KYBERNETIKA — VOLUME 47 (2011), NUMBER 5, PAGES 722–731

OPTIMIZATION OF AN SMD PLACEMENT MACHINE AND FLOWS IN PARAMETRIC NETWORKS

Katarína Cechlárová and Tamás Fleiner

In the minimization of the number of subtours made by the insertion head of an SMD placement machine a variant of the network flow problem arose. In a network with n vertices and m arcs a set F of arcs (parametrized arcs) is given. The task is to find a flow of a given size such that the maximum of flow values along the arcs from F is minimized. This problem can be solved by a sequence of maximum flow computations in modified networks where the capacities of the parametrized arcs are successively set to an increasing sequence of target parameter values. We show that it suffices to consider at most $O(|F|)$ different target values and so this approach leads to a strongly polynomial algorithm consisting of at most $O(|F|)$ maximum flow computations.

Keywords: SMD machine optimization, network flow, algorithm

Classification: 90B30, 90C27, 05C21

1. INTRODUCTION

Surface mount technology, compared with manual assembly, brought a real advance into the production of various electronic devices. An important part of the production process is the placement of electronic components onto their prescribed positions on the substrate. This requires a high degree precision when manipulating with tiny elements. Therefore it is usually performed by special sophisticated pick-and-place machines. Such machines are very expensive, so it is economically very important to use them efficiently and maximize their production throughput. The optimization of a pick-and-place machine is a very complicated task, involving several mutually dependent combinatorial optimization problems, many of them hard already in the stand alone setting (like e. g. the quadratic assignment problem [7], traveling salesman problem, bin location and sequencing problems [9] etc.).

Moreover, there are many various types of SMD placement machines that differ in their characteristics and operational modes. For each type, a different, specific optimization problem has to be formulated and solved. A detailed survey of optimization methods applied to different machine technologies can be found in [3]. The authors included more than 80 references and among the methods used they mention various types of heuristics for the involved location and sequencing problems, integer linear programming, genetic algorithms, knowledge-based systems, dynamic

programming, simulated annealing etc.

We study the operation of a special pick-and-place SMD machine that consists of three main parts: a worktable for holding the printed circuit boards (PCBs), feeder racks and a placement head. The PCB is firstly moved onto the worktable by a conveyor belt, where it remains fixed at a predefined location during the placement process. The feeder racks for small components are located on both sides of the PCB conveyor. In addition, on its one side is a tray feeder for large components (like integrated circuits) and on its other side is a vision camera. The placement head of the moving arm has four positions and each one of them can be equipped by a different vacuum nozzle, capable of processing different types of components.

One pick-and-place cycle (briefly: a subtour) consists of the following: the placement head picks one (or none) component by each nozzle, moves above the camera (to check the components, read their real dimensions, the positions of their pins etc) and places them onto their prescribed positions on the PCB. During one subtour, time savings can be achieved by a suitable choice of the sequence of picking and placing the components, but as the number of possibilities is small (remember that in one subtour there are at most 4 components) this task is computationally not so demanding. However, optimization of the throughput of such a machine requires to optimally choose a) the assignment of nozzles to the positions in the placement head, b) the assignment of the components to particular feeders, c) the sequence of the component pick-ups. Changing the nozzles is time consuming, changing the positions of components in feeders has to be done manually, and so it is prone to human mistakes. Practical experiences with the operation of the machine showed, in particular in the production of small series, that relatively greatest time savings can be achieved if the sequence of placed components is chosen so as to minimize the number of idle nozzles in the process which is equivalent to the minimization of the number of subtours.

Hence the optimization problem we deal with in this paper assumes that

- the assignment of components to particular feeders is fixed;
- the assignment of nozzles to the positions on the insertion head is fixed

and we want to minimize the number of subtours needed to handle the given set of components.

More formally, suppose the machine is to assemble a module where k types of components are used. The numbers of components of each type are $q_1, q_2, \ldots, q_k$, respectively, hence the total number of components to be placed onto the module is $Q = \sum_{i=1}^{k} q_i$. The insertion head is equipped with r vacuum nozzles and it is known which nozzle can handle which types of components.

Based on the data of the module we construct a flow network N in the following way. There is a source s and a sink t, vertices $u_1, u_2, \ldots, u_k$ corresponding to component types and vertices $w_1, w_2, \ldots, w_r$ corresponding to the nozzles. For each $i = 1, 2, \ldots, k$ there is an arc (s, u_i) with capacity q_i. The pair (u_i, w_j) is an arc of the network if and only if nozzle j can handle component type i and the capacity of this arc is ∞. There is also one arc (w_j, t) with infinite capacity for each $j = 1, 2, \ldots, r$. (Instead of infinity, it is sufficient to set the respective capacities to Q.)

Each integer $s-t$ flow x of value Q corresponds to a feasible organisation of module assembly. Each arc (s, u_i) has to be saturated in x as each components of type i must be placed. Clearly, the number of such components handled by nozzle j is the value of flow x on arc (u_i, w_j). The values of flow x along the arcs (w_j, t) correspond to the numbers of all components that were handled by nozzle j. Since in one subtour all the nozzles work simultaneously, the number of the subtours corresponding to the assignment of nozzles to particular components defined by flow x is equal to the maximum of flow values on arcs $(w_j, t), j = 1, \ldots, r$.

Hence the minimization of the number of subtours leads to computing a flow of a given size and minimum cost in a bipartite network, where the cost of the flow is equal to the maximum of flow values along a subset of arcs, later to be called *parametrized arcs*. One possibility is to consider this problem as a special combinatorial optimization problem with max-linear objective function; however, they are NP-hard in general [6].

On the other hand, there is a close connection of the minimization of the maximum flow value on the parametrized arcs with the parametric maximum flow. If the capacities of all parametrized arcs are set to be equal to some parameter λ then the optimum cost of the flow is equal to the minimum value of λ that makes the parametric network to admit a flow of the prescribed size. The subtour minimization problem is a special case of a pre-emptive scheduling problem, that was represented by a parametrized maximum flow problem in [5, 10, 13].

Algorithms for parametric maximum flow problem have been considered in several papers. Hamacher and Foulds [11] and Zhang, Ward and Li [15] propose algorithms for simultaneously computing the maximum flows for all values of the parameter in the given interval, however, they do not consider the computational complexity of their algorithms. If the only parametrized arcs are those incident with s (or t) and their capacities are increasing (or decreasing, respectively) functions of the parameter, Gallo, Grigoriades and Tarjan [10] showed how to modify the push-relabel maximum flow algorithm to compute the complete chain of maximum flows (and minimum cuts) for an increasing sequence of parameter values by performing just one maximum flow computation. This algorithm was extended to cases where the capacity functions are piece-wise linear, there are multiple parameters or the subgraph induced by parametrized arcs has an aborescence-shaped structure [13], all parametrized arcs are incident to one (critical) vertex different from s and t [2] and the parametrized arcs may be incident to s,t and one more critical vertex [14].

The low complexity bound of the above algorithms was ensured thanks to the fact that the minimum cuts have a so-called nesting property and hence the number of breakpoints of the function expressing the dependency of maximum flow size on the value of the parameter is not greater than the number of vertices in the network. However, this is due to the special structure of the set of parametrized arcs; if this is not the case, exponentially many breakpoints may occur [4].

We consider a more general version of the parametrized maximum flow problem, where the set of parametrized arcs can be arbitrary, but the capacity of each parametrized arc is equal to λ. We show that in this case it suffices to consider at most as many parameter values as is the number of parametrized arcs and show how

to jump to the next (greater) parameter value.

The organisation of the paper is as follows. The necessary definitions and basic properties for the parametrized maximum flow problem are introduced in Section 2. Section 3 is devoted to the computation of the next parameter value and the efficiency of this approach. We conclude in Section 4.

2. DEFINITIONS AND NOTATION

Suppose we are given a directed network $N = (V, E, s, t, c)$ with the node set V, $|V| = n$, arc set E, $|E| = m$, source and sink $s, t \in V$, arcs capacities $c : E \to \mathbb{N}$. We use a standard terminology and notation used in Graph Theory (see e. g. [12]). In particular, we shall denote by $\delta^+(X)$ the set of arcs leaving a set of vertices $X \subseteq V$, i. e. $\delta^+(X) = \{e = (u, w) \in E; u \in X, w \notin X\}$; the notation $\delta^+(\{u\})$ will be abbreviated to $\delta^+(u)$. A *flow* in N is a function $x : E \to \mathbb{R}_0^+$ satisfying the flow conservation conditions in all vertices except s and t; the *size* of flow x is $v(x) = \sum_{e \in \delta^+(s)} x(e)$. A flow in N is said to be *feasible*, if $x(e) \leq c(e)$ holds for each $e \in E$. The maximum size of a feasible flow in N will be denoted by $v(N)$. A feasible flow is called *acceptable with respect to the given value of* v (or simply *acceptable*, if no confusion can arise) if it is integer and its size $v(x)$ is equal to a given v. Clearly, a necessary and sufficient condition for the existence of an acceptable flow in a network N is $v(N) \geq v$.

Suppose further that a set $F \subset E$ is given, called the set of *parametrized* arcs. An instance of the *Parametrized Maximum Flow Problem*, PMF for short, is a triple (N, F, v) and the task is to find an acceptable flow x in N such that its cost, expressed by the objective function

$$z(x) = \max_{e \in F} x(e), \tag{1}$$

is minimized. $z(N, F, v)$ will denote the cost $z(x)$ of an optimal flow in an instance (N, F, v) of PMF.

An approach to find an optimal flow for the given instance of PMF is a standard technique used in parametric programming, called Newton's method (see also [5, 13]) and it consists of the following: for a given parameter value λ redefine the capacity of each $e \in F$ by

$$c_\lambda(e) = \min\{c(e), \lambda\}$$

and denote the modified network by $N(\lambda)$. (Notice that in what follows, we assume that the capacities of parametrized arcs are big enough, practically not smaller than v, so actually we shall always have $c_\lambda(e) = \lambda$.) Obviously, $z(x) \leq \lambda$ for any feasible flow in $N(\lambda)$. So finding an acceptable flow achieving the minimum value of (1) is equivalent to finding minimum λ such that $N(\lambda)$ admits a feasible flow of size v.

Now appropriately choose a sequence $\lambda_0 < \lambda_1 < \cdots < \lambda_\ell$ (as the first value one can set $\lambda_0 = 0$) of target parameter values until $N(\lambda_i)$ admits an acceptable flow. As $x(e) \leq v$ for each acceptable flow x and each arc e, we have that $z(x) \leq v$. Hence when applying a binary search on integers in the interval $\langle 0, v\rangle$ of possible target values, one needs at most $O(\log\ v)$ iterations, each one consisting of finding a maximum feasible flow in the modified network $N(\lambda_i)$. However, this approach will

not ensure a strongly polynomial algorithm. In particular, when applied to networks corresponding to SMD machine optimization problems, where the number of placed components $v = Q$ is usually much bigger than the size of the network (determined by the number of different component types), this will not be an efficient approach. In what follows we show that for the general PMF the number of necessary iterations can be bounded by the number of parametrized arcs.

3. MINIMUM CUTS IN PARAMETRIZED NETWORKS

An $s - t$ *cut* in a network $N = (V, E, s, t, c)$ is a set of arcs $\delta^+(X)$ determined by a set of vertices $X \subseteq V$ with $s \in X$ and $t \notin X$. The *capacity* of cut $\delta^+(X)$ is

$$c(\delta^+(X)) = \sum_{e \in \delta^+(X)} c(e).$$

The following theorem is the basic result in the theory of network flows ([8], see also [12], Lemma 8.3 and Theorem 8.6).

Theorem 3.1. Let x be any $s - t$ flow and $\delta^+(X)$ any $s - t$ cut in a network N. Then $v(x) \leq c(\delta^+(X))$. Moreover, $v(N)$ is equal to the minimum capacity of an $s - t$ cut.

Let λ be any nonnegative number. Then for the capacity $c(\delta^+(X))$ of a cut $\delta^+(X)$ in $N(\lambda)$ we have

$$c(\delta^+(X)) = c^*(\delta^+(X)) + \lambda\xi(\delta^+(X))$$

where $c^*(\delta^+(X))$ is the *capacity of non-parametrized arcs* in $\delta^+(X)$ and $\xi(\delta^+(X))$ is the *number of parametrized arcs* in $\delta^+(X)$. Denote by $\mathfrak{P}(\lambda)$ the set of all cuts with minimum capacity in $N(\lambda)$. We shall also use $v(\lambda)$ to denote the size of a maximum feasible flow in $N(\lambda)$.

Theorem 3.2. Let λ be such that $v(\lambda) < v$. Then

(i) If there exists a minimum capacity cut in $N(\lambda)$ containing no parametrized arc then there exists no acceptable flow in N.

(ii) Let each minimum capacity cut in $N(\lambda)$ contain at least one parametrized arc. If N admits an acceptable flow then

$$z(N, F, v) \geq \max_{\delta^+(X) \in \mathfrak{P}(\lambda)} \left\lceil \frac{v - c^*(\delta^+(X))}{\xi(\delta^+(X))} \right\rceil. \tag{2}$$

Proof. (*i*) is implied by Theorem 3.1.
(*ii*) If N admits an acceptable flow then there exists λ^* such that $N(\lambda^*)$ admits a feasible flow of size v. This means that for each cut $\delta^+(X)$ in $N(\lambda^*)$:

$$c^*(\delta^+(X)) + \lambda^*\xi(\delta^+(X)) \geq v > v(\lambda).$$

Hence a necessary condition for $N(\lambda^*)$ to admit an acceptable flow is that

$$\lambda^* \geq \frac{v - c^*(\delta^+(X))}{\xi(\delta^+(X))}$$

for each cut $\delta^+(X)$ in $N(\lambda^*)$ with $\xi(\delta^+(X)) > 0$. As each cut in $\mathfrak{P}(\lambda)$ fulfills $\xi(\delta^+(X)) > 0$ and each acceptable flow is integer, inequality (2) follows. □

However, the previous Theorem alone does not lead to an efficient algorithm, as it requires to find not just one minimum capacity cut, but to find one that fulfills an additional optimality criterion. In the following Theorem we show that to get a new target parameter value λ_{i+1}, it suffices to take *any* minimum capacity cut $\delta^+(X)$ in $N(\lambda_i)$. Moreover, the number of parametrized arcs in any minimum capacity cut in $N(\lambda_{i+1})$ will be smaller than in $\delta^+(X)$.

Theorem 3.3. Let λ_i be any integer such that $v(\lambda_i) < v$ and let $\delta^+(X) \in \mathfrak{P}(\lambda_i)$ with $\xi(\delta^+(X)) > 0$ be arbitrary. Denote

$$\lambda_{i+1} = \left\lceil \frac{v - c^*(\delta^+(X))}{\xi(\delta^+(X))} \right\rceil. \tag{3}$$

Then

(i) $\lambda_{i+1} > \lambda_i$;

(ii) if $v(\lambda_{i+1}) < v$ then $\xi(\delta^+(Y)) < \xi(\delta^+(X))$ for each cut $\delta^+(Y) \in \mathfrak{P}(\lambda_{i+1})$;

(ii) if $\xi(\delta^+(X)) > 0$ for each $\delta^+(X) \in \mathfrak{P}(\lambda_i)$ then $v(\lambda_{i+1}) > v(\lambda_i)$.

P r o o f. (*i*) Since $\delta^+(X)$ is a minimum capacity cut in $N(\lambda_i)$, Theorem 3.1 implies

$$c^*(\delta^+(X)) + \lambda_i \xi(\delta^+(X)) = v(\lambda_i),$$

hence, as λ_i is integer and $v(\lambda_i) < v$, we get

$$\lambda_i = \frac{v(\lambda_i) - c^*(\delta^+(X))}{\xi(\delta^+(X))} < \left\lceil \frac{v - c^*(\delta^+(X))}{\xi(\delta^+(X))} \right\rceil = \lambda_{i+1}$$

since rounding the second fraction up keeps the strict inequality.

(*ii*) As $\delta^+(X)$ is a minimum cut in $N(\lambda_i)$ and $\delta^+(Y)$ is also a cut in this network, we have

$$c^*(\delta^+(X)) + \lambda_i \xi(\delta^+(X)) \leq c^*(\delta^+(Y)) + \lambda_i \xi(\delta^+(Y)). \tag{4}$$

Further, definition of λ_{i+1} and the assumptions of this Theorem imply

$$c^*(\delta^+(X)) + \lambda_{i+1} \xi(\delta^+(X)) \geq v > v(\lambda_{i+1}) = c^*(\delta^+(Y)) + \lambda_{i+1} \xi(\delta^+(Y)). \tag{5}$$

By subtracting (4) from (5) we get

$$(\lambda_{i+1} - \lambda_i)\xi(\delta^+(X)) > (\lambda_{i+1} - \lambda_i)\xi(\delta^+(Y)),$$

or, equivalently

$$\big(\xi(\delta^+(X)) - \xi(\delta^+(Y))\big)\,(\lambda_{i+1} - \lambda_i) > 0.$$

As $\lambda_{i+1} - \lambda_i > 0$, we get $\xi(\delta^+(X)) - \xi(\delta^+(Y)) > 0$.

(*iii*) It is sufficient to show that the capacity of each cut $\delta^+(Y)$ in $N(\lambda_{i+1})$ is greater than $v(\lambda_i)$. This is trivial, if $\delta^+(Y) \notin \mathfrak{P}(N(\lambda_i))$. On the other hand, if $\delta^+(Y) \in \mathfrak{P}(N(\lambda_i))$, then $\xi(\delta^+(Y)) > 0$ and (*i*) implies the desired property. □

Corollary 3.4. To find an optimal solution in an instance (N, F, v) of the PMF or to determine that no acceptable flow exists, $O(|F|)$ maximum flow computations are sufficient.

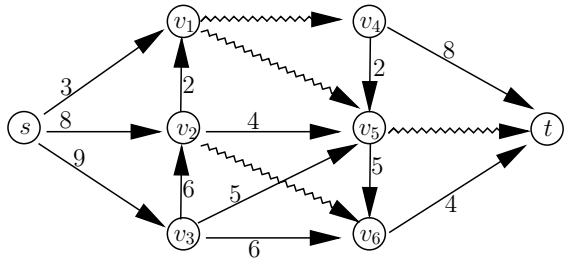

Fig. 1. Example instance.

Now we present an illustration of the sequence of networks $N(\lambda_i)$. The example instance (N, F, v) is shown in Figure 1. Parametrized arcs are represented by wavy lines, the prescribed flow value $v = 18$. Here and in the following figures, the numbers accompanying the arcs are their capacities, the second number after slash denotes the flow along the arc. If the flow along an arc is 0, the second number is simply missing. Similarly, no capacity and/or flow is given for parametrized arcs, if it is equal to 0. Vertices with the gray background represent the set X of a minimum cut in the network, the cut arcs also have a gray background.

Figure 2 shows the maximum flow in $N(\lambda_0)$ for $\lambda_0 = 0$. According to Theorem 3.3, $\lambda_1 = (18 - 4)/2 = 7$. In Figure 3 one can see a maximum flow in $N(\lambda_1)$ and from here we also get $\lambda_2 = (18 - 9)/1 = 9$. Finally, Figure 4 shows a minimum cut containing no parametrized arcs in $N(\lambda_2)$. Since $v(\lambda_2) = 18$, we got an acceptable flow x with $z(x) = 9$.

4. CONCLUSION

In this paper we extended the study of the Parametric Maximum Flow problem to a general structure of the set of parametrized arcs, so far not considered in literature. We derived a simple formula for the next parameter value to be considered in Newton's algorithm and we proved that the number of parametrized arcs in the

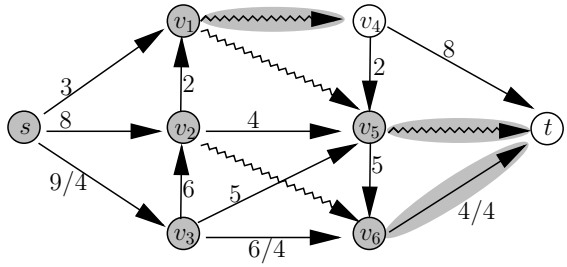

Fig. 2. $\lambda_0 = 0$.

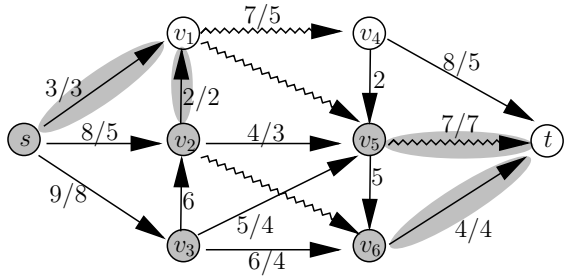

Fig. 3. $\lambda_1 = 7$.

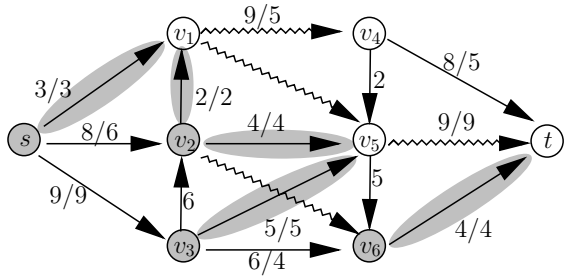

Fig. 4. $\lambda_2 = 9$.

corresponding minimum capacity cuts strictly decreases. Hence at most $O(m)$ maximum flow computations are necessary to compute the minimum cost of an acceptable flow or to determine that the given network admits none.

Still some open problems remain.

- In [5, 10, 13, 14] a more general dependence of the capacities of parametrized arcs on the parameter value is studied. If we suppose that these capacities are linear functions of the parameter of the form $c_\lambda(e) = c_e\lambda$ for positive coefficients c_e, formula (2) could be modified to

$$z(N, F, v) \geq \max_{\delta^+(X)\in\mathfrak{P}(\lambda)} \left\lceil \frac{v - c^*(\delta^+(X))}{\sum\limits_{e\in\delta^+(X)} c_e} \right\rceil . \tag{6}$$

  One can then show, similarly as in Theorem 3.3, that when moving to the new parameter value computed according to (6), the value $\sum_{e\in\delta^+(X)} c_e$ will decrease for the minimum capacity cut. However, this does not ensure the bound $O(m)$ for the number of relevant parameter values, as there may be exponentially many, in the cardinality of the set F, different sums of its subsets. Is it still possible to achieve a polynomial algorithm in this case?

- Exploiting the special structure of the set of parametrized arcs, it was shown that suitable extensions of the push-relabel algorithm of Goldberg and Tarjan enable to compute the minimum parameter value admitting an acceptable flow in a single max-flow computation [10, 13, 14]. While we doubt that a similar algorithm could be possible in the general case, it would be interesting to find other structures admitting an algorithm with the same bound.

ACKNOWLEDGEMENT

This work was supported by VEGA grants 1/0035/09, 1/0325/10, by the Agency of the Slovak Ministry of Education for the Structural Funds of the EU under project ITMS:26220120007 (Cechlárová), Hungarian National Research Fund OTKA K69027, MTA-ELTE Egerváry Research Group (Fleiner) and Slovak–Hungarian grant SK-HU-003-08 of the Slovak Research and Development Agency.

(Received December 7, 2010)

REFERENCES

[1] R. H. Ahuja, T. L. Magnanti, and J. B. Orlin: Network flows, Theory, Algorithms and Applications. Prentice Hall, Englewood Cliffs 1993.

[2] T. Arai, S. Ueno, and Y. Kajitani: Generalization of a theorem on the parametric maximum flow problem. Discrete Appl. Math. *41* (1993), 69–74.

[3] M. Ayob and G. Kendall: A survey of surface mount device placement machine optimisation: Machine classification. European J. Oper. Res. *186* (2008), 893–914.

[4] P. J. Carstensen: Complexity of some parametric integer and network programming problems. Math. Programming *26* (1983), 64–75.

[5] Y. L. Chen: A parametric maximum flow algorithm for bipartite graphs with applications. European J. Oper. Res. *80* (1995), 226–235.

[6] S. J. Chung, H. W. Hamacher, F. Maffioli, and K. G. Murty: A note on combinatorial optimization problems with max-linear objective function. Discrete Appl. Math. *42* (1991), 139–145.

[7] E. Duman and I. Or: The quadratic assignment problem in the context of the printed circuit board assembly. Comput. Oper. Res. *34* (2007), 163–179.

[8] L. R. Ford and D. R. Fulkerson: Maximal flow through a network. Canad. J. Math. *8* (1956), 399–404.

[9] L. R. Foulds and H. W. Hamacher: Optimal bin location and sequencing in printed circuit board assembly. European J. Oper. Res. *66* (1993), 279–290.

[10] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan: A fast parametric maximum flow algorithm and applications. SIAM J. Comput. *18* (1989), 30–55.

[11] H. W Hamacher and L. R. Foulds: Algorithms for flows with parametric capacities. ZOR – Methods and Models Oper. Res. *33* (1989), 21–37.

[12] B. Korte and J. Vygen: Combinatorial Optimization, Theory and Algorithms. Springer, Berlin 2008.

[13] S. T. McCormick: Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. Oper. Res. *47* (1999), 744–756.

[14] M. G. Scutella: A note on the parametric maximum flow problem and some related reoptimization issues. Ann. Oper. Res. *150* (2007), 231–244.

[15] B. Zhang, J. Ward, and Q. Feng: A simultaneous parametric maximum-flow algorithm for finding the complete chain of solutions. Hewlet-Packard Development Company, Preprint, 2005.

Katarína Cechlárová, Institute of Mathematics, Faculty of Science, P. J. Šafárik University, Jesenná 5, 040 01 Košice. Slovak Republic.

e-mail: katarina.cechlarova@upjs.sk

Tamás Fleiner, Department of Computer Science and Information Theory, Budapest University of Technology and Economics, H-1117, Magyar tudósok körútja 2, Budapest. Hungary.

e-mail: fleiner@cs.bme.hu