0.01	2	0.08
hut	0.01	0	0.01	0	0.07
helsinki	0.01	2	0.01	2	0.9
wireless	0.01	0	0.01	0	2.41
yahoo	0.01	1	0.01	1	1.13
geizhals	0.01	2	0.01	1	7.85
bbc	0.01	0	0.01	0	0.33
gigantti	0.02	1	0.02	1	3488.51
gsm-umts	0.01	0	0.01	0	0.07
gsm-umts2	0.01	2	0.01	1	54.72
gsm	0.01	0	0.01	0	0.42

heuristic1. The difference to the optimal solution in that case would have been $(0.1624, -0.0324)$ and the difference in the value of the scalarizing function 0.0019. In other words, this solution would have also been very good. However, the time needed by the heuristic would have remained the same. It should be noted that in five test instances, the initial solution was already very close to the optimal solution and the solution did not change in the improvement phase.

Both versions of the heuristic are fast: the CPU times used were typically around 0.01 seconds and always less than 0.04 s in all the test instances. Thus, the speed did not vary a lot between different instances. On the contrary, the times needed by CPLEX varied much, and in most instances the times were so long that they would not be acceptable. This is natural since exact solution methods, such as the branch-and-cut methods, are known to be slow. Yet, the solutions produced by the heuristics were very close to those of CPLEX, as mentioned above. Our heuristic may seem simple but in our problem setting it is a significant benefit. From the practical point of view, it is very important that we can say that we managed to achieve our goal of developing a computationally inexpensive method for solving the network connection selection problem without sacrificing the solution quality.

There was not much difference between the two versions of the heuristic, when we consider the quality of the solutions, the time used, and the number of improvement rounds taken. It is important to note that the ordering of the components done in the version heuristic2 did not increase the time used when compared to the other version. On the other hand, the solutions given by the different versions of the heuristic are very similar, even the average difference to the optimal solution is almost the same. Therefore, we cannot claim superiority over either of the versions.

We must point out that the computer used in the computational tests is quite efficient. Therefore, one might ask whether it would be at all possible to run

the heuristic in a mobile terminal. Actually, mobile terminals cannot yet run the heuristic as fast as the computer we used for the computational results, but the computational capacity of mobile terminals is growing fast. It is likely that in few years mobile terminals using, for example, an Intel XScale processor [8] will be able to run the heuristic as fast as our computer now.

6 Conclusions

In this paper, we have considered the network connection selection problem, which is a multiobjective scheduling problem, where a set of components is to be transferred using distinct network connections. The objective of the problem is to minimize both the time used for the transfers and the costs of the transfers.

We have presented a simple but efficient local search heuristic for the network connection selection problem. The idea of the heuristic is to improve an initial solution by simple moves called 1-exchanges. In a 1-exchange, a component is moved to be transferred using another connection if the solution quality improves. The core of our method is the novel idea of measuring the solution quality using a scalarizing function (found appropriate for producing a good compromise between the conflicting objectives in our previous study). The same scalarizing function is used also when forming the initial solution. The results obtained are very encouraging. The ideas of the heuristic are applicable also to other combinatorial multiobjective optimization problems and we can recommend using scalarizing functions as parts of heuristics in this field.

So far in our study, we have assumed that the time needed for the transmission is short. Then, we can assume that each connection has a fixed rate and a fixed price during the transmission. A topic of future research is to consider the problem with longer transmission times when the properties of the environment can change during the transmission.

Acknowledgements This research was supported by the GETA graduate school. The authors also wish to thank Dr. Marko M. Mäkelä and Dr. Heikki Maaranen for discussions on the heuristic and Mr. Aki Suihkonen for collecting data for the test instances.

References

1. Błażewicz, J.: Selected topics in scheduling theory. In: Martello, S., Laporte, G., Minoux, M., Ribeiro, C. (eds.) Surveys in Combinatorial Optimization, vol. 31 of Annals of Discrete Mathematics, p. 1–59. Elsevier, Amsterdam (1987)
2. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. The MIT Press and McGraw-Hill Book Company, New York (1990)
3. Ehrgott, M., Tenfelde-Podehl, D.: Computation of ideal and Nadir values and implications for their use in MCDM methods. Eur. J. Operat. Res. **151**(1), 119–139 (2003)
4. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Hammer, P.L., Johnson, E.L., Korte, B.H. (eds.) Discrete optimization II, vol. 5 of Annals of Discrete Mathematics, pp. 287–326. North-Holland Publishing Company, Amsterdam (1979)
5. Gustafsson, E., Jonsson, A.: Always best connected. IEEE Wirel. Commun. **10**(1), 49–55 (2003)
6. Hall, L.A.: Approximation algorithms for scheduling. In: Hochbaum, D.S. (ed.) Approximation Algorithms for NP-hard Problems. PWS Publishing Company, Boston (1997)

7. ILOG CPLEX 8.0 User's Manual. ILOG, (2002)
8. Intel XScale Microarchitecture, Technical Summary. Intel Corporation, (2000)
9. Jansen, K., Porkolab, L.: Improved approximation schemes for scheduling unrelated parallel machines. Math. Oper. Res. **26**(2), 324–338 (2001)
10. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms. Springer, Berlin Heidelberg New York (2002)
11. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Dordrecht (1999)
12. Miettinen, K., Mäkelä, M.M., Kaario, K.: Experiments with classification-based scalarizing functions in interactive multiobjective optimization. Eur. J. Oper. Res. (2006) (to appear)
13. Nagar, A., Haddock, J., Heragu, S.: Multiple and bicriteria scheduling: a literature survey. Eur. J. Oper. Res. **81**(1), 88–104 (1995)
14. Setämaa-Kärkkäinen, A., Miettinen, K., Vuori, J.: Best compromise solution for a new multiobjective scheduling problem. Comput. Oper. Res. **33**(8), 2353–2368 (2006)
15. T'Kindt, V., Billaut, J.-C.: Multicriteria scheduling problems. In: Ehrgott, M., Gandibleux, X. (eds.) Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys, pp. 445–491. Kluwer, Dordrecht (2002)
16. T'kindt, V., Billaut, J.-C.: Multicriteria Scheduling: Theory, Models and Algorithms. Springer, Berlin Heidelberg New York (2002)
17. Trick, M.A.: Scheduling multiple variable-speed machines. Oper. Res. **42**(2), 234–248 (1994)
18. Varshney, U., Jain R.: Issues in emerging 4G wireless networks. IEEE Comput. **34**(6), 94–96 (2001)

PIII

OPTIMAL USAGE OF MULTIPLE NETWORK CONNECTIONS

by

Anne Setämaa-Kärkkäinen and Jani Kurhinen 2008

In Proceedings of the 1st International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications, MOBILWARE 2008, ed. by Jiang (Linda) Xie and Tuna Tugcu

Optimal Usage of Multiple Network Connections

Anne Setämaa-Kärkkäinen
University of Jyväskylä
P.O. Box 35 (Agora)
FI-40014 University of Jyväskylä, Finland
annseta@mit.jyu.fi

Jani Kurhinen
University of Jyväskylä
P.O. Box 35 (Agora)
FI-40014 University of Jyväskylä, Finland
kurhinen@mit.jyu.fi

ABSTRACT

In the future mobile networks, a mobile terminal is able to select the best suitable network for each data transmission. The selection of a network connection to be used has been under a lot of study. In this paper, we consider a more extensive case in which we do not select a network connection but use several network connections simultaneously to transfer data. When data is transferred using multiple network connections, a network connection has to be selected for each component of the data. We have modelled this problem as a multiobjective optimization problem and developed a heuristic to solve the problem fast in a static network environment. In this paper, we discuss solving the problem in a dynamic network environment in which the availability as well as the rate and prizing of connections vary. We introduce an improved version of the heuristic that reacts to changing network conditions and improves the solution when possible.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication; G.1.6 [**Optimization**]: Integer programming

General Terms

Algorithms

Keywords

Wireless networks, network connection selection, scheduling, multiobjective optimization, heuristics

1. INTRODUCTION

The number of mobile users and mobile services available is increasing fast. The increasing amount of transferred data has created a need for new faster types of transmission methods. However, the faster the transmission rate is the smaller the wireless transmission range usually is. This progress forces terminal manufacturers to design multimode devices that can operate using several different kinds of network technologies. When these versatile terminals are used within an environment where there exist several network access providers, the users of mobile terminals can choose both the network access technology and the network access provider.

In this paper we consider a situation in which it is possible to use several different kinds of networks from different access providers simultaneously. In order to enable using several connections at the same time, an intelligent connection manager is needed to control the multiple seamless connections. This issue has been studied earlier mainly from the network-level point of view. However, there is also a need to take into account user-level parameters when making a decision on the network connections used. We have considered the network connection selection from the users point of view. The user usually wants to transfer data as fast and as cheaply as possible. Therefore, we consider the rates and the prices of the connections as the selection criteria. Also other issues could be taken into account since the solution method developed is general.

The selection of network connections is made using optimization. The optimization problem considered is which connection to use for transferring each component of the data. We call this problem the *network connection selection problem* (NCS) and it is formulated as a multiobjective optimization problem where the objective is to minimize both the costs and the time used in transferring. The NCS problem was first introduced in [10], in which the problem was formulated as a multiobjective optimization problem and different multiobjective optimization methods were compared for solving the problem. In [11] a heuristic for solving the problem in a static environment was presented. In this paper, we present an improved dynamic version of the heuristic that reacts to changing network conditions. We also discuss some technical issues involving the NCS problem.

This paper is organized as follows. In the next section, we discuss technical issues of using multiple network connections simultaneously. In Section 3, the network connection selection problem is modelled as a multiobjective optimization problem. Solving the problem is discussed in Section 4 where a heuristic algorithm developed for solving the problem in a static environment is briefly described. Dynamic environment is considered in Section 5 and the heuristic is modified to deal with changing network conditions. Sec-

Mobilware '08, February 12-15, 2008, Innsbruck, Austria.

tion 6 presents an illustrative example, and conclusions are drawn in Section 7.

2. NETWORK CONNECTION SELECTION

A current trend in networking is the convergence of different networks, including the process of wired and wireless networks joining together and forming a single fourth generation (4G) network. The 4G networks are not dependent on a certain frequency or a modulation method as the current generation of networks is [2]. The 4G networks are based on the idea of roaming between networks, even though those networks do not share the same technology. The mobility between different technologies is called vertical roaming, and it is needed when designing multi-technology networks. One downside of this mobility between different types of networks is that an access terminal will become more complicated, since it has to be able to receive and transmit data using different frequencies and modulations.

A technology that could provide an answer to this problem is called software defined radio [9]. With the software defined radio, the radio transceiver module is not hardwired to decode one particular transmission technology, but it has the ability to change the communication parameters. In this kind of a system, it is possible to define separate realizations of different types of transceivers as software which can be saved in the memory of the device. When a different type of network access is needed, one (or many, depending on the situation) of the stored transceivers can be loaded to be actively used. With programmable hardware this active transmitter/receiver system reconfigures itself at a hardware level to be able to execute efficiently the selected communication interface program. This way a network-independent access can be provided to the application layer.

We believe that in the near future the next generation networks will make it possible to use multiple network connections simultaneously. The concept of being always best connected (ABC) introduced by Gustafsson and Jonsson [5] supports our opinion. The ABC concept means that the users of mobile terminals have the opportunity of choosing the network access technology that is most suitable for them. According to the concept, users will be able to choose also the access provider. Thus, the selection of the network connection to be used is not limited to a certain network operator, but the user can use any network connection provided by any network operator. If the future networks provide this kind of an inter-operator roaming the operators will start competing more rigorously, from which the customers will benefit. Especially, if it was possible to change the network operator on the fly, it would enable new types of network access markets to be born.

There are many studies that consider the situation where there are multiple network connections available [1, 3, 6, 7, 8, 12]. In these studies the aim is to choose a network connection to be used in data transmission. We, however, use multiple network connections simultaneously and our aim is to select a network connection for each component of the data in such a way that the costs and the time used in transferring are minimized. The optimization method we propose in this paper can be used, for example, within the framework presented in [7] or within other similar frameworks. The framework proposed in [7] includes a virtual network interface that hides the actual physical channels among which the used network connection is selected. Below the virtual interface there are a policy manager and an interface manager which together make the selection. The model also requires one additional system component to monitor the availability of the network connections. In this proposal a user can control the policy manager and thereby affect the selection of the network connection.

In [3] a method based on multi-attribute decision making is presented to rank the networks available and to choose a network connection to be used in data transmission. We could use this kind of a method to select candidate networks to be considered in our algorithm. The selected network connections would be such that it is possible to use them to transfer data when we consider the properties of both the mobile terminal and the networks available as well as the quality of service required. In addition, we could disregard some network connections if the user finds them to be too expensive to use. This, however, is not the topic of this paper. From now on, we assume that some kind of a selection is made, and all the network connections we consider in the NCS problem are such that it is possible to transfer the data using them when considering the properties of the terminal and the quality of service required.

3. MATHEMATICAL MODEL

The NCS problem was first introduced in [10], and it can be defined as follows. We assume that a mobile terminal is able to use several different kinds of network connections simultaneously. The connections are used to transfer data that consists of separate components that can be transferred independently. This means that the components can be transferred using different networks at the same time. The problem is to decide which network connection should be used for transferring each data component. The aim is to minimize the costs and the time used in transferring.

In this section, we present a mathematical model of the NCS problem originally formulated in [10]. Let us assume that there are m network connections available in a mobile terminal and data consisting of n separate components is to be transferred using the connections. The components are assumed to be independent of each other so that it is possible to send them in any order. All the m network connections are assumed to be such that it is possible to transfer the data using them. In other words, the quality of service properties of these connections are suitable for the data transmission in question.

Let d_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$, denote the time that is needed in transferring component j using connection i, and let c_{ij} be the cost of using connection i to transfer component j. We define that a binary variable x_{ij} equals one, when component j is transferred using connection i, otherwise it is zero. Now, the NCS problem can be stated mathematically as follows:

minimize

$$f_1(x) = \max_{i=1,\ldots,m} \sum_{j=1}^{n} d_{ij} x_{ij} \quad \text{and} \quad f_2(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

subject to

$$\sum_{i=1}^{m} x_{ij} = 1, \quad \text{for all } j = 1, \ldots, n, \tag{1}$$

$$x_{ij} \in \{0, 1\}, \quad \text{for all } i = 1, \ldots, m \text{ and } j = 1, \ldots, n. \tag{2}$$

The objective function f_1 denotes the time used in transferring the components, that is, the connectivity time of the connection used for the longest time interval. The objective function f_2 expresses the costs of the transfers. The constraints (1) require that each component is transferred using exactly one connection. The constraints (1) and (2) form a set of feasible solutions S, from which the best solution to the problem is searched.

When the problem is solved, the solution tells which connection is used for transferring each component. It does not pay attention to the order in which the components are transferred on each connection because neither the time used in transfers nor the total costs depend on the order of the components. The components can be ordered on each connection according to some criteria. They can be transferred, for example, in the increasing order of their sizes. Note that in this model, we do not define how the cost of transferring a component is calculated but only assume that the cost c_{ij} can be expressed for each component j and each connection i. The cost c_{ij} can also be an estimate of the actual cost as long as the estimates for different connections are comparable.

4. SOLVING THE PROBLEM

Our aim is to develop an automatic method to solve the NCS problem fast. In addition, the method can use only little computational capacity so that it is possible to implement it as a middleware application in a mobile terminal. Since we have two objectives, we need a method that quickly and automatically obtains a good compromise between the objectives. In [10], we analyzed the main characteristics of the NCS problem and compared different multiobjective optimization methods for solving it. We found the following method to be the most suitable for our purposes. The method produces a compromise solution by minimizing the following function

$$\max_{i=1,2} w_i \frac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^*} + \rho \sum_{i=1}^{2} w_i \frac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^*} \qquad (3)$$

subject to $x \in S$, where z_i^{mid} is a middle point located in the middle of the ranges of the objective function, that is,

$$z_i^{mid} = \frac{z_i^{nad} + z_i^*}{2}$$

for $i = 1, 2$, and z^* and z^{nad} are the ideal and nadir objective vectors representing the best and worst objective function values, respectively. The coefficient ρ is a small positive scalar, and the ratio of the positive weighting coefficients w_1 and w_2 represents the rate at which the user of the mobile terminal is willing to trade off values of the objective functions. We use value 0.001 for the coefficient ρ and values 1 and 2 for the weighting coefficients w_1 and w_2, respectively. These values were used also in [10].

Before minimizing the objective function (3) the ideal and nadir objective vectors have to be calculated. The *ideal objective vector* z^* that represents the best values of the objective functions can be obtained by minimizing each objective function separately subject to $x \in S$. The *nadir objective vector* z^{nad} representing the worst values of the objective functions can be obtained at the same time the ideal objective vector is calculated since we consider only two objective functions [4]. Calculating the ideal and nadir objective vectors means additional computations slowing down the solution method considerably. This can be avoided by using their approximations because they are mainly needed for scaling purposes. We use vector $(0, 0)^T$ to approximate the ideal objective vector. The nadir objective vector is estimated from the problem data as follows. The first component of the vector is the objective function value $f_1(x)$ related to a solution x where every component is transferred using the slowest connection. The second component of the vector is the objective function value $f_2(x)$ related to a solution x where every component is transferred using the most expensive connection. These estimates may be very rough. However, the estimates are sufficient for our purposes [10].

Minimizing the objective function (3) is computationally demanding in the case of the NCS problem because of the integer variables [11]. In other words, it is time-consuming to solve large problem instances using exact methods. In addition, many exact methods require a lot of computational capacity to solve large integer programming problems. In mobile terminals, there is not much computational capacity available for this kind of optimization, and the users of mobile terminals are rarely prepared to wait more than a very short time. Therefore, we have developed a heuristic to solve the NCS problem [11]. The heuristic minimizes the objective function (3) approximately, achieving a good compromise solution. The heuristic is fast and, in addition, it does not require a lot of computational capacity, which makes it possible to implement it in a mobile terminal. The heuristic can be summed up as follows: (For more information on the heuristic, see [11].)

1. **Initial solution**. For each component $j = 1, \ldots, n$, find the network connection that gives the lowest value of the objective function (3) for the current partial problem, and assign the component to the connection. The current *partial problem* consists of the component j and the components that are already assigned (with fixed assignments). All the connections are included in the partial problem, and the ideal and nadir objective vectors are approximated for each partial problem separately.

2. **Improvements**. Set the initial solution as the current solution. Repeat twice: For each component $j = 1, \ldots, n$, search for the solution having the lowest value of the objective function (3) among the solutions in which the component j is transferred using another link instead of the link used in the current solution and the assignments of the other components are fixed. Set the new solution found as the current solution if the value of the function (3) is lower than in the current solution.

3. **Ordering**. Order the components assigned to each network connection according to some criteria, for example, the size. The components will be transferred in that order.

Note that originally in [11] the improvement phase in Step 2 is repeated until a stopping criterion (not explicitly defined) is satisfied. Because of the computational results presented in [11], we have decided to repeat the improvement phase twice. As mentioned earlier, the order in which the components are transferred has no impact on the time that

it takes to transfer the components nor on the costs of the transfer. Therefore, we can ignore the order when assigning the components to the connections and order the components afterwards in Step 3. The components need to be ordered to obtain a *transmission schedule* which is followed during the data transmission.

5. DYNAMIC ENVIRONMENT

In the previous section, we briefly presented a heuristic for solving the NCS problem. The heuristic was developed assuming a static environment where the properties of the connections do not change during the transmission. This kind of an assumption can be made only when the transmission time is short. When the transmission rates and the prices change during the transmission, a solution given by the heuristic may need revising. In addition, a network connection can be lost during the transmission if, for example, the mobile terminal enters an area that is not covered by that network. Then, the solution given by the heuristic may become infeasible and changing the solution is necessary in order to finish the transmission.

In order to cope with a dynamic environment the heuristic needs a rescheduling phase that improves the transmission schedule when possible during the transmission. The rescheduling phase takes into account changes in the transmission rates and in the pricing. In rescheduling, a component (or several components) is assigned to be transferred using another connection instead of the connection it is assigned to in the (current) schedule. We next describe the rescheduling phase in more detail.

When a change in the transmission rates or prices occurs, the amount of change is first controlled. If the change is larger than a prespecified limit B the rescheduling phase is run. The limit can be, for example, a certain percentage of the previous rate or price. The rescheduling phase is not run if the change in the transmission rates or prices is small because the solution rarely improves in this case. When there occur multiple changes at the same time the rescheduling phase is run if the largest change is large enough.

We also consider the case in which the network connections can be lost and later return. Now, we first need to control if a connection is lost or a new connection has appeared. If this is the case, the rescheduling phase is run. However, now the rescheduling phase begins by assigning the components that are assigned to the lost connections to other connections. The components considered here are the components that have not yet been transferred. In addition, if the transmission of a component was interrupted when a connection was lost, this component will also be assigned to another connection in order to ensure that the transmission will be finished. The assignment of the components to new links is done the same way as the initial solution is formed in the static version of the heuristic: Each component in turn is assigned to the connection that gives the lowest value of the objective function (3).

The rescheduling phase searches for the best solution (with respect to the objective function (3)) to a *partial problem* consisting of all the connections currently available and the components that have not yet been transferred. These components are collected into a set C and the network connections that are currently available are included in a set N. In addition, a set D is used for the components that are assigned to lost connections and need to be reassigned to other connections. The rescheduling phase consists of the following steps:

1. **Initialization**. Set $C = \emptyset$, $D = \emptyset$ and $N = \emptyset$. Control which connections are currently available and add those connections to the set N. Add to the set C all the components that are waiting to be transferred using the connections in N. Remember the current assignment of the components to the connections. The partial problem consists of the connections in N and the components in C.

2. **Initial solution**. Form an initial solution to the partial problem:

   - If there are no lost connections, the initial solution is the current assignment of the components in C to the connections in N. Go to Step 3.
   - Otherwise, add all the components assigned to the lost connections in the set D and remove the components from the transfer queues of the connections. If a transfer of a component was interrupted when a connection was lost, remove also this component from the transfer queues and add it to the set D.
   - Form the initial solution. For each component j in D: Remove the component j from the set D and add it to the set C. Assign the component j to the network connection i in N that gives the lowest value of the objective function (3) to the partial problem consisting of the components in C and the connections in N when the component j is assigned to the connection i and the assignments of the other components in C are fixed.

3. **Rescheduling**. Set the initial solution as the current solution. Use rescheduling to each component j in C: Form a set of solutions where the component j is assigned to another connection instead of the connection to which it is assigned in the current solution. Select among the solutions the one having the lowest value of the objective function (3) to be a *rescheduling solution*. If the rescheduling solution has a lower value of the objective function (3) than the current solution, set the rescheduling solution as the current solution.

It should be noted that we do not order the components again when rescheduling. The order of the components on each connection stays the same with the exception that the components that are moved to other connections are put last in the transfer queues. This is done at the same time as the rescheduling and, therefore, we do not have to perform any ordering afterwards, which saves some computational time. Remember that the rescheduling phase is run during the transmission and it should be as fast as possible to avoid delaying the transfer.

Figure 1 summarizes how the heuristic responds to a change in the network conditions.

6. ILLUSTRATIVE EXAMPLE

Next we present an example of the NCS problem. We use the term *static heuristic* to refer to the heuristic without the rescheduling phase and the term *dynamic heuristic* to refer

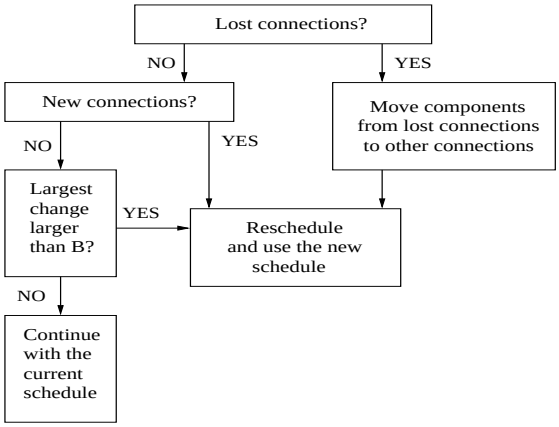


Figure 1: Responding to a change in the network conditions.

to the heuristic using the rescheduling phase. The problem instance we consider in this section has 86 components that form a real web page. The components are to be transferred using a set of eight network connections. The cost of transferring component j using network connection i is calculated using formula

$$c_{ij} = k_0 + k_1\sqrt{r_i} \cdot s_j \quad (4)$$

where r_i is the rate of the connection i (Kbits per second), s_j is the size of the component j in bytes, and k_0 and k_1 are connection-specific coefficients. It should be noted that this formula for calculating the cost is only an example which shows that the price can consist of a fixed cost and a variable cost. How the cost is calculated has no effect on the heuristic.

The eight connections form three groups that have different parameter values in the pricing model. The groups represent competing operators that offer similar network connections with different prices. In the first group, the values for the parameters k_0 and k_1 are 300 and 1.1, respectively. In the second group the values are 100 and 2.3, and in the third group 30 and 0.8, respectively. The rates of the connections are 14.4, 59.2, and 384 Kbits per second in the first group, 14.4, 59.2, and 115 Kbits per second in the second group, and in the last group 14.4 and 59.2 Kbits per second. The cost calculated using the formula (4) is multiplied by 10^{-5} in order to have it in euros.

Let us now consider a dynamic network environment in which the rates of the network connections vary during the data transmission. The original rates of the network connections are assumed to be given by the operators of the networks and they present the maximum rates of the connections. The actual rates of the connections vary during the transmission. In addition, it is possible that a network connection is lost during a transmission due to, for example, movement of the mobile terminal. In our simulation, a change in the transmission rates occurs randomly with a mean of 0.7 changes in a second per network connection. When a change in the rate of a network connection occurs, the network connection is lost with a probability of 0.01. Otherwise, the rate of the connection is uniformly distributed between the maximum rate divided by three and the maximum rate.

It should be noted that in this simulation example the pricing is fixed during the transmission and, therefore, the costs change only when the transmission schedule is changed. In addition, it should be noted that the number of changes occurring is large, which corresponds to a situation where the strength of the signal is weak or the mobile terminal is moving fast. This kind of a situation was chosen in order to demonstrate the performance of the dynamic heuristic.

Figures 2 and 3 illustrate the results of a simulation where the same data is transferred using two different strategies at the same time in the dynamic environment presented above. In one of the transmissions, the schedule given by the static heuristic is followed during the entire transmission. In the other transmission, the schedule is updated using rescheduling when a change in the transmission rates has occured and the change is larger than 20 percent of the previous rate. The times when a change occurs are marked by squares and circles in the figures. It should be noted that the transmission rates given to the static heuristic in the beginning of the simulation are the actual rates, not the maximum rates given by the operators. Using the maximum rates might result in a poor solution.

Figure 2 shows how the expected total time needed for transferring changes during the transmission. How the expected total costs change during the transmission is depicted in Figure 3. In this simulation example rescheduling makes the transmission faster but a bit more expensive. It should be noted that in this simulation the transmission using the static heuristic is stopped on a network connection during a period from 2.59 to 4.22 because the connection is not available then. (The other network connections are available and in use also during that period.) Therefore, in Figure 2 the total transmission time is not marked for that period (because it is not defined). The dynamic heuristic assigns the components waiting to be transferred on the lost connection to other connections using the rescheduling phase. Therefore, the transmission using the dynamic heuristic is not stopped during any period. This makes the total transmission time longer for the transmission using the static heuristic. However, this is only one reason why the dynamic heuristic makes the transmission faster. As we can see in Figure 2, the time needed for transferring using the static heuristic grows rapidly already in the beginning of the transmission. This is caused by a network connection that slows down considerably. The dynamic heuristic removes components from the connection keeping the transmission time shorter but at the same time making the transmission a bit more expensive.

This simulation example shows how the dynamic heuristic works. In this case, the dynamic heuristic makes the transmission shorter but a bit more expensive. In other simulation cases, the result may be the opposite or the dynamic heuristic may make the transmission both shorter and cheaper. Which is then better, to wait longer for the transmission to finish or pay little more for the transmission is a matter of preference. Our algorithm makes this decision automatically but, if wanted, user preferences can be taken into account in the method.

7. CONCLUSIONS

In this paper we have considered a system in which a mobile device can use multiple network connections simultane-

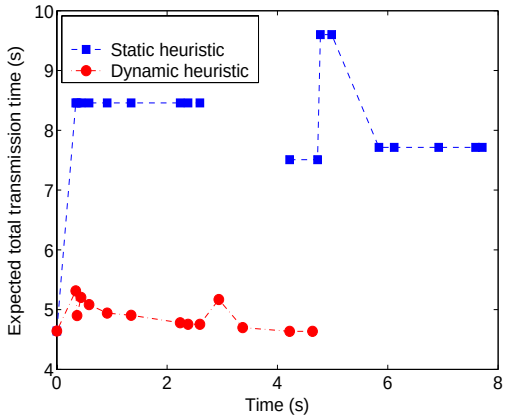


Figure 2: Example of how the expected total transmission time can vary during transmission.

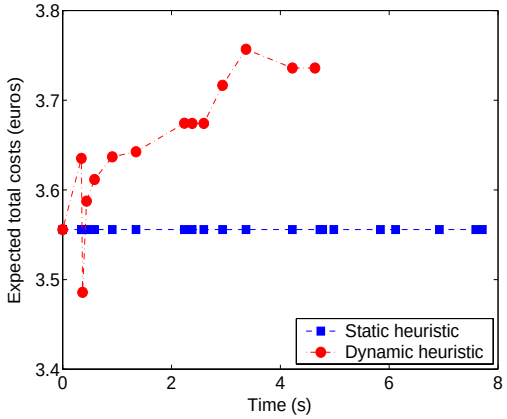


Figure 3: Example of how the expected total transmission costs can vary during transmission.

ously to transfer data. The problem is to select which network connection to use for each component of the data. This problem is called the network connection selection problem, and it has been formulated as a multiobjective optimization problem. To solve the problem, we have presented a simple but efficient heuristic that takes into account the dynamic network environment and improves the solution during the transmission when possible. The heuristic is fast and needs only little computational capacity and, therefore, it is possible to use it in mobile terminals during data transmission.

Future work will concentrate on adjusting the parameters of the heuristic using simulations. The main research question will be when it is profitable to use rescheduling. In other words, we need to determine when the change in the transmission rates and prices is large enough to make improving the current solution worthwhile.

8. ACKNOWLEDGEMENTS

This research was supported by the Emil Aaltonen foundation. The authors also wish to thank Prof. Kaisa Miettinen and Dr. Jarkko Vuori for their help with this research.

9. REFERENCES

[1] E. Adamopoulou, K. Demestichas, A. Koutsorodi, and M. Theologou. Intelligent access network selection in heterogeneous networks - simulation results. In *Proc. of the 2nd International Symposium on Wireless Communication Systems*, pages 279–283, 2005.

[2] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty. NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer Networks*, 50(13):2127–2159, 2006.

[3] F. Bari and V. C. M. Leung. Automated network selection in a heterogeneous wireless network environment. *IEEE Network*, 21(1):34–40, 2007.

[4] M. Ehrgott and D. Tenfelde-Podehl. Computation of ideal and Nadir values and implications for their use in MCDM methods. *European Journal of Operational Research*, 151(1):119–139, 2003.

[5] E. Gustafsson and A. Jonsson. Always best connected. *IEEE Wireless Communications*, 10(1):49–55, 2003.

[6] A. Iera, A. Molinaro, C. Campolo, and M. Amadeo. An access network selection algorithm dynamically adapted to user needs and preferences. In *Proc. of the 17th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2006.

[7] J. H. Li, S. Luo, S. Das, T. McAuley, M. Patel, A. Staikos, and M. Gerla. An integrated framework for seamless soft handoff in ad hoc networks. In *Proc. of the Military Communications Conference*, 2006.

[8] L. Liang, H. Wang, and P. Zhang. Net utility-based network selection scheme in CDMA cellular/WLAN integrated networks. In *Proc. of the IEEE Wireless Communications and Networking Conference*, pages 3313–3317, 2007.

[9] J. Mitola. Cognitive radio for flexible mobile multimedia communications. In *Proc. of the IEEE International Workshop on Mobile Multimedia Communications*, pages 3–10, 1999.

[10] A. Setämaa-Kärkkäinen, K. Miettinen, and J. Vuori. Best compromise solution for a new multiobjective scheduling problem. *Computers & Operations Research*, 33(8):2353–2368, 2006.

[11] A. Setämaa-Kärkkäinen, K. Miettinen, and J. Vuori. Heuristic for a new multiobjective scheduling problem. *Optimization Letters*, 1(3):213–225, 2007.

[12] Q. Song and A. Jamalipour. An adaptive quality-of-service network selection mechanism for heterogeneous mobile networks. *Wireless Communications and Mobile Computing*, 5(6):697–708, 2005.

PIV

HEURISTIC FOR SELECTING NETWORK CONNECTIONS IN A DYNAMIC ENVIRONMENT

by

Anne Setämaa-Kärkkäinen 2008

Reports of the Department of Mathematical Information Technology, Series B.
Scientific Computing, No. B 9/2008

Heuristic for selecting network connections in a dynamic environment

Anne Setämaa-Kärkkäinen*

Abstract

In this paper we consider a system in which a mobile terminal can use several network connections simultaneously to transfer data. The problem addressed here is how to optimize the data transmission in a dynamic network environment where the transmission rates and prices can vary during the transmission. The objective is to minimize both the time used in transferring and the costs. To solve this multiobjective optimization problem we have developed a heuristic that reacts to changes in the environment. The heuristic is tested using simulations with the aim of defining when it is profitable to change the current transmission schedule.

Keywords: network connection selection, wireless networks, scheduling, multiobjective optimization, heuristics, simulations

1 Introduction

Nowadays, mobile terminals are able to use different kinds of network connections. In the presence of multiple network connections, the user of a mobile terminal can make the selection of the network connection to be used. This is known as the concept of being always best connected (ABC) [5]. Here we take a step further and consider future wireless networks where mobile terminals can use multiple network connections simultaneously to transfer data. Then a new problem arises: which connections to use for each data transmission and which connection to use for transferring each component of the data. We call this problem the network connection selection problem [12] and our objective is to minimize both the time used in transferring and the costs.

This kind of a problem is difficult to solve and users of mobile terminals are rarely interested in making this kind of a selection themselves. Therefore, we aim at developing an automatic method to solve the problem. The method should be fast and use only little computational capacity so that it is possible to implement it

*Department of Mathematical Information Technology, University of Jyväskylä, P.O. Box 35 (Agora), FI-40014 University of Jyväskylä, Finland, annseta@mit.jyu.fi

in mobile terminals. In addition, since wireless networks are usually very dynamic, the method should take into account changing conditions in the environment and improve the solution during the transmission when needed.

There are many studies considering wireless network environments where there are multiple network connections available, see for example [1, 2, 6, 7, 8, 14, 15]. These studies concentrate on selecting a network connection to be used for data transmission. For example, in [15], the selection of a network connection is formulated as a general multiobjective optimization problem. The problem is solved using the weighting method and the user of the mobile terminal is given freedom to choose the objectives, the constraints and the weights. In this method, the user needs to make a lot of difficult decisions. Especially defining the weights for the weighting method is difficult since the influence of the weights on the solution is not always clear [9].

In [2], the selection of a network connection is based on ranking the available network connections using multi-attribute decision making. The ranking of network connections is based on the properties of the network connections such as the quality of service properties. This kind of ranking could be used before solving our problem, that is, which network connection to use for each component of data. The ranking would select those network connections that have required quality of service properties such that it is possible and reasonable to send the data using them. In addition, the preferences of the user of the mobile terminal could be taken into account in this selection phase. The user could, for example, prohibit the use of a network connection if he/she finds it too expensive. However, such considerations are out of the scope of this paper. From now on, we assume that some kind of selection is made by the mobile device and the network connections we consider in our problem settings are all such that it is possible to use them to transfer the data in question.

The *network connection selection problem* was first introduced in [12] in which it was formulated as a multiobjective integer programming problem. The problem can be briefly defined as follows: Data consisting of separate components is to be transferred using a set of network connections. The problem is to select a network connection to each component of the data in such a way that the time used in transferring and the costs are minimized. When solving the network connection selection problem, a compromise between the costs and the time used in transferring has to be made. Since we are developing an automatic solution method also the compromise is made automatically. How a good compromise can be attained was studied in [12] and an achievement scalarizing function was found best for our purposes.

The achievement scalarizing function produces a good compromise between the conflicting objectives but it is time-consuming to minimize this function in our problem case. Therefore, a heuristic using the achievement scalarizing function to measure solution quality was developed [13]. The heuristic solves the problem fast and finds a solution near the good compromise produced by minimizing the scalarizing function. The heuristic assumes a static environment, which means that the rates and the prices of the network connections do not change during the transmission.

This kind of an assumption is valid only if the transmission time is short and the mobile device is not moving. In order to be more general, the static heuristic was improved to take into account changing environment in [11]. The dynamic heuristic presented uses the static heuristic to form an initial schedule which is followed when transferring the components. The shedule is improved (if possible) during the transmission when a change in the network conditions occurs. By a change we mean changes in the transmission rates or prices as well as changes in the availability of the network connections.

The aim of this paper is to specify cases in which the improvement phase of the dynamic heuristic should be run in order to optimize the performance of the heuristic. Though the improvement method developed is fast, it can slow down the transmission if done repeatedly without actual improvement in the schedule. In addition, running the improvement phase uses the limited resources available in a mobile terminal, such as the battery. Therefore, it is important to investigate when it is useful to try to improve the schedule. The purpose is to limit the overall time used in searching for a better schedule during the transmission. We use simulations to test when the improvement phase should be run. We have run 2000 simulations with 25 different problem instances. Within these simulations, we have tested four versions of the dynamic heuristic with the difference being when the improvement phase is run in the heuristic. The results obtained are also compared to the results of using the static heuristic. The results are visualized using performance profiles introduced in [4].

This paper is organised as follows. In the next section, the static heuristic is briefly described. The dynamic heuristic is presented in Section 3. The test instances used in the simulations are described in Section 4, and the simulation environment is presented in Section 5. In Sections 6 and 7, the results of the simulations are shown and discussed, and finally conclusions are drawn in Section 8.

2 Static heuristic

In the network connection selection problem, data consisting of n separate components is to be transferred using a set of m network connections. The problem is to select a network connection to each component of the data in such a way that the time used in transferring and the costs are minimized. We assume that the sizes of the components as well as the rate and the price of each connection are known. It should be noted that we do not define the actually routing of the data but only select which connection to use for transferring each component. In other words, the solution to the network connection selection problem tells on which network connection each component is transferred. For the actual transmission, we need a *transmission schedule* that tells when each component is transferred. We assume that no idle time is needed between transfers of different components, that is, the transfer of a component can begin as soon as the transfer of the previous component on the same connection is finished. Therefore, we only need to define the order of the components on each connection for the transmission schedule.

Minimizing the time and the costs are conflicting objectives since usually the faster the transfer is, the more expensive it is. Therefore, we aim at finding a compromise between the objectives. By a compromise we mean a solution where none of the objectives (that is, the time and the costs) can be improved without deteriorating the other. This concept is also known as *Pareto optimality*. The network connection selection problem should be solved fast, and the solution method should use only little computational capacity in order to make an implementation in mobile terminals possible. A heuristic for solving the network connection selection problem fast in a static environment was proposed in [13]. By a static environment we mean that there occur no changes in the network environment during the transmission of the data in question. We next give a brief presentation of the heuristic, which we call the static heuristic.

The heuristic is a local search method in which the solution quality is measured using an achievement scalarizing function [9] that incorporates the two objective function f_1 and f_2 (time and costs) [13]. The achievement scalarizing function to be minimized is the following:

$$\max_{i=1,2} w_i \frac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^*} + \rho \sum_{i=1}^{2} w_i \frac{f_i(x) - z_i^{mid}}{z_i^{nad} - z_i^*}, \tag{1}$$

where z^{mid} is a middle point located in the middle of the ranges of the objective functions, that is,

$$z_i^{mid} = \frac{z_i^{nad} + z_i^*}{2}$$

for $i = 1, 2$, and z^* and z^{nad} are the best and worst objective function values in the set of compromise solutions, respectively. The coefficient ρ is a small positive scalar, and the ratio of the positive weighting coefficients w_1 and w_2 represents the rate at which the user is willing to trade off values of the objective functions. We have used a value 0.001 for the coefficient ρ and values 1 and 2 for the weighting coefficients w_1 and w_2, respectively (see [12] for more information). The denominators in the scalarizing function are used to scale the objective function f_1 and f_2 to be of a similar magnitude. The smaller the value of the achievement scalarizing function is, the better the solution is.

The scalarizing function needs information on the ranges of the objective functions, that is, the best and worst values in the set of compromise solutions. It is time-consuming to calculate them and because they are used mainly for scaling the objective functions, we use approximations. We use 0 to approximate the best objective function value for both the time and the costs, and the worst objective function values are estimated from the problem data. More information about the estimates can be found in [13] and more information on the scalarizing function in [12].

The static heuristic can be summarized as follows:

1. *Initial solution*. For each component $j = 1, \ldots, n$, find the network connection that gives the lowest value of the scalarizing function (1) for the current partial problem, and assign the component to that connection. The current *partial*

problem consists of component j and the components that are already assigned (with fixed assignments). All the connections are included in the partial problem, and the best and worst objective function values are approximated for each partial problem separately.

2. Set the initial solution formed in the previous step as the current solution.

3. *Improvements*. Repeat twice: For each component $j = 1, \ldots, n$, use 1-exchange to improve the current solution: Consider transferring component j using another connection than the one it is currently assigned to. Select the connection that gives the lowest value of the scalarizing function (1) when component j is transferred using it and all the other assignments are fixed. If the value of the scalarizing function is smaller than in the current solution, set component j to the selected connection and set this solution as the current solution.

4. *Ordering*. Order the components assigned to each network connection according to some criteria, for example, the size. The components will be transferred in that order.

It should be noted that originally in [13] the improvement round (Step 3) is repeated until a stopping criterion (not explicitly defined) is satisfied. Because of the computational results presented in [13], we repeat the improvement round twice.

Note also that neither the time used in transferring nor the costs depend on the order of the components on each network connections. Therefore, the order can be discarded when the components are assigned to the connections. However, the order is needed in a transmission schedule telling when each component is to be transferred. To form the transmission schedule the components are ordered on each connection after the components are assigned to the connections.

3 Dynamic heuristic

In wireless networks, the transmission rate can vary a lot during a transmission due to, for example, congestion. It is also possible that a network connection is lost during a transmission when a mobile terminal moves. In this kind of a situation, it is necessary to be prepared to change the transmission schedule in accordance to the changing environment. It has been proposed that pricing could be used to control congestion [3, 10]. Therefore, in addition to changing transmission rates, also pricing could vary during a transmission. Though it might be unlikely that the pricing of network connections would change during a transmission, we have taken into account also this aspect.

We have improved the heuristic presented in the previous section to take into account a changing network environment [11]. This version of the heuristic is called the dynamic heuristic. The dynamic heuristic reacts to changes in the transmission rates and prices and aims at improving the current transmission schedule rapidly during the transmission. The dynamic heuristic was briefly introduced in [11]. We

next present the dynamic heuristic in detail and discuss adjusting the parameters of the heuristic using simulations.

The dynamic heuristic uses the static heuristic to form the transmission schedule which is followed when the data is transferred. During the transmission, the dynamic heuristic tries to improve the schedule using rescheduling when a change in the environment has occured. The rescheduling phase considers a *partial problem* that consists of all the connections currently available and the components that have not yet been transferred. These components are collected into a set C and the connections available form a set N. Thus, the partial problem consists of assigning the components in C to the connections in N. In addition to the set C, a set D is used for components that are assigned to lost connections and need to be reassigned to other connections.

The rescheduling phase works as follows:

1. *Initialization*. Set $C = \emptyset$, $D = \emptyset$ and $N = \emptyset$. Control which connections are currently available and add those connections to the set N. Add to the set C all the components that are waiting to be transferred using the connections in N. Remember the current assignment of the components to the connections.

2. *Initial solution*. Form an initial solution to the partial problem:

   - If there are no lost connections, the initial solution is the current assignment of the components in C to the connections in N. Go to Step 3.
   - Otherwise, add all the components assigned to the lost connections in the set D and remove the components from the transmission schedule. If a transfer of a component was interrupted when a connection was lost, remove also this component from the transmission schedule and add it to the set D.
   - Form the initial solution. For each component j in D: Remove the component j from the set D and add it to the set C. Assign the component j to the network connection i in N that gives the lowest value of the scalarizing function (1) to the partial problem consisting of the components in C and the connections in N when the component j is assigned to the connection i and the assignments of the other components in C are fixed.

3. *Rescheduling*. Set the initial solution as the current solution. Use rescheduling to each component j in C: Consider a set of solutions where the component j is assigned to another connection instead of the connection to which it is assigned in the current solution. Select among the solutions the one having the lowest value of the scalarizing function (1) to be a *rescheduling solution*. If the rescheduling solution has a lower value of the scalarizing function (1) than the current solution, set the rescheduling solution as the current solution.

When a change in the environment occurs, the rescheduling phase is run. The transmission of those components that are currently being transferred is continued

(unless the transmission was interrupted because of a lost connection). The components that are waiting to be transferred are kept waiting until the rescheduling phase is run and a new schedule, that is, the current solution in Step 3, is formed. After the resceduling phase is run the transmission is continued and a new transmission schedule is followed if the transmission schedule was changed by the improvement phase.

When a component is moved from a connection to another in the rescheduling phase the transmission schedule changes. We do not order the components on each network connection after rescheduling but a component assigned to another connections is moved to the last position on the new connection in the transmission schedule. If a certain order is preferred the components can be ordered after the rescheduling phase.

It should be noted that we have not specified the order in which the components are dealt with in the heuristics. In [13], we tested a random order and a decreasing order of the components' sizes in the static heuristic. The results were very similar in both of the cases and, therefore, we have used a random order in our simulations. However, note that the order is kept the same during the entire heuristic. In other words, the random order we use in forming the initial solution in Step 1 of the static heuristic is also used in Step 3 of the static heuristic as well as in the rescheduling phase every time it is run.

Though the rescheduling phase is simple and fast, it is not necessarily wise to run it every time a change occurs. Therefore, the rescheduling phase should be run only when a connection is lost or a new connection appears or the change in the transmission rates or prices is big enough. It is not straightforward to define when a change is big enough. Therefore, we have tested the dynamic heuristic using simulations. In the tests, we have used three different values for *Tolerance* which tells the percentage how much a rate or a price must be changed before the rescheduling phase is run. If there occurs multiple changes simultaneously, the largest percentual change is decisive. The simulation environment and the results are presented in Sections 5 and 6, respectively. Before that we discuss the different pricing models and the test instances we have used in the simulations.

4 Test instances

Pricing of the future wireless networks is not naturally known today. Therefore, we have used three different kinds of pricing models in the simulations. These pricing models represent possible future pricing models that could be used in wireless networks.

The first function g_1 models the price being linearly dependent on the transmission rate B. Thus, the function is

$$g_1(B) = k_1 B,$$

where k_1 is a connection-specific coefficient. The higher the transmission rate is, the higher the price per bit is. In the second function g_2, the price has an exponential

relation to the transmission rate:

$$g_2(B) = k_1 \exp(k_2 B),$$

where k_1 and k_2 are connection-specific coefficients. This function models a situation where high-speed connections are extremely expensive. The reason for this may be technological or a vigorous attempt to prevent congestion in networks. The third function g_3 is the most often used model for network pricing [10]. It is a function in which the exponent is less than one. In our function, the exponent is 0.5. Thus, we have

$$g_3(B) = k_1 \sqrt{B},$$

where k_1 is again a connection-specific coefficient. In this function, the price is less strongly dependent on the transmission rate than in functions g_1 and g_2.

Based on these pricing models we have built a set of test instances for the network connection selection problem. The test instances were used in simulations and the results are presented in Section 6. In the test instances, we have calculated the costs of transferring component j through network connection i (denoted by c_{ij}) using formula

$$c_{ij} = [g(B_i) \cdot S_j + k] \cdot 10^{-5}, \tag{2}$$

where S_j is the size of component j in bits, B_i is the transmission rate of connection i (bits/s), $k \geq 0$ represents fixed costs of the transfer, and g is one of the functions g_1, g_2, and g_3. In order to define the parameters in the pricing models, we have gathered current data transmission prices for GSM and UMTS networks in Finland and fitted those prices into our pricing models using the method of least squares. This gives us fictitious pricing models with some resemblance to reality.

We have formed five different sets of network connections to be used in the simulations. The sets are the following: In the first set having three network connections, the prices are calculated using pricing function g_1. The values used for parameter k_1 are $5.6 \cdot 10^{-7}$, $6.1 \cdot 10^{-7}$, and $7.8 \cdot 10^{-7}$, whereas parameter k is given value 0. The (original) rates of the three connections are 14 400, 115 200, and 384 000 bit/s, respectively. The second set contains five network connections of which three are the same as in the first set. The two additional connections having the rates of 14 400 and 115 200 bit/s, respectively, are also priced using function g_1. The values used for parameter k_1 are now $5.7 \cdot 10^{-7}$ and $5.9 \cdot 10^{-7}$, and parameter k is again given value 0. The third set of connections consists of three network connections with the rates of 14 400, 32 000, and 56 000 bit/s that are priced using function g_2. The parameter values used in pricing are given in Table 1 in a row marked by a. The fourth set includes the same connections as the third set and two connections with the rates of 56 000 and 384 000 bit/s that use the same pricing function g_2 with parameters b given in Table 1. The last set contains eight connections that are priced using function g_3. The parameter values used in pricing are again given in Table 1. Three of the connections with the rates of 14 400, 59 200, and 384 000 bit/s use parameters given in row c, three connections with the rates of 14 400, 59 200, and 115 000 bit/s use parameters d, and the last two connections with the rates of 14 400 and 59 200 bit/s use parameters e in the pricing model.

Function		Parameters		
		k	k_1	k_2
g_2	a	9	0.04	$4.3 \cdot 10^{-5}$
	b	6	1.1	$1.7 \cdot 10^{-6}$
g_3	c	300	$4.3 \cdot 10^{-3}$	
	d	100	$9.0 \cdot 10^{-3}$	
	e	30	$3.1 \cdot 10^{-3}$	

Table 1: The parameter values used in the pricing functions g_2 and g_3 in the test instances.

For each set of network connections presented above we have (randomly) selected five web pages from the Internet in order to form five test instances. This way we have obtained 25 test instances that are listed in Table 2. In the table, the size of each test instance, the set of network connections and the pricing model used are given, as well as the total size (in bits) of the data to be transferred. We next present the simulation environment in which the test instances were used, and then we discuss the simulation results.

5 Simulation environment

The performance of the heuristic in a dynamic environment was tested using a simulator. We have made some simplifying assumptions in the simulator in order to reduce its complexity. The assumptions and the parameters used in the simulator are the following.

We have run two sets of simulations, with fixed pricing and with dynamic pricing. In the first set of simulations, the pricing of the network connections is fixed while the rates of the connections change. In these simulations, we use the maximum rates of the connections (given by the operators) when calculating the *fixed prices*. In addition to changing rates, the network connections can also disappear and later appear again. A network connection disappears randomly with a probability of 0.01. Every time a change appears on a network connection, the network connection may disappear (or appear if it has not been available) depending on a random number generated from a uniform distribution between zero and one. When a connection is lost during a transmission of a component, the component has to be transferred again from the beginning. In this case, the transmission of the rest of the component is discarded. Also the price of transferring the component on the lost connection is discarded. Thus, it is assumed that if a transmission of a component fails, we do not have to pay for that. When a network connection is available, the transmission rate is assumed to be uniformly distributed between the original rate divided by three and the original rate. By an *original rate* we mean the rate given by the operator of the network connection, which is the maximum rate of the connection. (The original rates of the connections used in the test instances are

Instance	n	m	Set	Pricing model	Total size
siam	6	3	3	g_2	32 299 bits
cplus	11	3	1	g_1	70 672 bits
lyx	11	5	2	g_1	51 515 bits
geta	12	3	3	g_2	100 348 bits
taylor	12	5	4	g_2	59 714 bits
zoo	13	5	4	g_2	151 705 bits
lth	13	8	5	g_3	236 729 bits
yahoo	14	3	1	g_1	85 672 bits
it	15	5	2	g_1	74 421 bits
ilmat	16	3	3	g_2	67 752 bits
sana	17	3	1	g_1	136 763 bits
ebbas	18	8	5	g_3	161 960 bits
prh	19	5	2	g_1	72 162 bits
ieee	21	3	3	g_2	128 507 bits
springer	23	8	5	g_3	87 605 bits
sjukvard	24	5	4	g_2	98 961 bits
cph	27	3	1	g_1	191 497 bits
svt	27	8	5	g_3	358 081 bits
tivoli	28	5	4	g_2	332 754 bits
yle	33	5	2	g_1	203 835 bits
subtv	42	3	3	g_2	828 898 bits
skane	44	3	1	g_1	229 234 bits
uutiset	54	5	4	g_2	163 268 bits
mtv	70	5	2	g_1	398 349 bits
nelonen	86	8	5	g_3	415 801 bits

Table 2: The test instances: the size (n components and m connections), the set of network connections and the pricing model used as well as the size of the data to be transferred in bits.

given in Section 4.)

In the second set of simulations, also the pricing varies. The price of a connection is dependent on the transmission rate, and when the transmission rate changes also the price changes accordingly. We call this situation *dynamic pricing*. The change in the transmission rate is observed immediately, and the rate can change during a transmission of a component. However, the price is always fixed during a transmission of a component, and the new price is valid from the transmission of the next component on that connection. This assumption is made because it would not be fair to customers to change a price in such a way that customers could not decide if they still wanted to use the connection with the new price. All the other assumptions and parameters are identical for the two sets of simulations.

The occurrence of a change in a transmission rate is exponentially distributed with a parameter λ. The parameter λ is called a rate parameter and it means that a transmission rate changes λ times in a time unit. The rate parameter λ is given values 0.2 and 0.7 in our simulations. Using $\lambda = 0.2$ represents a situation where the mobile device is moving with a slow rate, for example, when the user of the device walks. The parameter value $\lambda = 0.7$ is valid in a situation where the mobile device moves rapidly, for example, when the device is in a car that travels fast. The same value of λ is used for each connection in a simulation. It should be noted that the parameter λ could be different for each connection. In that case, the rate parameter would reflect the stability of the connection. In other words, the smaller the parameter the more stable the connection is.

As already mentioned, changes in the transmission rates are observed immediately. Thus, there is no delay in observing the changes. The time it takes to run the rescheduling phase is assumed to be negligible and it is not considered during the simulations. Running the rescheduling phase took less than 0.01 seconds in our experiments, so the assumption is justified. However, if the rescheduling phase is run many times during a transmission, the time used in it becomes significant. Therefore, we aim at finding a balance in which the rescheduling phase is not run too often and the transmission time and the costs are kept at a low level.

In our simulations, we have transferred data using two different strategies. The first strategy is to follow the transmission schedule produced by the static heuristic during the entire transmission. In other words, the assignments of the components to the connections are fixed in this transmission. In this case, when a connection is lost the components assigned to the lost connection that are not yet transferred are kept waiting for the return of the connection. If a transmission of a component was interrupted when the connection was lost, the transmission has to be repeated from the beginning when the connection returns. The other strategy uses the dynamic heuristic and the rescheduling phase to improve the solution when a change occurs, if the change is large enough. Based on the simulation results we compare the performance of these two strategies and, thus, the performance of the static and dynamic heuristics.

The test instances used in the simulations were presented in the previous section. For each test instance we have run 80 simulations with different random number

seeds. 40 simulations were run with the fixed pricing and 40 simulations with the dynamic pricing. We have used two values (0.2 and 0.7) for the parameter λ in the simulations and, thus, we have run 20 simulations for each value of the parameter λ and for each pricing. The dynamic heuristic was used with four different values of parameter Tolerance, which is the percentage how much a rate or a price has to be changed before the rescheduling phase is run. The values used for the parameter are 10%, 20%, 30%, and 40%. The simulations were run in a computer with a relatively slow 550 MHZ PA-RISC processor. In the near future, mobile terminals will have computational power similar to the processor we have used in the simulations.

6 Computational results

The simulation results are presented in Tables 3 and 4 and in Figures 1–3. The complete tables of the simulation results are omitted here because they are large and thereby it is more convenient for the reader to provide summaries of the results. In order to illustrate and analyse the simulation results, we have calculated performance profiles of the heuristics. A *performance profile* is a distribution function of a performance metric introduced in [4] for comparing optimization solvers. Let us assume we have a set S of n_s solvers that are used for solving a set P of n_p problems. A *performance ratio* $r_{p,s}$ that compares the performance of a solver s to the best performance of the solvers in S in the case of a problem p can be calculated as follows:

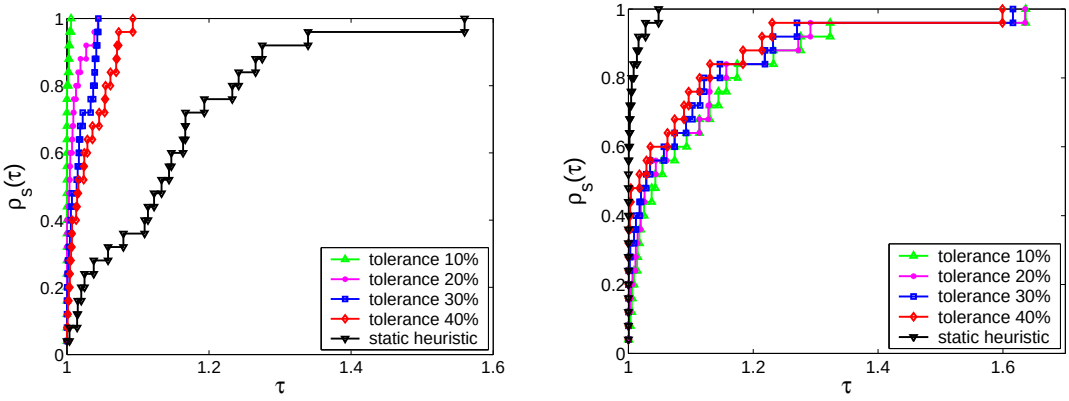
$$r_{p,s} = \frac{t_{p,s}}{min\{t_{p,s} : s \in S\}},$$

where $t_{p,s}$ is the value of the performance measure for solver s in the case of problem p. Note that here we assume that the smaller the value of the performance measure is the better the performance is. Now, the probability $\rho_s(\tau)$ that a performance ratio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio is

$$\rho_s(\tau) = \frac{1}{n_p} size\{p \in P : r_{p,s} \leq \tau\},$$

where $size$ is the cardinality of the following set, that is, the number of elements in the set. Function $\rho_s(\tau)$ is the distribution function of the performance ratio and, thus, gives the performance profile for solver s.

In our case, the solvers in S are the static heuristic and the four versions of the dynamic heuristic (with the different values of the parameter Tolerance). The problems in P are the 25 test instances used in the simulations, and the results of the simulations are being analysed. We have used the scalarizing function (1) as the performance measure. In addition to that we have also calculated performance profiles using the original objective functions (that is, the time used in transferring and the costs of the transmission) as the performance measure. It should be noted that since we are analysing simulation results, the values for the performance measures are averages of 20 simulations.



(a) Time as the performance measure.

(b) Costs as the performance measure.

Figure 1: Performance profiles with the time/costs as the performance measure: fixed pricing, $\lambda = 0.7$.

The performance profiles in which the time or the costs are used as the performance measure show that the objective functions are truly conflicting. Unfortunately, based on those performance profiles no strict preference between the different versions of the heuristic can be stated. Therefore, we show those performance profiles in Figure 1 only for the simulation case in which the fixed pricing was used with the parameter value $\lambda = 0.7$. The performance profiles for the three other cases (fixed pricing with $\lambda = 0.2$ and dynamic pricing with $\lambda = 0.2$ and $\lambda = 0.7$) with the time or the costs being the performance measure are similar to those in Figure 1. As already mentioned, Figure 1 illustrates the fact that the objectives are conflicting: If we want to keep the costs as low as possible, the static heuristic performs best. However, the performance of the static heuristic is the worst when we consider the time used in transferring. The performance profiles in Figure 1 show that the larger the value of the parameter Tolerance is, the closer the performance of the dynamic heuristic is to the performance of the static heuristic, as can be expected. However, there is not much difference in the performance between the different versions of the dynamic heuristic, and no strict preferences can be stated based on these profiles.

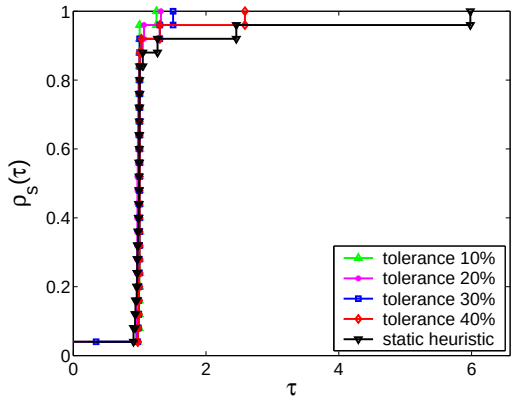
The performance profiles in which the scalarizing function (1) is used as the performance measure are shown in Figures 2 and 3. Figure 2 shows the performance profiles of the different versions of the heuristic for the four different simulation cases: with the fixed or dynamic pricing and with the value 0.2 or 0.7 for the parameter Tolerance. The performance profiles over all the simulations are shown in Figure 3. In Figure 3(a) a general view of the performance profiles is shown, whereas Figure 3(b) is a magnification of Figure 3(a) concentrating on the difference between the different versions of the dynamic heuristic. In the calculations of the scalarizing function values, we have used estimated values for the best and worst objective function values, z^* and z^{nad}. The best value used was zero in all the test instances and for the both objective functions. The worst value was estimated for each test in-

stance separately. Because the rates of the connections change during simulations, we used the maximum rates of the connections when estimating the worst values. Similarly, the prices used when estimating the worst values were calculated using the maximum rates. This way the scalarizing function values were more comparable among different simulations of each test instance.

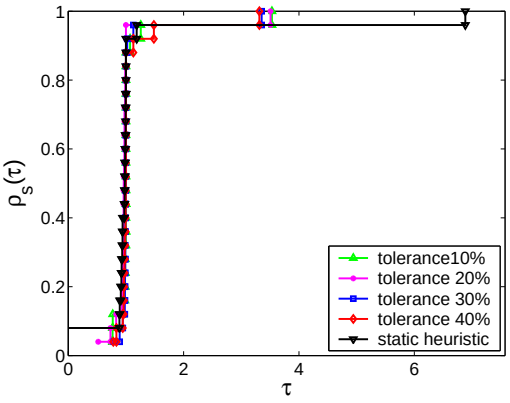
As in Figure 1, we can see in Figure 3 that the larger the value of the parameter Tolerance is, the closer the performance of the dynamic heuristic is to the performance of the static heuristic. However, when the scalarizing function (1) is used to measure the performance, the performance profiles show that the dynamic heuristic is performing better than the static heuristic. Thus, using the improvement phase of the dynamic heuristic during transmissions has an positive effect on the solution quality. The differences between the different versions of the dynamic heuristic are not very large. The performance of the dynamic heuristic with value 10% for the parameter Tolerance was surprisingly good, especially when the dynamic pricing was used. However, the difference to the results with other values for the parameter Tolerance is not very large, as already mentioned, and no strict preference can be stated based on the performance profiles.

In order to give some numerical results, we have collected the average relative differences to the static heuristic among all the test instances and simulations for different values of the parameter Tolerance in Table 3. Table 4, on the other hand, presents the average number of times the improvement phase is run in the dynamic heuristic during the simulations. As in the performance profiles, there is not much difference between the results with different values of Tolerance when the average relative difference is considered. However, the average number of times the improvement phase is run is lower for large values of Tolerance. This means that the time and the effort used in improving the transmission schedule is lower which is preferable when we consider the limited amount of capacity available in mobile terminals. This implies that a larger value (30% or 40%) is preferable than a lower one (10% or 20%). However, similarly as with the performance profiles, we cannot claim any strict preferences between the different values of the parameter Tolerance based on Tables 3 and 4.

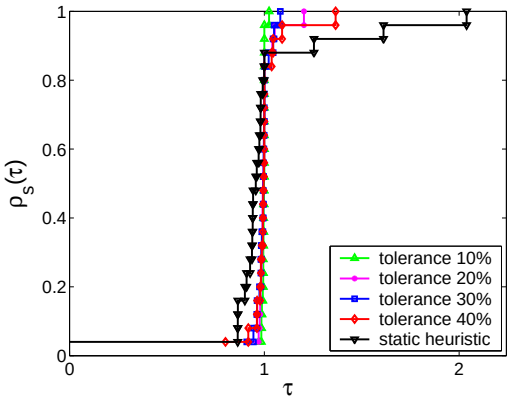
The performance profiles and Tables 3 and 4 illustrate the overall performance of the heuristics. However, they do not reveal any information about the transmission times or costs obtained in the simulations. Therefore, we have collected simulation results of five test instances in Tables 5 and 6. These instances were selected in such a way that the results show the general trends in the simulation results and each set of network connections is represented by an instance. Table 5 contains the average time used in transferring and the average costs of the transmission. Table 6 presents the average number of times the improvement phase is run in the dynamic heuristic. These averages are calculated over 20 simulations in which we have used the fixed pricing and the value 0.2 for the parameter λ (which tells how many times a transmission rate changes in a time unit).



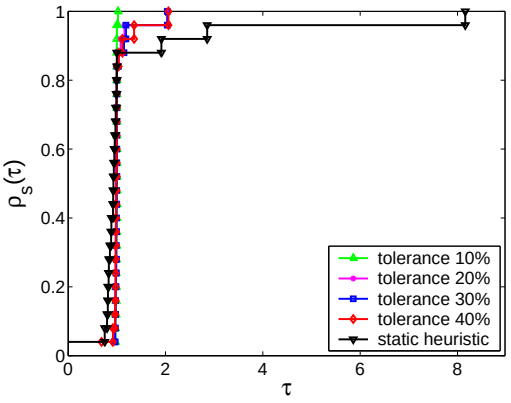
(a) Fixed pricing, $\lambda = 0.2$.



(b) Fixed pricing, $\lambda = 0.7$.

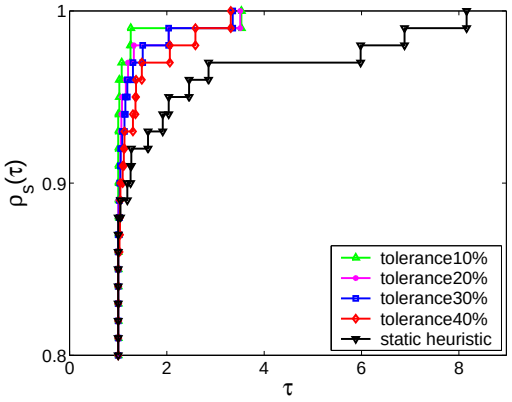


(c) Dynamic pricing, $\lambda = 0.2$.

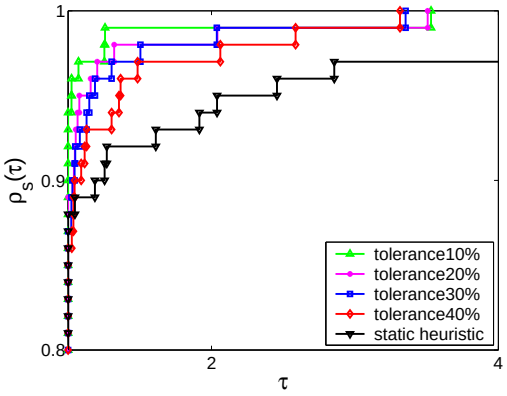


(d) Dynamic pricing, $\lambda = 0.7$.

Figure 2: Performance profiles with the scalarizing function as the performance measure.



(a) General view of the profiles.



(b) Magnification of a part of the profiles.

Figure 3: Performance profiles with the scalarizing function as the performance measure: the results of all the simulations.

Pricing	λ	Tolerance 10% time	Tolerance 10% costs	Tolerance 20% time	Tolerance 20% costs	Tolerance 30% time	Tolerance 30% costs	Tolerance 40% time	Tolerance 40% costs
Fixed	0.2	-0.06	0.04	-0.05	0.04	-0.05	0.03	-0.04	0.03
Fixed	0.7	-0.11	0.10	-0.10	0.09	-0.09	0.08	-0.08	0.07
Dynamic	0.2	-0.06	0.04	-0.05	0.03	-0.05	0.03	-0.05	0.03
Dynamic	0.7	-0.11	0.10	-0.11	0.09	-0.10	0.08	-0.09	0.07

Table 3: The average relative difference to the static heuristic.

Pricing	λ	Tolerance 10%	Tolerance 20%	Tolerance 30%	Tolerance 40%
Fixed	0.2	1.92	1.59	1.27	0.99
Fixed	0.7	5.59	4.62	3.75	2.90
Dynamic	0.2	1.95	1.63	1.31	1.06
Dynamic	0.7	6.12	5.06	4.10	3.18

Table 4: The average number of times the improvement phase is run.

	Tolerance 10%		Tolerance 20%		Tolerance 30%		Tolerance 40%		Static heuristic	
Instance	time	costs	time	costs	time	costs	time	costs	time	costs
cplus	0.9481	0.0528	0.9481	0.0528	0.9481	0.0528	0.9481	0.0528	0.9531	0.0519
prh	0.5028	0.0468	0.5028	0.0468	0.5028	0.0468	0.5029	0.0468	0.5030	0.0468
tivoli	4.4986	1.7146	4.4232	1.7459	4.4656	1.7923	4.4723	1.8048	4.5256	1.8044
subtv	20.7401	1.8514	21.1148	1.8044	22.6895	1.7285	23.3428	1.6696	32.6348	1.2323
nelonen	4.7090	3.5997	4.7115	3.6018	4.7608	3.5965	4.7180	3.6098	5.5763	3.5906

Table 5: Simulation results with fixed pricing and $\lambda = 0.2$: the average duration and costs of the transmission.

Instance	**Tolerance 10%**	**Tolerance 20%**	**Tolerance 30%**	**Tolerance 40%**
cplus	0.25	0.25	0.20	0.20
prh	0.40	0.35	0.25	0.15
tivoli	3.40	2.90	1.85	1.30
subtv	10.50	8.50	7.25	5.65
nelonen	6.85	5.70	4.40	3.15

Table 6: Simulation results with fixed pricing and $\lambda = 0.2$: the average number of times the improvement phase is run.

7 Discussion

In most of the test instances the simulation results are similar: The dynamic heuristic keeps the time needed for transferring shorter than the static heuristic but the costs are a bit higher. This is valid for both the fixed pricing and the dynamic pricing. The reason for this may be that the static heuristic assigns most of the components to the cheapest connections which often have low transmission rates. When a cheap connection becomes slower (or is lost) the dynamic heuristic moves components from that connection to a faster connection which is also more expensive. This way the costs increase while the time used in transferring is kept short. It should be remembered that the results we have dealt with are averages of a set of simulations and the result of a single simulation can be different from this average.

The difference between the static heuristic and the four versions of the dynamic heuristic can be invisible in practice, such as in the test instances cplus and prh in Table 5. On the other hand, in the case of the test instance subtv the difference is large, 9–12 seconds and 0.4–0.6 euros. Similarly, the difference in the number of times the improvement phase is run can be small or large. However, the performance profiles shown in Figure 3 imply that the dynamic heuristic performs better than the static heuristic with all the four tested values for the parameter Tolerance, at least when the scalarizing function (1) is used to measure the performance. Thus, using the improvement phase of the dynamic heuristic during transmissions is profitable. However, it is more problematic to decide when the improvement phase should be run.

Since the simulation results of the dynamic heuristic with different values for the parameter Tolerance are very similar, conclusions have to be made mainly based on the number of times the improvement phase is run in the dynamic heuristic. Each time the improvement phase is run the limited capacities of a mobile terminal, such as the battery, is used. In addition, if the improvement phase is run many times during a transmission it can delay the transmission. Changing the schedule constantly is also not preferred. Therefore, based on the simulation results, we suggest that the default value of the parameter Tolerance is 30%. Then, the improvement phase of the dynamic heuristic is run when a network connection is lost, a new connection appears or a change in the transmission rates or prices is more than 30 percent of the previous rate or price. (Note that the value 40% could also be selected but we prefer the value 30% because the performance profile of the latter is slightly better.)

The simulation results imply that it would be important to know the preferences of the user of the mobile terminal. However, it is not convenient to ask the user to define preferences during the transmission, and not even before each transmission. The best way to get some preference information would be to allow the user to adjust the preferences in the mobile terminal when wanted. This way those users that are not interested in giving any preferences need not to think about them and others can adjust the preferences whenever they want to do that. The preferences could be, for example, denying the use of some network connections (because of high costs or other factors) or declaring that the costs should always be as low as

possible. The latter case would imply, based on the simulation results, that the improvement phase is not run unless necessary, that is, when a connection is lost or a new connection appears. Naturally the improvement phase could be run also when the costs change but then the transmission schedule would be changed only if the costs were lower in the new schedule.

Thus, a topic of future research is to study how the user could easily give some preference information without being involved in the solution procedure and how this information could be used in the dynamic heuristic. The user could, for example, adjust a *user profile* that would define preferences regarding the use of different network connections. The user profile could contain explicit preferences, such as a field for denying the use of some connections, or implicit preferences that would express a trade-off between the time used in transferring and the costs. The user profile could also contain a field for adjusting the weighting coefficients of the scalarizing function (1). Adjusting them should however be done with a caution though the effect of the weighting coefficients is not as large as in the well-known weighting method [9, 12]. This kind of preference information could help in making rules when the improvement phase should be run. However, before that we need to study how this information could be easily given in a user profile.

8 Conclusions

In this paper, we have considered solving a network connection selection problem in a dynamic environment. The network connection selection problem consists of selecting a network connection for each component of data in such a way that the time used in transferring and the costs are minimized. We have presented a dynamic heuristic that reacts to changing network conditions and aims at improving the solution. The main question in this dynamic heuristic is when it is profitable to try to improve the solution. We have addressed this issue by running simulations and comparing different versions of the heuristic. The differences in the results between the different versions are small, and some kind of preference information from a user would be helpful. Therefore, we will continue our research by studying how the user could define preferences in a user profile in the mobile terminal without being involved in the actual process of selecting connections. Then, the preference information could be used when defining how often the improvement method in the dynamic heuristic is run during a transmission.

Acknowledgements

This research was partially supported by the Tekniikan Edistämissäätiö foundation and the Ellen and Artturi Nyyssönen foundation. The author wishes to thank Prof. Kaisa Miettinen for valuable comments and helpful discussions and Dr. Jarkko Vuori for providing the pricing models and helping with planning the simulations.

References

[1] E. Adamopoulou, K. Demestichas, A. Koutsorodi, and M. Theologou. Intelligent access network selection in heterogeneous networks - simulation results. In *Proc. of the 2nd International Symposium on Wireless Communication Systems*, pages 279–283, 2005.

[2] F. Bari and V. C. M. Leung. Automated network selection in a heterogeneous wireless network environment. *IEEE Network*, 21(1):34–40, 2007.

[3] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang. Pricing in computer networks: motivation, formulation, and example. *IEEE/ACM Transactions on Networking*, 1(6):614–627, 1993.

[4] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[5] E. Gustafsson and A. Jonsson. Always best connected. *IEEE Wireless Communications*, 10(1):49–55, 2003.

[6] A. Iera, A. Molinaro, C. Campolo, and M. Amadeo. An access network selection algorithm dynamically adapted to user needs and preferences. In *Proc. of the 17th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2006.

[7] J. H. Li, S. Luo, S. Das, T. McAuley, M. Patel, A. Staikos, and M. Gerla. An integrated framework for seamless soft handoff in ad hoc networks. In *Proc. of the Military Communications Conference*, 2006.

[8] L. Liang, H. Wang, and P. Zhang. Net utility-based network selection scheme in CDMA cellular/WLAN integrated networks. In *Proc. of IEEE Wireless Communications and Networking Conference*, pages 3313–3317, 2007.

[9] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[10] I. Ch. Paschalidis and J. N. Tsitsiklis. Congestion-dependent pricing of network services. *IEEE/ACM Transactions on Networking*, 8(2):171–184, 2000.

[11] A. Setämaa-Kärkkäinen and J. Kurhinen. Optimal usage of multiple network connections. In *Proc. of Mobilware'08*, to appear.

[12] A. Setämaa-Kärkkäinen, K. Miettinen, and J. Vuori. Best compromise solution for a new multiobjective scheduling problem. *Computers & Operations Research*, 33(8):2353–2368, 2006.

[13] A. Setämaa-Kärkkäinen, K. Miettinen, and J. Vuori. Heuristic for a new multiobjective scheduling problem. *Optimization Letters*, 1(3):213–225, 2007.

[14] Q. Song and A. Jamalipour. An adaptive quality-of-service network selection mechanism for heterogeneous mobile networks. *Wireless Communications and Mobile Computing*, 5(6):697–708, 2005.

[15] C. Yiping and Y. Yuhang. A new 4G architecture providing multimode terminals always best connected services. *IEEE Wireless Communications*, 14(2):36–41, 2007.