Simultaneous Parametric Maximum Flow Algorithm with Vertex Balancing

Bin Zhang, Julie Ward, Qi Feng
Intelligent Enterprise Technologies Laboratory
HP Laboratories Palo Alto

HPL-2005-121
June 28, 2005*

maximum flow, networks, parametric flow networks, graphs, optimization, selection, sequencing

Two new algorithms, SPMF^{simple} and SPMF^{fast}, for finding the complete chain of solutions of the selection model are presented in this paper. A special kind of residual path, called a λ-directed simple residual path, is identified to be the only kind of residual path necessary for SPMF^{simple}. By augmenting the right amount of flows along the λ-directed simple residual paths, the new algorithms are monotone convergent. SPMF^{fast} replaces the path-wise flow augmentation by flow-redistribution at each node, which provides a factor of ten speed-up for all the large datasets tested.

Simultaneous Parametric Maximum Flow Algorithm With Vertex Balancing

Bin Zhang, Julie Ward, Qi Feng

Hewlett-Packard Laboratories

1501 Page Mill Rd, Palo Alto, CA 94086

{bin.zhang2, jward, qfeng@hp.com}

Abstract. Two new algorithms, SPMF^{simple} and SPMF^{fast}, for finding the complete chain of solutions of the selection model are presented in this paper. A special kind of residual path, called a λ-directed simple residual path, is identified to be the only kind of residual path necessary for SPMF^{simple}. By augmenting the right amount of flows along the λ-directed simple residual paths, the new algorithms are monotone convergent. SPMF^{fast} replaces the path-wise flow augmentation by flow-redistribution at each node, which provides a factor of ten speed-up for all the large datasets tested.

1. Introduction

In the *selection problem*, introduced by Balinski [1970] and Rhys [1970], a finite set of items $P = \{p_i\}_{i=1}^{n_1}$ and a finite collection $O = \{o_j \mid o_j \subseteq P\}_{j=1}^{n_2}$ are given. There is a cost c_p associated with each item $p \in P$, and a benefit $R_o \geq 0$ conferred by each subset $o \in O$. The objective of the selection problem is to choose a group of items in P to maximize the net benefit of the selection. This problem can be stated mathematically as follows: let x_p be an indicator variable, taking values in $\{0,1\}$, associated with the selection of item p. Let variable $y_o \in \{0,1\}$ indicate whether all items in subset o are selected. Then the following integer programming model defines an optimal selection:

$$\max \sum_o R_o y_o - \sum_p c_p x_p \text{ s.t. } y_o \leq x_p \text{ for all } p \in o \text{ and } x_p, y_o \in \{0,1\} \quad (1.1)$$

Rhys [1970] showed that since the constraints of this integer program are totally unimodular, every extreme point is integral, and thus its linear programming relaxation will produce integer solutions. Thus, the selection problem can be stated more simply as a linear program:

$$\max \sum_o R_o y_o - \sum_p c_p x_p \quad \text{s.t. } y_o \leq x_p \text{ for all } p \in o \text{ and } 0 \leq x_p, y_o \leq 1. \quad (1.2)$$

The selection problem arises in many applications. Rhys introduced the problem in the context of freight terminal selection, in which one must select freight terminals to construct from a given set of candidate terminals, each of which has a construction cost. The construction of a pairs of terminals allows freight service to be offered between them, and thus a benefit in the form of expected service revenue is associated with each pair. The goal of the freight terminal selection problem is to choose a set of terminals to maximize the benefit minus the construction cost.

In many applications, the cost of selecting an item p consists of a cost c_p times a nonnegative weighting parameter λ. As the parameter λ varies, the relative importance of the revenue and cost components of the objective varies. The problem of finding a solution to the selection problem for all values of λ can be thought of as a *parametric selection problem*, stated as

$$\text{LR}(\lambda): \quad \max \sum_o R_o - \lambda \sum_p c_p x_p \text{ s.t. } y_o \leq x_p \text{ for all } p \in o \text{ and } 0 \leq x_p, y_o \leq 1. \quad (1.3)$$

One application of the parametric selection problem is in database record segmentation, proposed by Eisner and Severance [1976]. In this problem, the items in P are data items to be stored in a large shared database. Users retrieve subsets $o \in O$ of data items, and derive benefit if all items in a subset o reside in primary memory. The cost of storing item p in primary memory is λ times the size of item p, where λ is a tradeoff parameter between storage requirements and user benefit. The goal is to choose how to partition items among primary and secondary memory.

Another application, due to Stone [1978], is in determining the critical load factor in two-processor systems. The items in this context are program modules, each of which must be assigned to one of two computer processors, A or B. A module has a different execution cost depending on the processor to which it is assigned. The subsets $o \in O$ are pairs of program modules that need to communicate with each other, with communication costs if the two members of the pair are assigned to different processors. The parameter λ is the load factor, the fraction of time that processor A delivers useful cycles. The goal is to assign program modules to processors for each value of the load factor to minimize the execution costs and communication costs for each program module.

Mamer and Smith [1982] proposed an application to optimal repair kit selection. Items are tools or parts, with associated annual cost of carrying the tool or part in inventory. The sets correspond to repair jobs; each set contains tools and/or parts required for a particular repair job. Each repair job has an expected annual penalty associated with being unable to perform the job. The problem is to select the tools and parts for a repair kit to minimize expected annual penalties plus the inventory cost of the parts.

Determining how many more games our baseball team can lose partway through a season without being eliminated from finishing in first place can also be viewed as a parametric selection problem, as was shown by Gusfield and Martel [1992]. The items in this model are teams (other than our team), and the subsets are pairs of teams (not involving our team) that have remaining games to be played in the season.

Zhang, Ward and Feng [2004 & 2005] applied the parametric selection problem to product portfolio selection. A company wants to select a portfolio of products to offer to maximize the total benefit (eg, revenue or profit margin) of historical orders "covered" by the portfolio, minus some penalty λ associated with the portfolio size. Thus, the items are products, and the subsets are orders. Varying λ affects the relative importance of order benefit coverage and portfolio size.

A more detailed review of applications can be found in Hochbaum [2004].

In the remainder of the paper, we focus on the version of the selection problem in which the item costs $c_p = 1$ for each item $p \in P$. The results apply to the more general case.

1.2 Solving the Selection Model by Parametric Maximum Network Flows

Balinski [1970] showed that a selection problem is equivalent to the problem of finding a minimum cut in a particular bipartite network, illustrated in Figure 1.

The equivalence between the LR(λ) problem and the minimum cut problem is established by observing that $\min \sum_o R_o(1-y_o) + \lambda \sum_p x_p$ is the same as $\max \sum_o R_o y_o - \lambda \sum_p x_p$. It is a well-known result of Ford and Fulkerson [1956] that the value of a maximal flow equals the value of a minimum cut.

A capacitated flow network $\Omega_\lambda = \{N, E, C_\lambda, F\}$, where $V = P \cup O \cup \{s,t\}$, a source node s is at the far left and a sink node t at the far right (see Figure 1). Adjacent to the source node is the set of p-nodes. Adjacent to the sink node is the set of o-nodes. The set of arcs $E = \{<s,p> \mid p \in P\} \cup \{<p,o> \mid p \in o, o \in O\} \cup \{<o,t> \mid o \in O\}$. Let $n_1 = |P|$ and $n_2 = |O|$, the cardinalities of P and O respectively, and $m = |E|$, the number of arcs. The capacities on the arcs incident to s are all equal to λ. The capacity of the arc from o to t is R_o for any o. The capacity of arcs between a $p \in P_o$ and the o-node is +infinity. $f_{v,w}$ is the flow from any node v to any other node w.

A st-cut is a partition of the nodes into two subsets – the s-partition containing s and the t-partition containing t. The capacity of a st-cut is the sum of the capacities of arcs going from the nodes in the s-partition to the nodes in *t*-partition. A minimum cut is a *st*-cut with minimum capacity.

As we allow λ to vary, the problem LR(λ) becomes a parametric maximum flow problem. There are several known algorithms for parametric maximum flow problems, including that of Gallo, Grigoriadis and Tarjan [1986] and slightly improved algorithms by Gusfield and Martel [1992], Gusfiled and Tardos [1994]. In most of these algorithms, a series of maximum flow problems is solved, and the algorithm makes use of the previous problem's solution to speed up the solution at the next parameter value. By comparison, the algorithm presented in the next section finds the maximum flow in the network for all *breakpoints*, at which the set of edges in the minimum cut changes, of the parameter values simultaneously. The algorithm presented in this paper is much simpler and easier to implement in comparison to all prior parametric maximum flow algorithms.

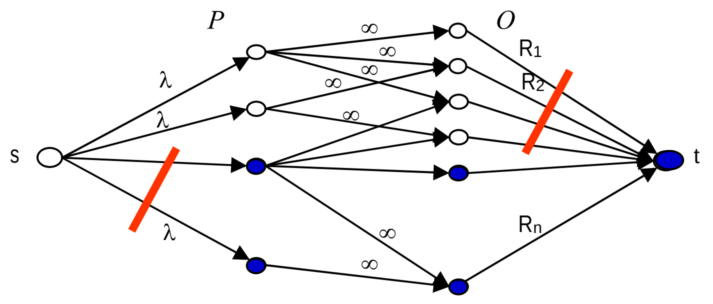

Figure 1. A bipartite minimum-cut/maximum flow problem corresponding to the Lagrangian relaxation LR(λ).

2. SPMF$^{\text{simple}}$ -- A Parametric Bipartite Maximum Flow Algorithm

A simple version SPMF$^{\text{simple}}$ of the simultaneous parametric maximum flow algorithm is presented in this section. A more general version of this algorithm that applies to more general capacities, both parametric and non-parametric ones, can be found in Zhang at el [2004].

SPMF$^{\text{simple}}$ works with a derived non-parametric network instead of the original parametric network. For the special case arising from selection models, the derivation is obtained by simply removing all the λ dependent capacities on the arcs incident to the source by +infinity, which is the first step of the algorithm as shown in Figure 2.

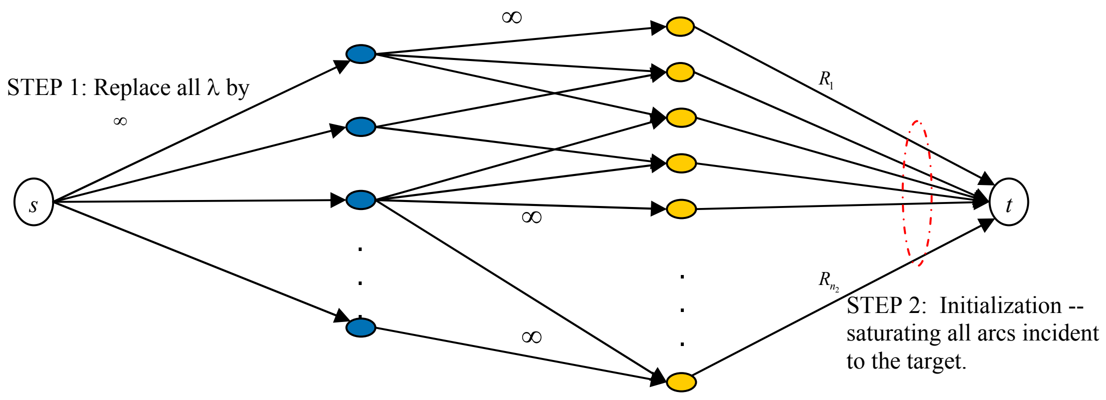
∞
STEP 1: Replace all λ by ∞
s
t
R_1
∞
∞
R_{n_2}
STEP 2: Initialization -- saturating all arcs incident to the target.

Figure 2. The derived non-parametric network.

The second step initializes the flows in the derived network. The initial flows in the derived network are set to saturate all arcs incident to the target. Since all other arcs not incident to the target have capacity of +infinity, this step is straightforward and does not require any special algorithm. After initialization, the total flow through the network remains fixed forever, which is always equal to the total flow to the target right after the initialization, or $\sum_{o\neq\varnothing} R_o$. Different initializations may have an impact on the amount of time the algorithm will take but it will not have any impact on the correctness of the algorithm.

The third step is the main body of the SPMF$^{\text{simple}}$ algorithm. It redistributes the flows through the arcs incident to the source in a regulated way which in the end will ultimately result in a *special state* of the flows F in the derived network, from which all minimum cuts and their associated maximum flows at all breakpoints of the parameter λ, in the original network, can be found from one linear scan of the vertices and the arcs in the derived network..

In the derived network, we define $\lambda_i \equiv f_{s,p_i}$. A residual path is called a λ-directed simple residual path if it consists a simple loop as $s \rightarrow p_i \rightarrow o \rightarrow p_j \rightarrow s$ with $\lambda_i = f_{s,p_i} < \lambda_j = f_{s,p_j}$ and $f_{p_j,o} > 0$.

The rules for augmenting the flows are

a) flows are augmented to only λ-directed simple residual paths,

b) the amount of flow to augment to *path* is

$$\delta = \min\{\hat{c}(path), (\lambda_j - \lambda_i)/2\}, \tag{2.1}$$

where $\hat{c}(path)$ is the residual capacity of the path.

We call such an operation a *move.* The moves continue as long as there are λ-directed simple residual paths in the network. From the rule b), it is clear that the order of the two λ-values involved in the residual path is never reversed after a move.

A particular implementation of SPMF$^{\text{simple}}$ algorithm:

Step 1: Initialization: All *p's* nodes are put into a Round-Robin queue, *Queue*, which is all active *p's*.

All arcs <*o,t*> are saturated.

Step 2: get current_p from the head of the *Queue*, scan all *s→current_p→o→p→s* paths. If a λ-directed simple residual path is found, make a move on the path; if inactive(p), set *p* active.

Step 3: If no move happened in the scan in Step 2, set *current_p* inactive.

Step 4: If *Queue* is empty, stop, else goto Step 2.

SPMF$^{\text{simple}}$ is proven to have monotonic convergence after each operation (Zhang at el. [2004] [2005]). Round-Robin queue of all *p*-nodes ensures that all λ-guided simple residue paths are visited.

3. *SPMF^{fast}* -- Redistributing Flows on All Arcs Incident to Each *o*

A much faster version of *SPMF* is presented in this section. The idea is to redistribute the flows on all arcs incident to a node *o* in a single operation such that no λ-directed simple residual path going through *o* remains.

SPMF$^{\text{fast}}$ is described in steps as follows:

Step 1: initialize all flows to zero. Set the status of all vertices $o \in O$ active.

Step 2: Run *CALC(o),* to be defined next, on each active *o* in the Round-Robin fashion, if *CALC(o)* return the message "nothing is changed" on a node *o*, set the status of *o* as "inactive". Step 2 is continued until no more active *o*'s left in the Round-Robin queue.

Step 3: Applying *CALC(o)* on each *o* to make sure that there is nothing left to be done (after an *o* is deactivated, its status could be changed by later operations). If *CALC(o)* does not return the message "nothing is

changed" on any *o*, return it to the Round-Robin queue.
Step 4: If the Round-Robin queue is empty, DONE, otherwise go to Step 2.

The subroutine *CALC(o)* is to find the state of flows on all the arcs incident to *o* so that no λ-directed simple residual path going through *o* remains after the redistribution.

CALC(o): Assume that there are K arcs incident to o and $p_1,...,p_K$ are the K *p*-nodes at the other end of these arcs as illustrated in Figure 4.

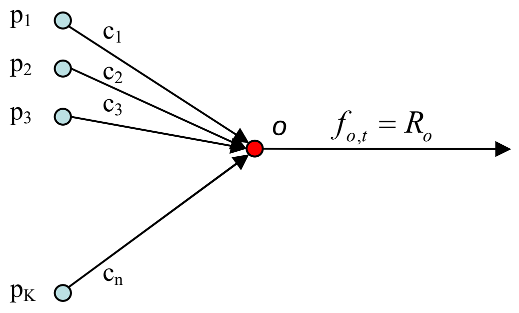

Figure 4. The situation at one *o*-node.

Let

$$\mu_k = f_{s,p_k} - f_{p_k,o}, k = 1,...,K \tag{3.1}$$

and from the flow balance condition at *o*,

$$f_{o,t} = \sum_{k=1}^{K} f_{p_k,o} \tag{3.2}$$

the total flow arrive at *o* (zero when just start).

First we sort the μ_k's in increasing order; without loss of generality, assume that they are sorted: $\mu_1 \le \mu_k ... \le \mu_K$. We assign the total flow $(= R_o)$ to the arcs so that no λ-directed simple residual paths is created. This is done by split the total flow $f_{o,t} = R_o$ as

$$R_o = \sum_{k=1}^{K'} f'_{p_k,o}, K' < K \text{ and } f'_{p_k,o} = 0, K' < k < K \tag{3.3}$$

where $f'_{p_k,o}$ are the new flows to be assigned by *CALC(o)*. The new λ-values will be

$$\begin{aligned} \lambda'(p_k) &= \mu_1 + f'_{s,p_k}, 1 \le k \le K' \\ \lambda'(p_k) &= \mu_k, K' < k \le K \end{aligned} \tag{3.4}$$

To eliminate all the λ-directed simple residual paths in this sub-graph, we must have

$$\lambda'(p_1) = ... = \lambda'(p_{K'}) \le \lambda'(p_{K'+1}) ... \le \lambda'(p_K) . \tag{3.5}$$

The system of equations in (4.5) can be solved in time linear to K and the flows are calculated from the new λ-values,

$$f'_{p_k,o} = \begin{cases} \lambda'(p_k) - \mu_k & 1 \le k \le K' \\ 0 & K' < k \le K \end{cases}. \tag{3.6}$$

The cost of calling *CALC(o)* is *O(K *log(K))* because of the sorting involved, where *K* is the degree of node *o*.

Theorem 1: $\sum_{k=1}^{K} \lambda^2(p_k)$ is monotone decreasing under the operation of *CALC(o)*.

Proof: Actually, after *CALC(o)*, the new λ-values is the solution of $\min \sum_{k=1}^{K} \lambda^2(p_k)$. The minimization is over all possible choices of λ-values that do not violate the flow balance equation at vertex *o*. The rest of the proof is done by direct verification of $\sum_{k=1}^{K_b} \left[\lambda^2(a_k) - \lambda'^2(a_k) \right] \ge 0$.

4. Experimental Comparisons

We compare the new SPMF$^{\text{fast}}$ algorithm with the SPMF$^{\text{simple}}$ algorithm. The characteristics of the data sets are shown in Table 1. There are no multiarcs in the data sets. All data sets are from real-world problems. All timings are shown in seconds. Data loading time is common to both algorithms and excluded from the comparison. Timing started right before the initialization step of the algorithms. The experiments are carried out on HP Workstation XP8000. The best maximum flow implementation is by Cherkassky & Goldberg [1997], which calculates a single minimum cut at one particular λ-value. We run their code on the same machine with timing excludes the data input reading time and output printing time. Timing started just before the algorithm initialization and ended just before output printing.

Table 1. The major characteristics of the data sets.

Data Set	**|*P*|**	**|*O*|**	**#arcs**	#Minimum Cuts Found by SPMFs	SPMF$^{\text{simple}}$ *(seconds)*	SPMF$^{\text{fast}}$ *(seconds)*	Goldberg Maximum Flow (one min-cut)
D1	263	39,106	416,481	189	10.7	0.67	0.313
D2	232	53,565	572,134	155	8.9	0.94	0.47
D3	344	123,933	1,280,298	222	18.9	1.88	0.844
D4	439	286,604	3,111,773	298	118	9.1	2.60

5. Conclusions

Both SPMF algorithms were implemented in C++. Their performances, as shown in Section 6, are very fast even with real data set with above one million arcs and above 100,000 nodes. However, without access to an implementation of the parametric maximum flow algorithm by Gallo at el, experimental comparison with their algorithm is still missing.

Another advantage of SPMF algorithms is its simplicity which made its implementation very easy as shown in Section 7.

SPMF has been generalized to more general network flow problems with general capacity constraints for both the constant and parametric constraints (which were first used in Gallo [1989], see Zhang [2004]).

Acknowledgement We thank Yunhong Zhou for Proofreading this document and his comments.

References

Ahuja, R.K., Magnanti, T.L., & Orlin, J.B. [1993], Network Flows, Prentice Hall, New Jersey.

Ahuja, R.K., Orlin, J., Stein, C. and Tarjan, R. [1994], Improved algorithms for bipartite network flow. *SIAM Journal on Computing,* **23**, pp. 903-933

Balinksi, M. L. [1970], On a selection problem. *Management Science,* 17:3, 230-231.

Cherkassky, B.V. and Goldberg, A.V. [1997], On Implementing the Push-Relabel Method for the Maximum Flow Problem, *Algorithmica* 19(4):390-410

Eisner, M.J., and Severance, D.G. [1976], Mathematical Techniques for Efficient Record Segmentation in Shared Databases, J. Assoc. Comput. Mach*.,* 23, pp. 619-635.

Elias, P., Feinstein, A., & Shannon, C.E. (1956), Note on Maximum Flow Through a Network, IRE Transformations on Information Theory IT-2, 117-119.

Ford & Fulkerson 1956. Maximum Flow Through a Network, *Canadian Journal of Mathematics* 8, 339-404.

Ford, L.R. & Fulkerson, D.R. [1962], Flows in Networks, Princeton University Press, Princeton, NJ.

Gallo, G., Grigoriadis, M. D., & Tarjan R.E. [1989], A Fast Parametric Maximum Flow Algorithm and Applications, SIAM J. Computing, Vol. 18, No. 1, pp. 30-55, February.

Goldberg, A.V. & Tarjan, R.E. [1988], A New Approach to the Maximum Flow Problem, Proc. 18th Annual ACM Symposium on Theory of Computing, 1986, pp. 136-146; J. Assoc. Comput. Mach., 35.

Gusfield, D. and Martel, C [1992], A fast algorithm for the generalized parametric minimum cut problem and applications, Algorithmica, 7:499-519.

Gusfield, D. and Tardos, E. [1994], A faster parametric minimum-cut algorithm, Algorithmica 11:278-290.

Hochbaum, D. [2004], Selection, provisioning, shared fixed costs, maximum closure, and implications on algorithmic methods today, Management Science 50:6, pp 709-723.

Mamer, J.W. and Smith, S.A. [1982], Optimizing field repair kits based on job completion rate, Management Science 28:11, pp 1328-1333.

Rhys, J. M. W. 1970. A selection problem of shared fixed costs and network flows. *Management Science,* 17:3, 200-207.

Zhang, B., Ward, J. and Feng, Q. (2004), A Simultaneous Parametric Maximum Flow Algorithm for Finding the Complete Chain of Solutions, http://www.hpl.hp.com/techreports/2004/HPL-2004-189.html

Zhang, B., Ward, J. and Feng, Q. (2005), A Simultaneous Maximum Flow Algorithm for the Selection Model, http://www.hpl.hp.com/techreports/2004/HPL-2005-91.html