AMO - Advanced Modeling and Optimization, Volume 4, Number 2, 2002

SOME BUDGET APPLICATIONS OF THE MAXIMAL FLOW

Sabin TABIRCA* Tatiana TABIRCA**

*University College Cork, Computer Science Department
**University of Manchester, Computer Science Department
email: s.tabirca@cs.ucc.ie

Abstract. The purpose of this article is to introduce mathematical models for two budget problems. The first problem, which deals to the construction of a particular budget, is solved based on an algorithm for the maximum flow problem in bipartite network. The method constructs an efficient budget by repeating a maximum flow computation. The second problem is the *budget exercise* and studies how the budget can be modeled in time. A mathematical model based on the maximal dynamic flow is proposed and an algorithm to solve it is outlined.
Key words: Maximum flow problem, bipartite network, budget simulation..

Network flow problems are major problems in operational research, computer science and engineering. Within these the maximum flow problem is one of the most important problem, having several real world applications. Since Ford and Fulkerson [1956] introduced the maximum flow problem, much work has been done either to find new solutions or to develop efficient algorithms or to propose applications of it. This article follows one of these directions proposing two applications of the maximum flow problem for some budget problems.

1. INTRODUCTION

The aim of this section is to review some results concerning the maximum flow problem in a bipartite network. This particular flow problem reflects the simplest case when the flow is sent from the set of source nodes to the set of sink nodes not using intermediate nodes. Let $S=\{s_1, s_2,\dots ,s_n\}$ and $T=\{t_1, t_2,\dots ,t_m\}$ be two disjoint sets representing the source nodes and the sink nodes.

Definition 1.

The network $R=(G=(S \cup T, E), x, y, c)$ is named bipartite if

a) $G=(S \cup T, E)$ is a bipartite graph.
b) $c : E \rightarrow (0,\infty)$ is a capacity function for the arcs E.
c) $x = (x[i],\ i =1,n)$ is the source array, where $x[i]$ is the maximum amount of flow of the node s_i.
d) $y = (y[i], i=1,m)$ is the sink array, where $y[i]$ is the maximum amount of flow to sink node t_i.

The function $c:E \rightarrow (0,\infty)$ can be extended to a matrix $c\in M_{n,m}([0,\infty))$ as follows

$$c_{i,j} = \begin{cases} 0, (i,j) \notin E \\ c((i,j)), (i,j) \in E \end{cases}. \tag{1}$$

Definition 2.

A matrix $f\in M_{n,m}([0,\infty))$ is a flow in the bipartite network $R=(G=(S \cup T, E), x, y, c)$ if the following equations are true:

$$0 \le f_{i,j} \le c_{i,j}\ (\forall i = 1,n)(\forall j = 1,m) \tag{2a}$$

$$\sum_{j=1}^{m} f_{i,j} \le x[i] (\forall i = 1,n) \tag{2b}$$

$$\sum_{i=1}^{n} f_{i,j} \le y[j] (\forall j = 1,m) \tag{2c}$$

If f is a flow in R then the value of the flow f is $v(f) = \sum_{i=1}^{n}\sum_{j=1}^{m} f_{i,j}$.

The maximum flow problem in a bipartite network finds the flow f* that maximises the value $v(f)=\sum_{i=1}^{n}\sum_{j=1}^{m} f_{i,j}$. This problem can be also approached as a linear programming problem, as showed in (3).

$$\max \quad v(f)=\sum_{i=1}^{n}\sum_{j=1}^{m} f_{i,j}$$
$$\begin{cases} \sum_{i=1}^{n} f_{i,j} \le y[j](\forall j=1,m) \\ \sum_{j=1}^{m} f_{i,j} \le x[i](\forall i=1,n) \\ 0 \le f_{i,j} \le c_{i,j}\ (\forall i=1,n)(\forall j=1,m) \end{cases} . \tag{3}$$

Tabirca [1998a] studied this problem proposing a generic algorithm for its solving. Moreover, Tabirca [1998a] showed how the classical maximum flow algorithms can be adapted for (3) and suggested that the Dinic algorithm [1970] represents the simplest way to find an algorithm for our problem. The algorithm is given by the following procedure and the complexity is $O(m\cdot n\cdot (m+n)^2)$ [1998a].

Procedure *Dinic*(n,m,x,y,c,f)

begin

for i:=1 **to** n **do for** j:=1 **to** m **do** $f_{i,j}$:=0;

for i=1 **to** m+n-1 **do**

Repeat

Search for an augmenting path D by the length i by using a *bfs* search.

Compute the residual capacity ε of the path D.

Augment the flow f by ε along D.

until there is not a flow augmenting path by the length i.

end;

Above, a flow-augmenting path for the flow f in the bipartite network R is a succession of nodes $D=(s_{i_1},t_{j_1},s_{i_2},t_{j_2},...,s_{i_p},t_{j_p})$ in the following properties:

$$f_{i_k,j_k} < c_{i_k,j_k}\ (\forall k=1,p) \text{ and } 0 < f_{i_{k+1},j_k}\ (\forall k=1,p-1) \tag{4a}$$

$$\sum_{j=1}^{m} f_{i_1,j} < x[i_1] \tag{4b}$$

$$\sum_{i=1}^{n} f_{i,j_p} < y[j_p] \tag{4c}$$

In spite of simplicity of (3), many practical applications can be modeled by using it. Ahuja *et.al.* [1993] gave a complete list of theoretical and practical applications of the maximum flow problem. Tabirca [1998] found more than 20 of them that can be reduced to (3). The main of them are: the tournament problem, the baseball elimination problem, the freight handling terminals problem, the network reliability testing problem, the flyaway kit problem, the optimal destruction of military targets, the building Evacuation problem, *etc*.

2. The Budget Problems

In this section, a budget policy problem is presented as an application of the maximum flow problem in a bipartite graph. This budget problem analyses an economical system from the financial point of view studying three elements: budget sources, budget destinations, budget strategy.

Let $S=\{s_1, s_2, \ldots, s_n\}$ be the budget sources. For each source s_i, an incoming budget equal to $a[i]$ is estimated. Let $T=\{t_1, t_2, \ldots, t_m\}$ be the budget destinations. The budget resource required by the destination t_i is estimated at $b[i]$. The budget policy distributes the money from sources to destinations. Let us suppose that the distribution of money for the source s_i is given by $c_{i,1}, c_{i,2}, \ldots, c_{i,m}$, where $c_{i,j}$ represents the amount of money allocated from s_i to t_j. The following equations should be satisfied for an ideal budget.

$$\sum_{j=1}^{m} c_{i,j} = a[i] \left(\forall i = 1,2,\ldots,n\right) \tag{5a}$$

$$\sum_{i=1}^{n} c_{i,j} = b[j] \left(\forall j = 1,2,\ldots,m\right) \tag{5b}$$

These equations establish a financial equilibrium for the system so that the total amount of money provided by sources equals to the total amount of money required by destinations $\sum_{i=1}^{n} a[i] = \sum_{j=1}^{m} b[j]$. Obviously, this ideal situation seldom occurs in reality and for various reasons such as inflation, delay for payment or unexpected expenses, the amount of money of sources or destinations is different that the expected amount. Let $x[i]$ be the real amount of money for source s_i and $y[j]$ be real amount of money for destination t_j.

The budget problem is defined by $(a,b,c) \in R^n \times R^m \times M_{n,m}(R)$ giving the ideal budget and $(x,y) \in R^n \times R^m$ giving the real source and destination arrays. The budget policy problem gives a distribution of money such that:

- The maximum amount of money that goes from s_i is $x[i]$. (6a)
- The maximum amount of money that goes from t_j is $y[j]$. (6b)
- From the source s_i goes to the destination t_j at most $c_{i,j}$ amount of money. (6c)

Let $f = \left(f_{i,j}\right)_{i=1,n;\, j=1,m}$ be a matrix with the property $f_{i,j}$ represents the amount of money from the source s_i to the destination t_j. The budget problem should determine a maximal distribution of money from the sources to the sinks. Let us call it maximal budget. Obviously, the mathematical model of this problem is given by (3). Moreover, a solution of this budget problem can be found using an algorithm for the maximum flow problem in a bipartite network.

3. Algorithm for Constructing the Budget

In this section, a method for constructing an efficient budget is presented. Let us suppose that we want to find a budget for the budget sources $S=\{s_1, s_2, \ldots, s_n\}$ and the budget destinations $T=\{t_1, t_2, \ldots, t_m\}$. Firstly, it is started from an initial budget $(a,b,c) \in R^n \times R^m \times M_{n,m}(R)$. This budget can be simulated for several real amounts $(x,y) \in R^n \times R^m$. For the problem $(a,b,c) \in R^n \times R^m \times M_{n,m}(R)$ and $(x,y) \in R^n \times R^m$, the maximal budget $f^* = \left(f^*_{i,j}\right)_{i=1,n;\, j=1,m}$ is found using for example the Dinic algorithm. Some interesting remarks about the initial budget can be made from f^*. The fortunate cases are:

1. $\sum_{j=1}^{m} f^*_{i,j} = x[i] \Rightarrow$ the real amount of money from s_i is completely distributed to destinations.
2. $\sum_{i=1}^{n} f^*_{i,j} = y[j] \Rightarrow$ the real amount of money to t_j is completely covered.

There are also some unfortunate situations:

3. $\sum_{j=1}^{m} f_{i,j}^{*} < x[i] \Rightarrow$ the real amount of money $x[i]-\sum_{j=1}^{m} f_{i,j}^{*}$ is left to be distributed.

4. $\sum_{i=1}^{n} f_{i,j}^{*} < y[j] \Rightarrow$ the destination t_j has a real amount of $y[j]-\sum_{i=1}^{n} f_{i,j}^{*}$ that is not covered using all the sources.

The situations 3-4 should be analysed from financial point of view to find solutions, which can correct them. When the source s_i satisfies the condition 3, then the real amount $x[i]$ is not properly distributed. In this case, the values $\left(c_{i,j},\ j=\overline{1,m}\right)$ are not efficiently chosen therefore, they should be redefined. The worst case is when the destination t_j satisfies the condition (4). This case can be corrected both redefining the values $\left(c_{i,j},\ j=\overline{1,m}\right)$ and reducing the amount $y[j]$. Therefore, all of these corrections increase or decrease the values *c, x, y* finding a new budget problem $(a', b', c') \in R^n \times R^m \times M_{n,m}(R)$ and $(x', y') \in R^n \times R^m$. This new problem can be also solved and improved. This scheme can be repeated until a correct budget is found. The procedure *budget* describes the above scheme.

```
procedure budget(n,m,f);
begin
        construct an initial problem (c,x,y);
        repeat
                call Dinic(n,m,x,y,c,f);
                analyse f ;
                if the budget is OK then return;
                define the new elements (c',x',y');
                (c,x,y):=(c',x',y');
        until false;
end;
```

The complexity of the procedure *budget* depends on the number of repetitions. In each repetition, the budget *f* is found applying the Dinic algorithm and the complexity is $O(m \cdot n \cdot (m+n)^2)$. If the number of the repetitions is *nr*, then $O(nr \cdot m \cdot n \cdot (m+n)^2)$ gives the final complexity.

4. The Budget Exercise Problem

The purpose of this section is to introduce the budget exercise problem. The starting point is given by the simple remark that a budget is usually studied in time. Therefore, the static analysis for the above budget problem [Tabirca, 1999] should be extended considering the time as a parameter.

Let us suppose that the time is represented by $Time = \{0,1,...,P\}$. Let us also suppose that the budget for the period [*t*, *t*+1) is $(a_t, b_t, c_t) \in R^n \times R^m \times M_{n,m}(R)$ and is well-balanced. Therefore, the budget for whole period is defined by $\left((a_t, b_t, c_t) \in R^n \times R^m \times M_{n,m}(R)\right)_{t=\overline{0,P-1}}$.

As in the budget problem [Tabirca, 1999], the real incoming and required amounts may be different to the budget amounts. Suppose that the real incoming amounts are given by $\left(x_t \in R^n\right)_{t=\overline{o,P-1}}$, where $x_t[i]$ is the real incoming amount for the source s_i at the time *t*. The real required amounts are given by $\left(y_t \in R^m\right)_{t=\overline{1,P}}$, where $y_t[i]$ is the real required amount for the destination t_i at the time *t*.

The budget exercise problem studies in time a distribution of money from the sources $S=\{s_1, s_2, \ldots, s_n\}$ to the destinations $T=\{t_1, t_2, \ldots, t_m\}$ such that :

- The budget is represented by $\left((a_t, b_t, c_t) \in R^n \times R^m \times M_{n,m}(R)\right)_{t=\overline{0,P-1}}$.
- The real incoming amounts are given by $\left(x_t \in R^n\right)_{t=\overline{o,P-1}}$.

- The real required amounts are given by $\left(y_t \in R^m\right)_{t=\overline{1,P}}$.
- At a time *t*, a part of a real incoming amount for the source s_i may be transferred to the same source for the further times.

An incorrect solution for this problem can be obtained by finding the maximal budget for each period $[t,t+1)$ based on the information given by (a_t, b_t, c_t), x_t, y_t. In this case, the solution is maximal for each period $[t,t+1)$ but, it may not be maximal for the whole period $[0,P]$. A correct model for this problem based on the dynamic bipartite network is proposed in the following. The dynamic bipartite network is a special network for which the capacities, the source and sink arrays are dependent on time. This network is defined in the following.

Let $G=(S\cup T, E)$ be a bipartite graph and $Time=\{0,1,...,P\}$ be a sequence of times.

Definition 3.

The dynamic bipartite network is

$$\left(R_t=\left(G=(S\cup T,E),x_t,y_t,c_t\right)\right)_{t\in\overline{0.P}} \tag{7}$$

where

- $x_t \in R^{|S|}$ is the source array and $x_t[i]=x(i;t)$ represents the amount of the source s_i at the moment *t*.
- $y_t \in R^{|T|}$ is the sink array and $y_t[i]=y(i;t)$ represents the amount of the sink t_i at the moment *t*.
- $c_t \in M_{|S|\times|T|}(R)$ is the capacity function and $c_t[i,j]=c(i,j;t)$ represents the capacity of the arc *(i,j)* at the moment *t*.

For this network, the maximum dynamic flow problem is defined as follows:

$$\max \sum_{t=0}^{p-1}\sum_{i=1}^{n}\sum_{j=1}^{m} f(i,j;t)$$

subject to:

$$\sum_{j=1}^{m} f(i,j;t) \le x(i;t)\left(\forall i=\overline{1,n}\right)\left(\forall t=\overline{0,P-1}\right) \tag{8a}$$

$$\sum_{i=1}^{n} f(i,j;t-1) \le y(j;t)\left(\forall j=\overline{1,m}\right)\left(\forall t=\overline{1,P}\right) \tag{8b}$$

$$0\le f(i,j;t)\le c(i,j;t)\left(\forall i=\overline{1,n}\right)\left(\forall j=\overline{1,m}\right)\left(\forall t=\overline{0,P}\right) \tag{8c}$$

If *f* satisfies the equations (8a - 8c), then *f* is called the dynamic flow for the bipartite network. Equation (8a) reflects the fact that the outgoing amount of money $\sum_{j=1}^{m} f(i,j;t)$ does not exceed $x(i;t)$ at the moment *t*. Equation (8b) gives that the incoming amount of money $\sum_{i=1}^{n} f(i,j;t-1)$ is less than $y(j;t)$ at the moment *t*.

4.1. An Algorithm for the Budget Exercise Problem

Problem (4) is the right mathematical model for the *Budget Exercise Problem.* This problem is very difficult to be solved with classical methods because the number of equations is quite large. Therefore, special methods are required to find its solution. In the following, an algorithm for the *Budget Exercise Problem* is developed. This algorithm is obtained by transforming in two steps the dynamic bipartite network into a static network. The maximal dynamic flow for Problem (4) is equivalent to a maximal flow in a bipartite network.

Step 1. The Extended Dynamic Network.

The extended dynamic network is constructed at this step. This network is an intermediate between the dynamic bipartite network and the static network, which will be introduced at Step 2.

Let $ss, tt \notin S\cup T$ be two nodes.

A new graph *GG=(S∪T∪{ss, tt}, {ss}×S∪E∪T×{tt})* is constructed by connecting the new node *ss* with all the nodes of *S* and all the nodes of *T* with the node *tt*. The dynamic network with the graph support *GG* is

$$\left(RR_t = \left(GG = (V(GG), E(GG)), cc_t\right)\right)_{t\in\overline{0,P}}, \tag{9}$$

where the capacity function *cc* is defined by:

$$cc(ss,i;t) = x(i;t)\left(\forall i = \overline{1,n}\right). \tag{10a}$$

$$cc(j,tt;t) = y(j;t)\left(\forall j = \overline{1,m}\right). \tag{10b}$$

$$cc(i,j;t) = c(i,j;t)\left(\forall i = \overline{1,n}\right)\left(\forall j = \overline{1,m}\right). \tag{10c}$$

Problem (8) is rewritten based on the network (9) as follows:

$$\max \sum_{t=0}^{p-1}\sum_{i=1}^{n} f(ss,i;t)$$

subject to:

$$\sum_{j=1}^{m} f(i,j;t) = f(ss,i;t-1)\left(\forall i = \overline{1,n}\right)\left(\forall t = \overline{0,P-1}\right) \tag{11a}$$

$$\sum_{i=1}^{n} f(i,j;t-1) = f(j,tt;t)\left(\forall j = \overline{1,m}\right)\left(\forall t = \overline{1,P}\right) \tag{11b}$$

$$0 \le f(i,j;t) \le cc(i,j;t)\left(\forall i = \overline{1,n}\right)\left(\forall j = \overline{1,m}\right)\left(\forall t = \overline{0,P}\right) \tag{11c}$$

$$0 \le f(ss,i;t) \le cc(ss,i;t)\left(\forall i = \overline{1,n}\right)\left(\forall t = \overline{0,P}\right) \tag{11d}$$

$$0 \le f(j,tt;t) \le cc(j,tt;t)\left(\forall j = \overline{1,m}\right)\left(\forall t = \overline{0,P}\right) \tag{11e}$$

Step 2. The Static Network

A static network associated with the dynamic network $\left(RR_t = \left(GG = (V(GG), E(GG)), cc_t\right)\right)_{t\in\overline{0,P}}$ is constructed at this step. This construction follows an idea introduced by Ford and Fulkerson [1962]. *P*+1 new nodes are introduced for each node. If *x* is a node, then these new nodes are denoted by *x(t)* , *t*=0, 1,..., *P*.

The static network is $\overline{R} = \left(\overline{G} = (\overline{V}, \overline{E}), ss(0), tt(P), \overline{c}\right)$ and is defined as follows:

$$\overline{V} = \bigcup_{t=0}^{P}\{ss(t), tt(t)\} \cup \bigcup_{t=0}^{P}\bigcup_{i=1}^{n}\{s_i(t)\} \cup \bigcup_{t=0}^{P}\bigcup_{i=1}^{m}\{t_i(t)\} \text{ is the set of nodes} \tag{12a}$$

$$\overline{E} = \bigcup_{t=0}^{P-1}\bigcup_{x\in V(GG)}\{(x(t), x(t+1))\} \cup \bigcup_{t=0}^{P-1}\bigcup_{(x,y)\in E(GG)}\{(x(t), y(t+1))\} \text{ is the set of arcs.} \tag{12b}$$

The capacity function $\overline{c}$ is defined by:

$$\overline{c}(x(t), x(t+1)) = \infty\left(\forall x \in V(GG)\right). \tag{12c}$$

$$\overline{c}(x(t), y(t+1)) = c(x,y;t)\left(\forall (x,y) \in E(GG)\right). \tag{12d}$$

Obviously, this static network has $n_s = (P+1)\cdot(n+m+2) = O\left(P\cdot(n+m)\right)$ nodes and at most $m_s = P\cdot(n\cdot m + n + m + 2) = O(P\cdot n\cdot m)$ arcs. For this final network, the maximum flow $ff : \overline{E} \to R_+$ can be found by applying one of the efficient maximum flow algorithms [Ahuja, 1993]. For example, if the improved FIFO algorithm [Ahuja, 1993] is applied then, the complexity is

$$O\left(n_s \cdot m_s \cdot \log\frac{n_s^{\,2}}{m_s}\right) = O\left(P^2\cdot(m+n)\cdot n\cdot m\cdot\log P\cdot\frac{(n+m)^2}{n\cdot m}\right).$$

Ford and Fulkerson [1962] proved that the maximal dynamic flow in the dynamic network $\left(RR_t = \left(GG = (V(GG), E(GG)), cc_t\right)\right)_{t\in\overline{0,P}}$ is equivalent to the maximal flow in the network

$\overline{R} = \left(\overline{G} = (\overline{V}, \overline{E}), ss(0), tt(P), \overline{c}\right)$. Therefore, the maximal dynamic flow can be easier found from the maximal flow *ff* using

$$f(i, j; t) = ff(s_i(t), t_j(t+1)) \left(\forall i = \overline{1,n}\right)\left(\forall j = \overline{1,m}\right)\left(\forall t = \overline{0, P-1}\right). \tag{13}$$

In conclusion, the budget exercise problem which is given by $\left((a_t, b_t, c_t) \in R^n \times R^m \times M_{n,m}(R)\right)_{t=\overline{0,P-1}}$, $\left(x_t \in R^n\right)_{t=\overline{o,P-1}}$ and $\left(y_t \in R^m\right)_{t=\overline{1,P}}$ is solved by the following steps:

1. Construct the dynamic network $\left(R_t = \left(G = (S \cup T, E), x_t, y_t, c_t\right)\right)_{t \in \{0,1,\ldots,P\}}$.
2. Construct the static network $\overline{R} = \left(\overline{G} = (\overline{V}, \overline{E}), ss(0), tt(P), \overline{c}\right)$.
3. Find a maximal flow in the network $\overline{R} = \left(\overline{G} = (\overline{V}, \overline{E}), ss(0), tt(P), \overline{c}\right)$.
4. Find the solution of the budget exercise problem applying (13).

Because both networks are constructed in at most $O(P \cdot n \cdot m)$, we found that the complexity of the above solution is given only by the maximum flow computation. Therefore, an efficient solution can be found in $O\left(P^2 \cdot (m+n) \cdot n \cdot m \cdot \log P \cdot \frac{(n+m)^2}{n \cdot m}\right)$.

5. Conclusions

Although budget problems have a vital importance in practice and can arise in many economical systems, they have not been sufficiently studied. This article comes to study a particular budget proposing two mathematical models for the static and dynamic cases. There are a few real budget problems that can be studied with them and the most important is the local Romanian budget.

Another interesting remark concerns the complexity of the algorithms. The maximum flow problem in a bipartite network can be also solved using special algorithms. The complexity can be reduced from $O(m \cdot n \cdot (m+n)^2)$ to $O(m \cdot n \cdot (m+n) \cdot \log(n+m))$ for the *dynamic tree algorithm* [Ahuja *et. al*., 1993]. Therefore, we can find versions of the algorithm *budget* with the efficient complexity $O\left(nr \cdot m \cdot n \cdot (m+n) \cdot \log(m+n)\right)$.

References

Ahuja, R.K., Magnanti, T.L. and Orlin, J.B. (1993) *Network Flow:Theory, Algorithm and Applications*, Prentice-Hall, New Jersey.

Dinic, E. A. (1970) Algorithm for Solution of a Problem of Maximal Flow in Networks with Power Estimation, *Soviet Mathematics Doklady*, **11**, 1277-1280.

Ford, L. R. and Fulkerson, D. R. (1956) Maximal Flow Through a Network, *Canadian Journal of Mathematics*, **8**, 399-404.

Ford, L.R. and Fulkerson, D.R. (1962) *Flow in Networks*, Princeton University Press, Princeton, NJ.

Tabirca, S. (1998) *Parallel and Sequential Aspects of the Maximum Flow Problem*, PhD Thesis, Brunel University, UK.

Tabirca, S., Tabirca, T. (1998a) The Maximal Flow Problem for Bipartite Networks, *Bulletin of the "Transilvania" University of Brasov*, **Vol. 5**.(40), 43-49.

Tabirca, S., Tabirca, T. (1999) Mathematical Model for a Budget Problem, *The Proceedings of the Fourth International symposium on Economic Informatics*, Bucharest, 443-449.