RAIRO Operations Research
RAIRO Oper. Res. **35** (2001) 1-20

BOTTLENECK CAPACITY EXPANSION PROBLEMS WITH GENERAL BUDGET CONSTRAINTS [*]

Rainer E. Burkard[1], Bettina Klinz[1] and Jianzhong Zhang[2]

Abstract. This paper presents a unified approach for bottleneck capacity expansion problems. In the bottleneck capacity expansion problem, BCEP, we are given a finite ground set E, a family $\mathcal{F}$ of feasible subsets of E and a nonnegative real capacity $\widehat{c}_e$ for all $e \in E$. Moreover, we are given monotone increasing cost functions f_e for increasing the capacity of the elements $e \in E$ as well as a budget B. The task is to determine new capacities $c_e \geq \widehat{c}_e$ such that the objective function given by $\max_{F\in\mathcal{F}} \min_{e\in F} c_e$ is maximized under the side constraint that the overall expansion cost does not exceed the budget B. We introduce an algebraic model for defining the overall expansion cost and for formulating the budget constraint. This models allows to capture various types of budget constraints in one general model. Moreover, we discuss solution approaches for the general bottleneck capacity expansion problem. For an important subclass of bottleneck capacity expansion problems we propose algorithms which perform a strongly polynomial number of steps. In this manner we generalize and improve a recent result of Zhang *et al.* [15].

Keywords: Capacity expansion, bottleneck problem, strongly polynomial algorithm, algebraic optimization.

Mathematics Subject Classification. 90C57, 90C31, 90C32.

Received May, 2000. Accepted February, 2001.

[*] *The first and the second author acknowledge financial support by the Spezialforschungsbereich F 003 "Optimierung und Kontrolle", Projektbereich Diskrete Optimierung. The third author acknowledges partial support of Hong Kong University Grant Council under the grant CITYU-9040465.*

[1] Institut für Mathematik B, TU Graz, Steyrergasse 30, 8010 Graz, Austria;
e-mail: `burkard@opt.math.tu-graz.ac.at`, `klinz@opt.math.tu-graz.ac.at`

[2] Department of Mathematics, City University of Hong Kong, Hong Kong;
e-mail: `mazhang@cityu.hk.edu`

1. Introduction

In many practical applications we are faced with the situation that we wish to expand the performance of a system while keeping the necessary investments within a prespecified budget limit. A typical example of this type arises if we wish to adjust a road network to increasing traffic demands by increasing the capacities of the roads. Similar expansion problems occur in connection with enlarging the capacity in telecommunication networks and with expanding the capacity of production processes.

Common to a large class of these capacity expansion problems is that there is an underlying combinatorial structure, composed of a ground set E and a set $\mathcal{F}$ of feasible subsets of E. The elements of the ground set may represent roads, communication links, production steps, etc. Each element $e \in E$ is associated with a capacity $\widehat{c}_e$ which may be increased to meet increasing needs. For the class of bottleneck capacity expansion problems investigated in this paper the capacity of a feasible set $F \in \mathcal{F}$ is given by the smallest capacity of an element in F, and the capacity of the combinatorial structure $(E, \mathcal{F})$ is defined to be the maximal capacity of a feasible set $F \in \mathcal{F}$.

To expand (increase) the capacity of the structure $(E, \mathcal{F})$, we need to increase the capacities $\widehat{c}_e$, $e \in E$. We assume that increasing the capacity of element e to the value $t > \widehat{c}_e$ incurs a cost of $f_e(t)$ where f_e is a monotone increasing function. This cost may, for example, measure the monetary investments required for increasing the capacity of element e, or the time which is needed for performing the expansion, or a combination of both. Moreover, we are given a budget B. The task in the bottleneck capacity expansion problem consists in determining new increased capacities $c_e \geq \widehat{c}_e$ so as to increase the capacity of the structure $(E, \mathcal{F})$ as much as possible while keeping the overall expansion cost within the budget limit B. Depending on the actual application, the overall expansion cost may be defined in various different ways. For example, if the costs f_e represent monetary investments, then a natural measure for the overall expansion cost will be the sum of the individual expansion costs $f_e(c_e)$. If the function f_e measures the time needed for performing a capacity expansion for element e and if we assume that the capacity expansions on different elements can be done in parallel, a natural measure for the overall expansion cost will be the maximum of the individual expansion costs $f_e(c_e)$. By this type of budget constraint we bound the longest time needed for a capacity expansion.

Outline of the paper. In Section 2, we first propose a model for the bottleneck capacity expansion problem in the framework of algebraic optimization. This model offers great flexibility for choosing the cost functions f_e and for modelling the total expansion cost. We then derive a parametric reformulation of the bottleneck capacity expansion problem. In Sections 3, 4 and 5, we propose efficient solution approaches for a large class of bottleneck capacity expansion problems. In Section 3, we propose a parametric search algorithm which is based on techniques of Megiddo [11, 12]. We show that this algorithm solves the bottleneck capacity expansion problem within a strongly polynomial number of steps under rather mild assumptions on the structure of the budget constraint provided a strongly polynomial time algorithm is available for the underlying algebraic combinatorial optimization problem. In Section 4, we deal with the special case of the bottleneck capacity expansion problem which results if the overall expansion cost is obtained as sum of monotone increasing, continuous piecewise affine-linear functions f_e, $e \in E$. We show that the Newton-type approach of Radzik [14] yields a strongly polynomial time algorithm for this class of bottleneck capacity expansion problems provided that the underlying combinatorial optimization problem can be solved in strongly polynomial time. For many combinatorial optimization problems, this second approach turns out to be faster than the general Megiddo-type approach. In Section 5, we deal with two special cases of the bottleneck capacity expansion problem which can be solved within a strongly polynomial number of steps by a binary search approach. In Section 6 we close the paper with some concluding remarks.

2. Bottleneck capacity expansion problems

2.1. A general model for bottleneck capacity expansion problems

The *bottleneck capacity expansion problem* BCEP considered in this paper consists of three ingredients, namely (i) the underlying combinatorial structure, (ii) the cost arising by expanding the capacity of a single element and (iii) the form of the budget constraint.

To model the underlying combinatorial structure, we assume that we are given a finite ground set $E = \{e_1, e_2, \dots, e_n\}$ and a family $\mathcal{F}$ of subsets of E, called *feasible solutions.* Every element $e \in E$ is associated with a nonnegative real capacity $\widehat{c}_e$. The *capacity* $\widehat{c}(F)$ of a feasible solution $F \in \mathcal{F}$ with respect to the capacities $\widehat{c}_e$ is defined to be the smallest capacity of an element in F, *i.e.*,

$$\widehat{c}(F) = \min_{e \in F} \widehat{c}_e. \tag{1}$$

The capacity of the combinatorial structure $(E, \mathcal{F})$ is given by the capacity of a feasible set with maximum capacity, *i.e.*, by

$$\max_{F \in \mathcal{F}} \widehat{c}(F) = \max_{F \in \mathcal{F}} \min_{e \in F} \widehat{c}_e. \tag{2}$$

As outlined in the introduction, there are many practical situations where it is possible to expand (increase) the capacities $\widehat{c}_e$ at a given cost. This paper deals with the problem of increasing the capacities $\widehat{c}_e$ in order to increase the capacity of the structure $(E, \mathcal{F})$ as much as possible while keeping the overall costs resulting from capacity expansions within a given budget. The cost incurred by increasing the capacity $\widehat{c}_e$ of an element $e \in E$ may for example measure the money or the time needed for the expansion, or a combination of both. In order to arrive at a general and flexible model, we are going to model the cost functions and the associated budget constraint in an algebraic way. To this end, let $(H, \oplus, \preceq)$ be a totally ordered commutative semigroup such that the composition $\oplus$ is compatible with the linear order $\preceq$, *i.e.*,

$$a_1 \preceq a_2 \quad \Longrightarrow \quad a_1 \oplus b \preceq a_2 \oplus b \qquad \text{for all } a_1, a_2, b \in H.$$

We use the notation $a \prec b$ to denote the situation that $a \preceq b$, but $a \neq b$. We assume that H contains a neutral element denoted by 0, *i.e.*, we have $a \oplus 0 = a$ for all $a \in H$. For examples of algebraic structures $(H, \oplus, \preceq)$ which are relevant in practical bottleneck capacity expansion problems see Section 2.2. In analogy to the summation sign $\sum$ we introduce the notation $\bigoplus_{k=1,\dots,r} a_k$ to denote the composition $a_1 \oplus a_2 \oplus \ldots \oplus a_{r-1} \oplus a_r$ for $a_k \in H$, $k = 1, \dots, r$.

To model the costs incurred by capacity expansions, we associate each element $e \in E$ with a monotone increasing function $f_e : \mathbb{R}_0^+ \to H$. For $t > \widehat{c}_e$, the value $f_e(t)$ represents the cost incurred by increasing the capacity of element e from $\widehat{c}_e$ to t. For the sake of convenience, we assume that the functions f_e, $e \in E$, are normalized, *i.e.*,

$$f_e(t) = 0 \qquad \text{for} \qquad t \leq \widehat{c}_e, \tag{3}$$

where the 0 on the right hand side of (3) denotes the neutral element in $(H, \oplus, \preceq)$. For examples of cost functions f_e which we have in mind and which typically arise in applications, we refer to Section 2.2. Note that we do not require the functions f_e to be continuous. In fact, in most cases the functions f_e will have jumps. In some of the following sections we will require that the functions f_e fulfill further properties, like continuity from the left or right or piecewise linearity. These properties will be stated explicitly in the respective sections. We are now ready to formulate the budget constraint. Let $B \in H$ with $0 \preceq B$ be the budget which is available for increasing the capacities $\widehat{c}_e$. The general budget constraint for $F \in \mathcal{F}$ has the following form:

$$\bigoplus_{e \in F} f_e(c_e) \preceq B \tag{4}$$

where $c_e \geq \widehat{c}_e$ denotes the new capacity of element $e \in E$.

Summarizing, we can formulate the *bottleneck capacity expansion problem*, BCEP for short, as follows:

$$\begin{aligned} &\max_{F \in \mathcal{F}} \min_{e \in F} c_e \\ \text{s.t.} \quad &\bigoplus_{e \in F} f_e(c_e) \preceq B. \end{aligned}$$

Note that due to the assumption (3), the constraints $c_e \geq \widehat{c}_e$ for $e \in E$ can be omitted.

2.2. Some examples of bottleneck capacity expansion problems relevant in practice

Let us start with some examples of algebraic structures $(H, \oplus, \preceq)$ which lead to bottleneck capacity expansion problems which are of practical relevance.

(H1) Let $H = \mathbb{R}$, choose the addition in $\mathbb{R}$ as composition $\oplus$ and the usual order $\leq$ as order relation $\preceq$. The neutral element is the real number 0. For this model, the budget constraint (4) turns into the constraint

$$\sum_{e \in F} f_e(c_e) \leq B$$

which requires that the sum of all expansion costs stays within the budget $B \in \mathbb{R}$.

(H2) Let H be the set of reals extended by $-\infty$, *i.e.*, $H = \overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty\}$ and let $\preceq$ be the natural order $\leq$. The composition $\oplus$ is defined by $a \oplus b := \max\{a, b\}$. The neutral element is $-\infty$. For this case, the budget constraint (4) turns into the constraint

$$\max_{e \in F} f_e(c_e) \leq B.$$

This model can be applied when we want to minimize the maximum individual expansion cost. Such a situation arises, for example, if $f_e(c_e)$ represents the time needed to increase the capacity of element e to the value c_e, and the capacity expansion has to be finished by time B.

(H3) Let $H = \overline{\mathbb{R}} \times \mathbb{R}$ and let $\preceq$ be the lexicographical order on H. The composition $\oplus$ is defined by

$$(a_1, b_1) \oplus (a_2, b_2) := \begin{cases} (a_1, b_1) & \text{if} \quad a_1 > a_2 \\ (a_1, b_1 + b_2) & \text{if} \quad a_1 = a_2. \end{cases}$$

The neutral element is $(-\infty, 0)$.

A possible application for this model is the following situation: let $f_e(t) = (g_e(t), h_e(t))$ and assume that $g_e(t)$ represents the time to increase the capacity of $e \in E$ to a value $t > \widehat{c}_e$ and that $h_e(t)$ represents the cost to increase the capacity of $e \in E$ to a value $t > \widehat{c}_e$. Let $B = (B_1, B_2)$. The budget constraint (4) asks for a capacity expansion which can be performed within the prescribed time bound B_1 and whose sum of expansion costs stays within the cost bound B_2 in case that the expansion needs exactly B_1 units of time.

(H4) In a similar spirit as in the example above, we can define cost functions with $k \geq 2$ components and use the lexicographic order as order $\preceq$. The compositions which are within each component have to be chosen such that the resulting composition $\oplus$ is compatible with the order $\preceq$.

Let us now mention four classes of cost functions that are important from the practical point of view. Recall that in the general model we only require that the cost functions f_e are monotone increasing and fulfill the normalization property (3). The functions within the classes (F1–F4) are all real-valued functions.

(F1) The first class consists of the class of real-valued functions $f_e : \mathbb{R}_0^+ \to \mathbb{R}$ which are affine-linear for $t \geq \widehat{c}_e$, *i.e.*, there is some nonnegative number α_e such that

$$f_e(t) := \begin{cases} 0 & \text{if } t \leq \widehat{c}_e \\ \alpha_e(t - \widehat{c}_e) & \text{if } t > \widehat{c}_e. \end{cases}$$

In this model, there is a cost of α_e per unit of increase of the capacity $\widehat{c}_e$. Note that the functions f_e in this class are continuous.

(F2) The second class of functions consists of real-valued functions with one step. They are of the form

$$f_e(t) := \begin{cases} 0 & \text{if } t \leq \tau_e \\ \beta_e & \text{if } t > \tau_e \end{cases}$$

where $\tau_e \geq \widehat{c}_e$ and β_e are given nonnegative real numbers. The functions in this class have a jump at the point τ_e and are thus not continuous. In this model we pay a fixed cost of β_e units to increase the capacity of element e beyond the limit τ_e.

(F3) The third class of functions are fixed-charge functions of the following type:

$$f_e(t) := \begin{cases} 0 & \text{if } t \leq \tau_e \\ \beta_e + \alpha_e(t - \tau_e) & \text{if } t > \tau_e \end{cases}$$

where α_e, β_e and $\tau_e \geq \widehat{c}_e$ are given nonnegative real numbers. This third class comprises the first two classes.

(F4) The fourth and so far most general class consists of functions which are monotone increasing and piecewise affine-linear. Let $\widehat{c}_e \leq \tau_e^{(1)} < \tau_e^{(2)} < \ldots < \tau^{(r_e)}$ be the points where the cost function f_e changes its slope (is not differentiable) and/or has a jump. These points are referred to as *breakpoints* of f_e. Between two breakpoints f_e is assumed to be affine-linear. We thus obtain the following shape

$$f_e(t) := \begin{cases} 0 & \text{if} \quad t \leq \tau_e^{(1)} \\ \beta_e^{(1)} + \alpha_e^{(1)} t & \text{if} \quad \tau_e^{(1)} < t \leq \tau_e^{(2)} \\ \beta_e^{(2)} + \alpha_e^{(2)} t & \text{if} \quad \tau_e^{(2)} < t \leq \tau_e^{(3)} \\ \vdots & \qquad \vdots \\ \beta_e^{(r_e)} + \alpha_e^{(r_e)} t & \text{if} \quad t > \tau_e^{(r_e)} \end{cases} \tag{5}$$

where $\alpha_e^{(q)} \geq 0$ and $\beta_e^{(q)}$, $q = 1, \dots, r_e$, are given real numbers. (Note: these constants have to be chosen such that f_e is monotone increasing.)
If all $\alpha_e^{(q)}$, $q = 1, \dots, r_e$, are zero, f_e is called a *step function* with r_e steps. In this case the monotonicity implies $0 < \beta_e^{(1)} < \ldots < \beta_e^{(r_e)}$. Step functions have the following shape:

$$f_e(t) := \begin{cases} 0 & \text{if} \quad t \leq \tau_e^{(1)} \\ \beta_e^{(1)} & \text{if} \quad \tau_e^{(1)} < t \leq \tau_e^{(2)} \\ \beta_e^{(2)} & \text{if} \quad \tau_e^{(2)} < t \leq \tau_e^{(3)} \\ \vdots & \qquad \vdots \\ \beta_e^{(r_e)} & \text{if} \quad t > \tau_e^{(r_e)}. \end{cases} \tag{6}$$

2.3. Previous and related results

Zhang *et al.* [15] have investigated the following special bottleneck capacity expansion problem:

$$\begin{aligned} & \max_{F \in \mathcal{F}} \min_{e \in F} c_e \\ \text{s.t.} \quad & \textstyle\sum_{e \in F} \alpha_e (c_e - \widehat{c}_e) \leq B \\ & c_e \geq \widehat{c}_e \qquad\qquad \text{for all } e \in E. \end{aligned}$$

This version of the bottleneck capacity expansion problem BCEP arises by choosing (H1) as algebraic system and using cost functions f_e of type (F1). Zhang *et al.* proposed a polynomial time algorithm for this special case of the BCEP and a strongly polynomial time algorithm for the special case where the underlying combinatorial optimization problem given by the structure $(E, \mathcal{F})$ is the minimum spanning tree problem.

Several authors have investigated expansion problems where the capacity (weight) of a structure $(E, \mathcal{F})$ is defined as

$$\min_{F \in \mathcal{F}} \sum_{e \in F} \widehat{c}_e \tag{7}$$

and where the task is to find new capacities $c_e \geq \widehat{c}_e$ so as to increase the capacity of $(E, \mathcal{F})$ as much as possible while keeping the expansion costs $\sum_{e \in E} f_e(c_e)$ within a given budget B. Most of the results are for affine-linear costs f_e, but some other cost models have been investigated as well. The known results include results for the maximum flow/minimum cut problem by Ahuja and Orlin [1], for the shortest path problem by Fulkerson and Harding [8], for the minimum spanning tree problem and certain matroid optimization problems by Frederickson and Solis-Oba [6, 7]), and for submodular flow problems by Jüttner [9].

Another class of related problems arises if the capacities (weights) can be decreased in order to decrease the capacity (weight) of the structure $(E, \mathcal{F})$. Typically, these problems involve lower bounds on the new capacities (weights) which turns most of them into NP-hard problems. See *e.g.* Philipps [13] for the maximum flow/minimum cut case and Drangmeister *et al.* [5] for the minimum spanning tree case. The paper by Krumke *et al.* [10] is worth to be mentioned here since it considers multi-criteria budget constraints (such constraints can be formulated within the framework of our algebraic model).

2.4. Parametric reformulation of the bottleneck capacity expansion problem

Let F^* be an optimal solution of the bottleneck capacity expansion problem BCEP and let t^* denote the corresponding objective function value, *i.e.*, $\min_{e \in F^*} c_e = t^*$. Since the functions f_e are monotone increasing, it makes no sense to increase the capacity of an element $e \in E$ if this increase is not necessary to achieve the objective function value t^*. Hence we can assume that the increased capacities c_e are chosen such that

$$c_e = \begin{cases} t^* & \text{if } \widehat{c}_e < t^* \text{ and } e \in F^* \\ \widehat{c}_e & \text{otherwise.} \end{cases}$$

This means that we can assume that the capacities c_e of elements whose capacity is increased, *i.e.*, those for which $c_e > \widehat{c}_e$ holds, all have the same value. Making use of the normalization property (3), the BCEP can be reformulated and attacked by a parametric approach.

Let $t \in \mathbb{R}$. Consider the algebraic optimization problem

$$\min_{F \in \mathcal{F}} \bigoplus_{e \in F} f_e(t) \tag{8}$$

and define the function z by

$$z(t) := \min_{F \in \mathcal{F}} \bigoplus_{e \in F} f_e(t). \tag{9}$$

Since the functions f_e are monotone increasing, it follows that the function z is monotone increasing as well. Thus the BCEP is equivalent to the following problem:

$$\text{find the largest parameter } t^* \text{ such that } z(t^*) \preceq B. \tag{10}$$

t^* corresponds to the optimal objective function value in the BCEP. If the functions f_e are continuous, then z is continuous as well and the above problem can be simplified to

$$\text{find the largest parameter } t^* \text{ such that } z(t^*) = B. \tag{11}$$

Similar parametric search problems as in (10) and (11) arise in connection with combinatorial fractional programming problems (see Megiddo [11,12] and Radzik [14]) and with inverse parametric optimization problems (see *e.g.* Burkard *et al.* [2]).

Clearly a basic ingredient of an algorithm for determining t^* will be a subroutine for evaluating the function z for a fixed value of t, *i.e.*, for solving the problem (8) for a fixed value of t. Note that for a fixed parameter value t, the problem (8) is an algebraic optimization problem. In particular, if $\oplus$ is the normal addition on the set of real numbers, we get a classical combinatorial optimization problem with sum objective function. If $\oplus$ is the maximum operation, we get a combinatorial optimization problem with bottleneck objective function.

For solution routines for general algebraic optimization problems consult Burkard and Zimmermann [4] or Zimmermann [16]. For the purpose of this paper it suffices to mention that efficient algorithms are known for a large class of algebraic optimization problems including algebraic spanning tree problems, algebraic assignment problems and algebraic path problems. For some problems the known algorithms make, however, use of additional assumptions on the algebraic system $(H, \oplus, \preceq)$. For example, the algorithm of Burkard *et al.* [3] for algebraic assignments works for commutative semigroups which are so-called *d*-monoids. *d*-monoids fulfill the following so-called divisor-rule:

$$\text{if } a \preceq b \ \text{ then there exists an element } d \in H \text{ such that } a \oplus d = b.$$

In the next section we will show how a large class of bottleneck capacity expansion problems can be solved within a strongly polynomial number of steps by taking advantage of the parametric reformulation obtained above.

3. Megiddo-type approaches for solving bottleneck capacity expansion problems

In this section we will apply the approach of Megiddo [11,12] to solve the parametric reformulation (10) of the bottleneck capacity expansion problem. Recall that our task is to find the largest value of t^* such that $z(t^*) \preceq B$ holds where z is given by (9).

3.1. Technical preliminaries

In order to make sure that the function $\bigoplus_{e\in F} f_e$ does not behave too badly, we will make the following assumptions: let the cost functions $f_e : \mathbb{R}_0^+ \to H$ be members of a class $\mathcal{M}$ of monotone increasing functions which fulfills the following three properties:

(P1) $\mathcal{M}$ is closed under the operation $\oplus$, *i.e.*, for any two functions $u, v \in \mathcal{M}$ the composition $u \oplus v$ belongs again to $\mathcal{M}$.

(P2) Each function $f \in \mathcal{M}$ has only finitely many jumps (= points of discontinuity).

(P3) For any two functions $u, v \in \mathcal{M}$, $u \neq v$, there are only finitely many proper intersection points of u and v. By a proper intersection point we mean a point t fulfilling the following two properties:
- $u(t) = v(t)$;
- there exists a real number $\epsilon_0 > 0$ such that we either have $u(t-\epsilon) \neq v(t-\epsilon)$ for all $0 < \epsilon \leq \epsilon_0$ or $u(t+\epsilon) \neq v(t+\epsilon)$ for all $0 < \epsilon \leq \epsilon_0$.

(This definition is used to exclude inner points of intervals on which u and v coincide.)

Note that the class of monotone increasing, piecewise affine-linear functions with a finite number of breakpoints fulfills the above properties (P1–P3) if we take $\oplus = +$ or $\oplus = \max$. Another example is obtained by taking the class of all polynomials with positive coefficients and using the multiplication as operation $\oplus$.

3.2. A first Megiddo-type approach

We are now ready to describe the basic version of our Megiddo-type approach. Throughout the algorithm we will maintain an interval I such that the optimal parameter value t^* is contained in I. We may start with the interval $I := [0, \infty)$. This parameter interval can be further reduced by the following considerations: Let $n_{\max}$ and $n_{\min}$ denote the maximum and the minimum cardinality of a feasible solution, respectively. Moreover, let $b_{\min} \in H$ be defined by

$$b_{\min} := \max \left\{ b \in H \mid b^{n_{\max}} = \underbrace{b \oplus b \oplus \cdots \oplus b}_{n_{\max}\text{-times}} \preceq B \right\}.$$

(Note that by assumption $0 \preceq B$. Thus $b_{\min}$ is well-defined.) Similarly, let $b_{\max} \in H$ be defined by

$$b_{\max} := \max \left\{ b \in H \mid b^{n_{\min}} = \underbrace{b \oplus b \oplus \cdots \oplus b}_{n_{\min}\text{-times}} \preceq B \right\}.$$

(Note that the max operation that occurs in the computation of $b_{\min}$ and of $b_{\max}$ is with respect to the order $\preceq$.) Now let

$$t_{\min} := \sup \{ t \in \mathbb{R} \mid f_e(t) \preceq b_{\min} \text{ for all } e \in E \} \tag{12}$$

be the largest value of the parameter t such that all cost functions $f_e(t)$ have values $\preceq b_{\min}$. Consequently, $t_{\min}$ is a lower bound for the optimal solution value t^*. In particular, if $t_{\min} = \infty$ holds, the capacity of the structure $(E, \mathcal{F})$ can be made arbitrarily large within the given budget which implies that the bottleneck capacity expansion problem does not have a finite optimal solution. If $t_{\min}$ is finite, we determine an upper bound $t_{\max}$ for the optimal parameter value t^* by

$$t_{\max} := \inf \{ t \mid b_{\max} \prec f_e(t) \text{ for all } e \in E \} \cdot \tag{13}$$

Obviously, $t_{\max}$ is an upper bound for t^*. Thus we can replace the original interval I by $I := [t_{\min}, t_{\max}]$. From the computational point of view, it will, however, often be preferable to work with the starting interval $[0, \infty)$ since the computation of $t_{\min}$ and of $t_{\max}$ might be time consuming for more complicated algebraic systems $(H, \oplus, \preceq)$.

The basic ingredient for our parametric algorithm to determine t^* will be an algorithm A for evaluating the function z for a fixed value of t, *i.e.*, for solving the algebraic optimization problem (8) for a fixed value of t. The main idea of a Megiddo-type approach is to apply such an algorithm parametrically, *i.e.*, with input data that are functions of the parameter t instead of constants. To make this approach work, we assume that the set of operations of algorithm A is limited to applying the algebraic operation $\oplus$ and to performing comparisons with respect to the order $\preceq$. (Actually, we could go along with a weaker assumption. Namely, we could allow operations which can be performed without knowing t^* and which do not produce functions $\notin \mathcal{M}$. For real valued functions and $\oplus = +$ we can, for example, allow multiplications of a scalar with a function.)

We extend algorithm A which works for single cost elements drawn from H to the case of cost functions $f_e : \mathbb{R}_0^+ \to H$. As long as A does not arrive at a comparison with respect to $\preceq$, we can proceed with the functions depending on the parameter t in the same way as we would do for constant costs. As soon as a comparison occurs we proceed as follows: Suppose we need to compare the two functions u and v. First, we compute the proper intersection points of u and v which lie within the current interval I. Let the candidate set $\mathcal{S}$ contain all these intersection points as well as all points of jump discontinuity of u and v which lie within the interval I. Let $s_1 < s_2 < \ldots < s_r$ be the sorted sequence of the points in $\mathcal{S}$. If this sequence is empty, u and v are comparable in I, *i.e.*, we will either have $u(t) \preceq v(t)$ for all $t \in I$ or $v(t) \preceq u(t)$ for all $t \in I$. In this case we can immediately decide the outcome of the comparison and proceed with running the algorithm A. If $\mathcal{S}$ is, however, nonempty, we determine the median s_M of the set $\mathcal{S}$ and solve the subproblem (8) for $t := s_M$ to evaluate $z(s_M)$. If we have $z(s_M) \preceq B$, then it is clear that $t^* \geq s_M$ holds. Therefore, we can reduce the current interval I by replacing its left endpoint by s_M. Moreover, we remove all points from $\mathcal{S}$ which are smaller than s_M. Similarly, if $B \prec z(s_M)$ holds, we replace the right endpoint of I by s_M and we remove from $\mathcal{S}$ all points which are larger than s_M. We then proceed with the new smaller set $\mathcal{S}$ in the same way until we get an interval I for which the two functions u and v are comparable. At that point we can decide the outcome of the comparison and continue with the algorithm A. At the end of the algorithm, we are left with an interval I which contains the optimal value t^* and with a feasible solution $F^* \in \mathcal{F}$ which is an optimal solution of the problem (8) for all $t \in I$. By determining the largest t such that

$$\bigoplus_{e \in F^*} f_e(t) \preceq B$$

the optimal parameter t^* can be found (in case we get $t^* = \infty$, the BCEP is unbounded and does not have a solution).

Let us now analyse the running time of this algorithm. Suppose that algorithm A performs $O(T_A)$ operations $\oplus$ and comparisons with respect to the order $\preceq$ when it is applied to an input with constant costs. Usually the running time of an algorithm is measured by the number of elementary arithmetic operations it performs. In our case we cannot count this number without making assumptions on $\preceq$ and $\oplus$. To overcome this difficulty, we count the number of steps the algorithm performs, where by step we either mean a comparison between two elements a, $b \in H$, or an operation $a \oplus b$ for $a, b \in H$, or an elementary arithmetic operation.

For the analysis of the parametric algorithm it is essential how many steps are needed for carrying out an operation $u \oplus v$ or a comparison between u and v for functions $u, v \in \mathcal{M}$. Suppose that it takes $O(T_\oplus)$ steps to perform an operation of type $\oplus$. To bound the number of steps needed per comparison, suppose that $O(T_S)$ steps are needed to compute the candidate set $\mathcal{S}$. To compute $\mathcal{S}$ we need to compute all proper intersection points of two functions u and v from the class $\mathcal{M}$ and add all points where u or v have a jump. Since the cardinality of $\mathcal{S}$ is roughly halved in each iteration, the comparison between u and v can be decided after $O(\log |\mathcal{S}|)$ iterations. In each iteration we need to evaluate the function z for a fixed value of t which can be done in $O(T_A)$ steps by applying algorithm A. Since a single median computation requires time linear in the number of elements of the set, all median computations can be accomplished in $O(|\mathcal{S}|)$ time. (In the first iteration we need $O(|\mathcal{S}|)$ steps, in the second $O\left(\frac{|\mathcal{S}|}{2}\right)$, in the third $O\left(\frac{|\mathcal{S}|}{4}\right)$ etc.) All in all, we get that a comparison of two functions in $\mathcal{M}$ can be performed using $O(T_A \log |\mathcal{S}| + T_S + |\mathcal{S}|)$ steps.

Let K_1 be an upper bound on the number of proper intersection points of any two functions from $\mathcal{M}$ and let K_2 be an upper bound on the number of jumps of a function in $\mathcal{M}$. Then $|\mathcal{S}|$ can be bounded from above by $K = K_1 + 2K_2$. Consequently, our algorithm performs $O(T_A(T_A \log(K+1) + T_S + K + T_\oplus))$ steps. Since the function class $\mathcal{M}$ satisfies the properties (P1–P3) by assumption, our algorithm will terminate after a finite number of steps if T_S and $T_\oplus$ are finite. The number of steps will be polynomial (strongly polynomial) if T_S, K, $T_\oplus$ and T_A are polynomial (strongly polynomial). More specific results can be obtained for specific classes of cost functions $\mathcal{M}$. Summarizing, we have obtained the following result.

Theorem 3.1. *Consider the class of bottleneck capacity expansion problems with cost functions f_e from a class $\mathcal{M}$ which fulfills properties (P1–P3). Let K be an upper bound on the number of all proper intersection points and jumps of two functions from $\mathcal{M}$. Suppose it takes $O(T_S)$ steps to determine all proper intersection points and jumps of two functions from $\mathcal{M}$, and $O(T_\oplus)$ steps to apply $\oplus$ to* *two functions in $\mathcal{M}$. Moreover, assume that $O(T_A)$ steps are required to solve the algebraic optimization problem (8) for a fixed value of t. Then the corresponding bottleneck capacity expansion problem can be solved within $O(T_A(T_A \log(K+1) + T_S + K + T_\oplus))$ steps.*

The case where all cost functions f_e are monotone increasing, piecewise affine-linear functions with a finite number of breakpoints is of particular importance in practical applications and deserves special attention. This class of functions clearly satisfies properties (P2) and (P3). To make sure that property (P1) is satisfied as well, we need to restrict our attention to the cases where the composition of two piecewise affine-linear functions is piecewise affine-linear again. (For real valued functions this is, for example, true for $\oplus = +$ and $\oplus = \max$.)

Let $\mathcal{P}$ denote the class of monotone increasing, piecewise-affine linear functions from H to $\mathbb{R}_0^+$ with a finite number of breakpoints, and let $\mathcal{P}^{(L)}$ denote the subclass of $\mathcal{P}$ with at most L breakpoints.

Theorem 3.2. *Consider the class of bottleneck capacity expansion problems with cost functions $f_e \in \mathcal{P}$ where we assume that $\mathcal{P}$ is closed with respect to $\oplus$. Furthermore, assume that all functions which are involved in comparisons throughout the parametric algorithm belong to the class $\mathcal{P}^{(L)}$. This class of bottleneck capacity expansion problems can be solved within $O({T_A}^2 \log L + T_A L)$ steps.*

Proof. This theorem follows immediately from Theorem 3.1. It can easily be checked that we get $K = O(L)$, $T_S = O(L)$ and $T_\oplus = O(L)$. □

Observe that in many cases L can be bounded in a nice way. Suppose that all f_e belong to the class $\mathcal{P}^{(L_1)}$, *i.e.*, they have at most L_1 breakpoints. Then in many cases we will have $L = O(nL_1)$ (this holds in particular in cases where the functions which are compared to each other by the parametric algorithm represent cost values of feasible solutions or of partial solutions). In other cases, we might have $L = O(T_A L_1)$. (Note that we perform at most $O(T_A)$ operations of type $\oplus$ throughout one run of algorithm A. So under mild assumptions on the nature of algorithm A, we can conclude that we have to deal only with functions with at most $O(T_A L_1)$ breakpoints.)

Consequently, most cases of the BCEP restricted to piecewise-affine linear cost functions f_e can be solved within a strongly polynomial number of steps if the underlying algebraic optimization problem (8) can be solved within a strongly polynomial number of steps.

3.3. Modifications and improvements of the basic Megiddo approach

In this section we will deal with modifications which in many cases lead to a speed-up of the basic Megiddo approach proposed in the preceding section.

One disadvantage of the basic approach became evident in the discussion at the end of the previous section. Even if the given cost functions f_e are of simple structure (*e.g.* piecewise affine-linear with only one breakpoint), the composed functions which are built up in the course of the parametric algorithm can become much more complicated. This observation suggests the following approach: we start by putting all the breakpoints (= points of discontinuity or nondifferentiability) of the given n cost functions f_e, $e \in E$, into a joint sequence. Let the resulting sequence (in sorted order) be

$$t_1 < t_2 < \ldots < t_r.$$

For notational convenience, we set $t_0 := 0$ and $t_{r+1} = \infty$. By applying binary search we can find the interval $J := [t_j, t_{j+1})$ with $j \in \{0, \ldots, r\}$ for which we have $z(t_j) \preceq B$ and $B \prec z(t_{j+1})$. Clearly this interval J contains the optimal solution t^*. We then use J as start interval I in the parametric search. The advantage of this approach is that, by construction, the cost functions f_e have no breakpoints in the interior of J. Observe that we even can neglect possibly existing breakpoints at the left end of J because the functions f_e are monotone increasing by assumption. Hence, we either find a solution $t^* \in (t_j, t_{j+1})$ or we have $t^* = t_j$. The advantage of this approach is that the functions with which we have to work throughout the parametric algorithm are often of much simpler structure than in the case of the basic approach described in Section 3.2.

Let us now analyse the running time of this preprocessing step. Let L_2 be the total number of breakpoints of the cost functions f_e. Then we need $O(T_A \log L_2 + L_2)$ steps to determine the interval J. To see this, note that the binary search takes $O(\log L_2)$ iterations, and in each iteration we need to evaluate the function z which takes $O(T_A)$ steps. The term $O(L_2)$ accounts for the time which is needed for computing the test points needed in the binary search. This can either be achieved by median computations as explained in the previous section, or by sorting the sequence of the breakpoints of the cost functions f_e (where for the latter alternative we need to assume that the breakpoints of the cost functions f_e are given in sorted order in the input which will normally be the case).

In many cases the number of steps performed by the modified Megiddo approach with preprocessing will be smaller than the number of steps performed by the basic approach proposed in Section 3.2. To illustrate this point, let us consider the special case of real-valued piecewise affine-linear cost functions and sum-budget constraints, *i.e.*, we have $\oplus = +$. After the preprocessing we are left with cost functions f_e which are affine-linear over the interval of interest. Note that the sum of two affine-linear functions is affine-linear again. It is easy to check that for this special case we get $K = O(1)$, $T_\oplus = O(1)$ and $T_S = O(1)$ in Theorem 3.1. Thus, it follows that the parametric algorithm performs at most $O({T_A}^2)$ steps. Including the number of steps performed during the preprocessing we are done within $O({T_A}^2 + T_A \log L_2 + L_2)$ steps. This bound is not directly comparable to the bound in Theorem 3.2, but it is easy to see that in most cases the modified algorithm performs a smaller number of steps than the basic algorithm. Typically, one can expect to win at least a log-factor. Consider, for example, the special case where the cost functions f_e have at most L_1 breakpoints and where we have $L = O(nL_1)$ in Theorem 3.2 (which is a rather favourable assumption for that case). Clearly we then have $L_2 \leq nL_1$ and hence the modified approach finishes within $O({T_A}^2 + T_A \log(nL_1) + nL_1)$ steps while the basic approach requires $O({T_A}^2 \log(nL_1) + nT_AL_1)$ steps.

Another possibility to reduce the number of steps performed by the basic Megiddo approach is possible in cases where there exists a parallel algorithm for solving the algebraic optimization problem (8). Let us assume that the parallel algorithm uses P processors. Megiddo [12] observed that the number of iterations in the parametric search approach can be reduced by exploiting the parallelization. The idea is to simulate the parallel computation on a single serial processor. We let each of the processors, one after the other, perform its work until it arrives at the first comparison between two functions. Next we compute the candidate set for each of these P comparisons and then combine these candidate sets into a common candidate set. Then we proceed as described in the previous section. After the P comparisons have been decided, we continue in the same way until each processor arrives at its next comparison or at the end of its task. This approach helps in reducing the number of subproblems of type (8) which have to be solved throughout the course of the algorithm.

4. A Newton approach for bottleneck capacity expansion problems with a sum budget constraint and continuous piecewise affine-linear cost functions

In this section we consider the class of bottleneck capacity expansion problems with a sum budget constraint which results from the algebraic system $(H, \oplus, \preceq) = (\mathbb{R}, +, \leq)$. Let $B > 0$ be the given budget. Moreover, we assume that the cost functions f_e are monotone increasing, continuous piecewise affine-linear functions. In other words, the functions f_e are of the shape (5) where we additionally assume that the functions f_e are continuous and that the slopes $\alpha_e^{(q)}$ are positive.

We first perform the same preprocessing as in the modified Megiddo approach discussed in Section 3.3. Let again

$$t_1 < t_2 < \ldots < t_r$$

denote the sorted sequence of breakpoints of the cost functions f_e. Let $t_{r+1} := \infty$. By applying binary search we can then find out which interval $[t_j, t_{j+1})$, $j \in \{1, \ldots, r\}$, contains the optimal solution t^*. Let us call this interval $J = [t^{(l)}, t^{(u)})$.

Since the cost functions f_e are assumed to be continuous, the function z from (8) is continuous as well. Hence we can use the parametric reformulation (11), *i.e.*, we want to find the largest value of t^* such that $z(t^*) = B$. The construction of the interval J implies that the cost functions $f_e(t)$ are affine-linear functions in I.

More specifically, for $t \in J$ we obtain

$$f_e(t) := \begin{cases} 0 & \text{if } t^{(l)} < \tau_e^{(1)} \\ \beta_e^{(q)} + \alpha_e^{(q)} t & \text{if } \tau_e^{(q)} \le t^{(l)} < \tau_e^{(q+1)} \text{ for a } q \in \{1, \dots, r_e - 1\} \\ \beta_e^{(r_e)} + \alpha_e^{(r_e)} t & \text{if } t^{(l)} \ge \tau_e^{(r_e)}. \end{cases} \tag{14}$$

For notational convenience we define α_e and β_e such that $f_e(t) = \alpha_e t + \beta_e$ for $t \in J$. (This can be done since f_e is affine-linear in J.) For a feasible set $F \in \mathcal{F}$, let $\alpha(F) := \sum_{f \in F} \alpha_f$ and let $\beta(F) := \sum_{f \in F} \beta_f$. Using this notation the computation of $z(t)$ for $t \in J$ can be rewritten as follows:

$$z(t) := \min_{F \in \mathcal{F}} \{ \alpha(F)t + \beta(F) \} \cdot \tag{15}$$

It is easy to see that z is a monotone increasing, concave, piecewise affine-linear function. Note that evaluating z for a fixed value of t corresponds to solving the combinatorial optimization problem

$$\min_{F \in \mathcal{F}} \left\{ \sum_{e \in F} \gamma_e \right\} \tag{16}$$

where $\gamma_e = \alpha_e t + \beta_e$, $e \in F$, are constants for fixed t.

Our goal is to find the largest parameter value t such that $z(t) = B$. This is equivalent to finding the smallest value of t such that $z(t) \ge B$, *i.e.*, the smallest value of t such that

$$\min_{F \in \mathcal{F}} \{ \alpha(F)t + \beta(F) \} \ge B. \tag{17}$$

Inequality (17) can be rephrased as

$$\alpha(F)t + \beta(F) \ge B \qquad \text{for all } F \in \mathcal{F}. \tag{18}$$

Note that all slopes in (14) are nonnegative. Thus, it follows immediately that we have $\alpha(F) \ge 0$ for all $F \in \mathcal{F}$. Since the slopes $\alpha_e^{(q)}$, $q = 1, \dots, r_e$, are positive by assumption, it follows from (14) that $\alpha(F') = 0$ implies $\beta(F') = 0$. As we are looking for values of t such that (18) is fulfilled, it thus cannot happen that there exists a feasible set $F \in \mathcal{F}$ such that $\alpha(F) = 0$. Therefore, condition (18) can be rephrased as

$$t \ge \frac{B - \beta(F)}{\alpha(F)} \qquad \text{for all } F \in \mathcal{F}. \tag{19}$$

We wish to find the smallest value of t such that (19) holds. It is easy to see that this task is equivalent to the following problem

$$\max_{F\in\mathcal{F}} \frac{B-\beta(F)}{\alpha(F)}\,. \tag{20}$$

The problem (20) is a so-called *linear fractional combinatorial optimization problem* (*cf.* [11,14]).

The algorithm proposed below is a Newton-type algorithm which has turned out to be effective for fractional combinatorial optimization problems (see Radzik [14]).

1. Start with $\tau_1 = t^{(l)}$ and set $i = 1$.
2. Determine $z_i := \min_{F\in\mathcal{F}} \sum_{e\in F} f_e(\tau_i)$.
   Let F_i be the corresponding optimal solution.
3. If $z_i = B$, terminate. $t^* = \tau_i$ is the optimal parameter.
   Otherwise continue with the next step.
4. Let $a_i := \alpha(F_i)$ and let $b_i := \beta(F_i)$.
5. Set $\tau_{i+1} := \dfrac{B-b_i}{a_i}$, $i := i+1$ and go to Step 2.

Note that it cannot happen that we arrive at $a_i = 0$ in Step 4 of the above algorithm. ($a_i = 0$ would imply $b_i = 0$ and since z is a piecewise affine-linear, concave function, we would have $z(t) = 0$ for $t \geq \tau_i$, but this contradicts the fact that the interval I contains the optimal parameter value t^* which satisfies $z(t^*) = B$.)

Radzik [14] obtained a strongly polynomial bound on the number of steps performed by this Newton approach. His analysis applies to our case as well. In this manner it can be shown that the algorithm above terminates after $O(n^2 \log^2 n)$ iterations. In each iteration we have to evaluate the function $z(t)$ for a fixed value of t. Summarizing we obtain the following theorem.

Theorem 4.1. *Consider a bottleneck capacity expansion problem over the algebraic system* $(\mathbb{R}, +, \leq)$ *with continuous, piecewise-affine linear cost functions* f_e, $e \in E$. *Let* L_1 *be an upper bound on the number of breakpoints of each individual cost function* f_e. *Moreover, let* T *denote the time needed to solve the combinatorial optimization problem (16).*

This case of the bottleneck capacity expansion problem is solved by the above Newton-type algorithm in $O(T\log(nL_1) + nL + Tn^2\log^2 n)$ *time.*

The $O(T\log(nL_1) + nL_1)$ part of the time bound corresponds to the complexity of the first phase of the algorithm in which the interval $[t^{(l)}, t^{(u)}]$ is determined.

5. Strongly polynomial time binary search algorithms for special cases of the bottleneck capacity expansion problem

In this section we will deal with two special cases of the bottleneck capacity expansion problem which can be solved within a strongly polynomial number of steps by applying a binary search approach. Section 5.1 deals with the case where the cost functions f_e are step functions with finitely many steps, and Section 5.2 deals with bottleneck capacity expansion problems where the budget constraint (4) is a bottleneck constraint.

5.1. Bottleneck capacity expansion problems with step functions as cost functions

In this section we will deal with the case of the bottleneck capacity expansion problem whose cost functions f_e, $e \in E$, are step functions which are continuous from the left and have finitely many steps, *i.e.*, which are of the form (6). (Step functions which are continuous from the right can be treated in an analogous way.)

Let $t_1 < t_2 < \ldots < t_{L_1}$ be the set of parameter values where at least one of the functions f_e, $e \in E$, has a jump. Let I be an interval which contains the optimal parameter value t^*. As discussed in Section 3, we can use $I := [0, \infty)$ or use the interval $I := [t_{\min}, t_{\max}]$ determined by (12) and (13), respectively. (It might, however, be more time consuming to compute $t_{\min}$ and $t_{\max}$ than to work with a larger interval I.) We can restrict our consideration to those values t_i which lie within the interval I. If I contains t_{L_1}, we first determine

$$z(t_r) := \min \bigoplus_{e \in F} f_e(t_r).$$

If $z(t_r) \preceq B$, the capacity of the instance can be made arbitrarily large within the given budget and we are finished. Otherwise, we perform a binary search on the set of t_i values within interval I. By the same argument as used in Section 3 it follows that the optimal value t^* can be found within $O(T_A \log L_1 + L_1)$ steps. (We need $\log L_1$ iterations in the binary search and each iteration requires the solution of an instance of the algebraic optimization problem (8)). If each of the step functions f_e has at most L_2 jumps, the number of steps performed by the algorithm will be within $O(T_A \log(L_2 n) + L_2 n)$.

5.2. Bottleneck capacity expansion problems with a bottleneck constraint

In this section we will deal with bottleneck capacity expansion problems for which the budget constraint (4) is a bottleneck (time) constraint, *i.e.*, is of the form

$$\max_{e \in F} f_e(t) \leq B. \tag{21}$$

In this case we can use a direct threshold algorithm for finding an optimal expansion which can be performed within the given budget (time) bound B. To avoid technical difficulties, let us assume that all cost functions f_e are continuous from the left. For every element $e \in E$ we determine the parameter value t_e as

$$t_e = \sup \{ t \mid f_e(t) \leq B \} \cdot \tag{22}$$

Then we sort these values, *i.e.*,

$$t_{e_1} \leq t_{e_2} \leq \ldots \leq t_{e_n}.$$

Now we consider a sequence of feasibility problems of the form

$$\text{does there exist a feasible solution } F \subseteq \{e_k, e_{k+1}, \ldots , e_n\}? \tag{23}$$

By applying binary search we can find the largest index k for which a feasible solution exists. Let k^* be this largest index and let F_{k^*} be a corresponding feasible solution. F_{k^*} yields an optimal solution of the bottleneck capacity expansion problem with the budget constraint (21). Its objective function value is $t_{e_{k^*}}$. Summarizing we get:

Theorem 5.1. *Consider the bottleneck capacity expansion problem with a bottleneck budget constraint. Let T_1 be the time needed to compute the parameter value t_e in (22) for a fixed $e \in E$ and let T_2 be the time needed to solve the feasibility problem (23) for a fixed value of k. Then the bottleneck capacity expansion problem can be solved in $O(n \log n + nT_1 + T_2 \log n)$ time.*

6. Conclusion

In this paper we presented a unified approach for bottleneck capacity expansion problems. First, we introduced a generic model for the BCEP by defining the overall expansion cost and the budget constraint in an algebraic way. Then we derived a parametric reformulation of the BCEP. Based on this parametric formulation we proposed generic solution approaches for the general bottleneck capacity expansion problem. For an important subclass of bottleneck capacity expansion problems we obtained algorithms which perform a strongly polynomial number of steps.

We hope that the approach taken in this paper will contribute to a better understanding which types of capacity expansion problems are efficiently solvable. As research question for future work we suggest to extend the work of this paper to other types of capacity expansion problems where the capacity of the structure $(E, \mathcal{F})$ is defined in a different manner.

References

[1] R.K. Ahuja and J.B. Orlin, A capacity scaling algorithm for the constrained maximum flow problem. *Networks* **25** (1995) 89-98.

[2] R.E. Burkard, K. Dlaska and B. Klinz, The quickest flow problem. *Z. Oper. Res. (ZOR)* **37** (1993) 31-58.

[3] R.E. Burkard, W. Hahn and U. Zimmermann, An algebraic approach to assignment problems. *Math. Programming* **12** (1977) 318-327.

[4] R.E. Burkard and U. Zimmermann, Combinatorial optimization in linearly ordered semimodules: A survey, in *Modern Applied Mathematics*, edited by B. Korte. North Holland, Amsterdam (1982) 392-436.

[5] K.U. Drangmeister, S.O. Krumke, M.V. Marathe, H. Noltemeier and S.S. Ravi, Modifying edges of a network to obtain short subgraphs. *Theoret. Comput. Sci.* **203** (1998) 91-121.

[6] G.N. Frederickson and R. Solis-Oba, Increasing the weight of minimum spanning trees. *J. Algorithms* **33** (1999) 244-266.

[7] G.N. Frederickson and R. Solis-Oba, Algorithms for robustness in matroid optimization, in *Proc. of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (1997) 659-668.

[8] D.R. Fulkerson and G.C. Harding, Maximizing the minimum source-sink path subject to a budget constraint. *Math. Programming* **13** (1977) 116-118.

[9] A. Jüttner, *On budgeted optimization problems.* Private Communication (2000).

[10] S.O. Krumke, M.V. Marathe, H. Noltemeier, R. Ravi and S.S. Ravi, Approximation algorithms for certain network improvement problems. *J. Combin. Optim.* **2** (1998) 257-288.

[11] N. Megiddo, Combinatorial optimization with rational objective functions. *Math. Oper. Res.* **4** (1979) 414-424.

[12] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms. *J. ACM* **30** (1983) 852-865.

[13] C. Phillips, The network inhibition problem, in *Proc. of the 25th Annual Symposium on the Theory of Computing* (1993) 776-785.

[14] T. Radzik, Parametric flows, weighted means of cuts, and fractional combinatorial optimization, in *Complexity in Numerical Optimization*, edited by P.M. Pardalos. World Scientific Publ. (1993) 351-386.

[15] J. Zhang, C. Yang and Y. Lin, A class of bottleneck expansion problems. *Comput. Oper. Res.* **28** (2001) 505-519.

[16] U. Zimmermann, *Linear and Combinatorial Optimization in Ordered Algebraic Structures.* North-Holland, Amsterdam, *Ann. Discrete Math.* **10** (1981).