Proceedings of
the XXVII Summer School, Sozopol'01

Applications of Mathematics in Engineering and Economics

edited by
D. Ivanchev and M.D. Todorov

FACULTY OF
APPLIED MATHEMATICS & INFORMATICS
TECHNICAL UNIVERSITY OF SOFIA

HERON PRESS · Sofia · 2002

Applications of Mathematics in Engineering and Economics'27
eds. D. Ivanchev and M.D. Todorov, Heron Press, Sofia, 2002

Quickest path problem

L. A. Pilipchuk[1], Y. H. Pesheva[2]

[1] Faculty of Applied Mathematics and Computer Science,
Belorussian State University, F.Skarina Avenue 220050, Minsk, Belarus
[2]Technical University of Sofia, P.O.Box 384, 1000 Sofia, Bulgaria

Let us consider the quickest path of a given length k (k-flow) problem. It allows us to determine the quickest path among all paths that contain k arcs. For the first time the quickest path problem was mentioned in [1]. The mathematical model of the quickest path of a given length problem is the following:

$$\sum_{(i,j)\in U}\left(c_{ij}+\delta_{ij}\frac{G}{d_{ij}}\right)x_{ij}\to\min,$$

$$\sum_{j\in I_i^+(U)}x_{ij}-\sum_{j\in I_i^-(U)}x_{ji}=\begin{cases}1, & i=s,\\ -1, & i=t,\\ 0, & i\in I\setminus\{s,t\}.\end{cases}$$

$$\delta_{ij}=\begin{cases}1, & (i,j)=(i_*,j_*), \quad \text{if} \quad d_{i_*j_*}=\min_{x_{ij}\neq 0}\{d_{ij},(i,j)\in U\},\\ 0, & (i,j)\in U\setminus(i_*,j_*).\end{cases}$$

$$x_{ij}\in\{0,1\},\ (i,j)\in U,$$

$$\sum_{(i,j)\in M}x_{ij}=k,\ M\subseteq U,$$

$$I_i^+(U)=\{j:(i,j)\in U\},\qquad I_i^-(U)=\{j:(j,i)\in U\}$$

where $S=\{I,U\}$ is a connected directed net without parallel arcs and loops with a set of nodes I and a set of arcs U; d_{ij} is the capacity of the arc (i,j); x_{ij} is the flow of the arc (i,j); $c_{ij}\geq 0$ is the time needed to pass the arc $(i,j)\in U$, passing time; $c_{ij}\geq 0$, s is the source, t is the destination, M is the set of arc of the path from s to t without repeated nodes. Capacity d_{ij} of the arc (i,j) stands for the maximum number of data units, transferred from the node i to node j within a time unit. The time c_{ij} of the arc (i,j) traversal stands for the time consumed while sending G data units from the node i to the node j through the arc (i,j).

While solving the problem of the quickest path of a given length k from node s to node t the aim is to get a set of the prevalent paths for node t.

Recall [2] that the path L_i from the node s to the node t with capacity equal to c_i is prevalent over the path L_j with the capacity equal to c_j if the following relations hold true:

$$T_i < T_j,\ c_i \geq c_j,\ \text{or}\ T_i \leq T_j,\ c_i > c_j,$$

$$T_i = \sum_{(i,j)\in L_i} c_{ij},\ T_j = \sum_{(i,j)\in L_j} c_{ij},$$

T_i is the time of the path L_i traversal and T_j is the time of the path L_j traversal. Then the time of the flow G transference through the path L_p is equal to

$$T_p + \frac{G}{c_p},\ c_p = min_{(i,j)\in L_p} c_{ij},\ p = i, j.$$

Let Q_t stand for the set of prevalent paths for node t. This set has a lot of characteristics [2].

Theorem 1 *For any paths $L_i, L_j \in Q_t$ the following inequalities hold:*

$$T_i < T_j,\ c_i < c_j,\ or\ T_i > T_j,\ c_i > c_j. \tag{1}$$

Proof. Let $T_i > T_j$, and $c_i < c_j$. The equality is possible in only one of these inequalities. Then the path L_i is prevalent over L_j. This contradicts to the fact that the path L_j is prevalent and belongs to the set of prevalent paths. The case $T_i < T_j$ and $c_i > c_j$ can be proved in the same way. The theorem is proved.

The set of prevalent paths can be ordered with the passing time and the capacity ascending.

Theorem 2 *Let $L_t \in Q_t$ be some prevalent path, which passes through node $i \in I$. Then the subpath L_i (part of the path from the source to node i) is a prevalent path for this node i, i.e., $L_i \in Q_i$.*

Proof. Let there be a path $L_j \in Q_i$, which is prevalent over the path $L_i \in Q_i$. Then the passing time and the capacity of the paths L_j satisfy the relations (1). Let us substitute the subpath L_i of the path L_t with the prevalent path L_j. According to (1), the passing time of the obtained path L_i is less than the passing time of the path L_t with the same or greater capacity or with the greater capacity and the same passing time. But this contradicts to the statement that the path is a prevalent path for the node t. The theorem is proved.

So, for any paths $L_i, L_j \in Q_t$ the following inequalities hold true:

$$T_i < T_j,\ c_i < c_j,\ \text{or}\ T_i > T_j,\ c_i > c_j.$$

In one of the inequalities of each group the equality is possible. Hence, the set of the prevalent paths can be arranged in the order, in which the traversal time and the capacity are ascending.

If $L_t \in Q_t$ is a prevalent path for the node t, which passes through node $i \in I$, then the subpath L_i (part of the path L_t from the source to node i) is a prevalent path $L_i \in Q_i$ for this node.

Though not every path from the set of prevalent paths is a solution with some G. The alternative is possible when the prevalent path is not the quickest one no matter what the transferred data quantity is. The set of prevalent paths has certain characteristics that allow to derive conditions of the completion of the algorithm "in advance" while solving the quickest path problem with the transferred data quantity G given a priori.

Let G be given and a certain path be determined. The cost function for this path with this G is F_G. If the time of another path traversal is greater than F_G, that path cannot be the quickest one no matter what the capacity is.

When using the algorithm, that extracts prevalent paths with traversal time ascending we can stop the algorithm exactly when the traversal time of the new path exceeds the best cost function value for all already known paths.

The same can be derived for the case of the prevalent path extraction with the capacity descending. Let G be given and a certain prevalent path be determined. Let the cost function value for this path with this G is F_G. Then if the capacity of some other path is less or equal to $\frac{G}{F_G}$, then this path cannot be the quickest one no matter what the traversal time is. When using the algorithm, that extracts prevalent paths with the capacity descending, we can stop the algorithm exactly when the newly derived path capacity is greater or equal to $\frac{G}{F_G}$.

Let d_i be the already known minimum time of the path traversal from s to i. Let $d_i = \infty, i \in I \setminus \{s\}$ and $d_s = 0$. Let us order the arcs of the net with their capacities descending, and add them to the net just in the same order. In the process of adding the arc (i, j) with capacity d_{ij} and traversal time c_{ij} the following steps should be carried out:

1. If $d_i \neq \infty$ and $d_i + c_{ij} \leq d_j$, that is if we have found a path to j with a lower time than known before, then we should recalculate graph's labels starting with node j. Dijkstra algorithm with the priority queue [3] or basic algorithm modification can be used for this recalculation.

2. If d_t has been changed after recalculation, we add a new path to the list of the prevalent paths of node t. The capacity of this path is equal to d_{ij} and traversal time is equal to d_t.

So, while adding the appropriate arc in the descending order, it is checked for each capacity, whether a path from s to t with time traversal which is less than already known ones exists. If it exists and its time traversal is less than the time traversal of the already derived path, then there are no prevalent paths over this newly derived one and it is added to the set of the prevalent paths. Otherwise the already known paths are prevalent over the newly derived one. So, this algorithm extracts the whole set of prevalent paths for the given node.

During the realization of the algorithm which finds the shortest paths in the formed graph of the connected components there is an opportunity to modify the algorithm in order to consider the data that has been already obtained at the previous steps of the algorithm. According to Dijkstra's algorithm which uses heaps it can be realized in the following way: when recalculating the marks d_t we should take into consideration the marks that were obtained at the previous algorithm steps. It means to store the mark values, while applying Dijkstra's algorithm to the already known ones. So, it is necessary to solve the quickest path problem from node s to node t on the net where the traversal time stands for the arc length.

It is obvious that the path with the length k which minimizes this value is the quickest path of the given length in the initial net.

Theorem 3 *The number of operations of the algorithm where capacities stand in the descending order is equal to* $O(m^2 + mn \log n)$ *where* n *stands for the number of nodes, and* m *stands for the number of arcs of the initial net* S.

Proof. The maximum number of algorithm steps is equal to the maximum number of arcs (equal to m) in the net. In the worst case at each step of the algorithm the shortest path problem is being solved. The process of solving the shortest path problem takes time which is equal to $O(m + n \log n)$ [3]. So, the total time for each step of the algorithm is equal to $O(m + n \log n)$, and the total time for the algorithm is equal to $O(m^2 + mn \log n)$. The theorem is proved.

This modification has a lot of advantages. It solves the problem on a small net gradually extending, it and in case of the problem with a given quantity of the transferred data allows us to find the solution without analyzing the whole net. One of the conditions, that stops the algorithm is the case when capacity of the next added arc is less than G/F_G.

With the help of this algorithm one can solve the problem of the quickest path to all net nodes or to some subset of the nodes. In order to do this one should check if the marks of nodes of the given set have changed after the recalculation of the marks. This algorithm reduces the number of operations while finding the prevalent path for the next capacity because it effectively uses the information about the net that has been obtained from the previous steps.

Let us motivate this algorithm. Because the traversal time of the path is the key element in the heap in our case. The paths extracted from the heap, have the

minimal time, because the passing time of the path is the key element in the heap in our case. Such path extraction for some node i (its capacity is greater than d_i) means that there are no prevalent paths to the given node over our one. If its capacity is less than d_i than the prevalent path exists. If the path is prevalent for the given node then all possible continuations should be considered, except for the ones with the already known prevalent paths. Thus, at the moment the the algorithm stops, each node will contain the list of prevalent paths to it from node s.

Let us show the work of the algorithm on the example for the following net (Figure 1):

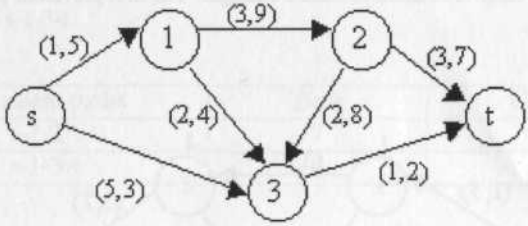

Figure 1.

Capacity	9	8	7	5	4	3	2
Time	3	2	3	1	2	5	1
Arc	(1,2)	(2,3)	(2,t)	(s,1)	(1,3)	(s,3)	(3,t)

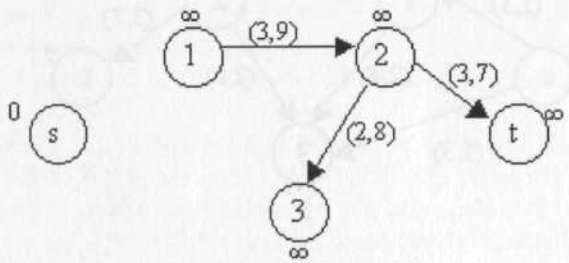

Figure 2. Add arcs (1,2), (2,3), (2,t).

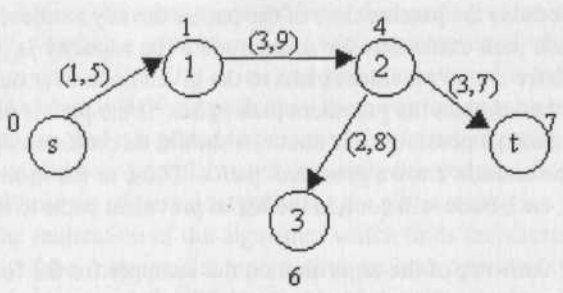

Figure 3. Add arc (s,1). The mark of node t changes. The first prevalent path of the given length 3 is s-1-2-t.

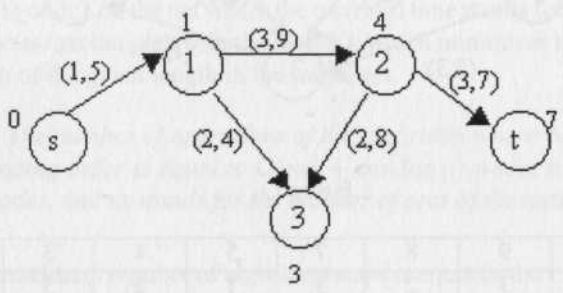

Figure 4. Add arc (1,3).

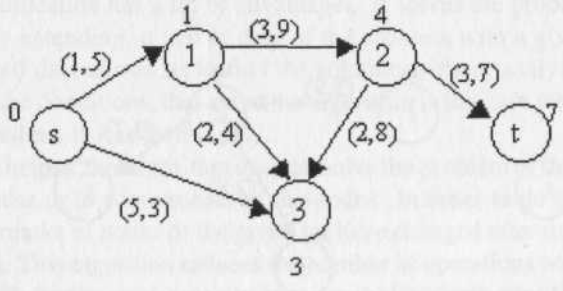

Figure 5. Add arc (s,3).

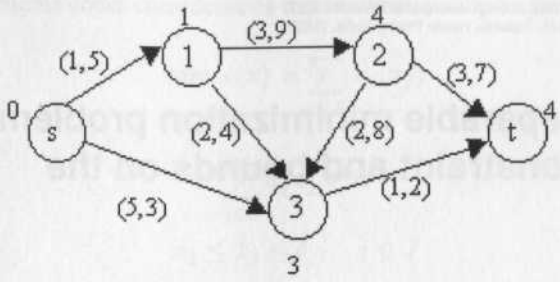

Figure 6. Add arc (3,t). The mark of node t changes. The second prevalent path of the given length 3 is s-1-3-t.

Prevalent paths	Time	Capacity
s-1-2-t	7	5
s-1-3-t	4	2

References

[1] Chen, Y.L., Chen, Y.H. (1992) The quickest path problem, *Computers and Operations Research.*

[2] Pilipchuk L.A., Laput I.V. (2000) The quickest path problem, *International science conference "Computer Network Technologies". October, 25-29, Minsk.*

[3] T.Kormen, C. Leiserson, R.Rivest (1999) *Algorithms: Construction and Analysis*, MCNMO, M.