EGERVÁRY RESEARCH GROUP
ON COMBINATORIAL OPTIMIZATION

TECHNICAL REPORTS

TR-2005-10. Published by the Egerváry Research Group, Pázmány P. sétány 1/C, H–1117, Budapest, Hungary. Web site: `www.cs.elte.hu/egres`. ISSN 1587–4451.

An $O(n^2)$ algorithm for ring routing

Zoltán Király

October 2, 2005

An $O(n^2)$ algorithm for ring routing

Zoltán Király[⋆]

Abstract

The purpose of this note is to effectively solve routing (multicommodity flow) problems specialized to ring networks (circles). This note describes an $O(n^2)$ algorithm for solving ring routing. The solution is always half-integer (if the input is integer), and it is integer if the problem is Eulerian. We show a simple modification that solves the integral routing problem also for the non-Eulerian case.

A lot of nice algorithms are known for similar, but more general problems, see e.g. [5],[3]. However not known specialization of these algorithms to rings runs in $O(n^2)$. Instead we start with an algorithm of Schrijver, Seymour and Winkler [4] developed for rings. They claimed $O(n^4)$ running time. Here we show that with some modification of their algorithm and with a careful implementation the running time can be decreased to $O(n^2)$.

1 Introduction, Notation

The undirected ring is defined by $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_n\}$, and $e_i = v_iv_{i+1}$. In the terminology we assume that the numbering of the vertices is counter-clockwise. In the indices addition and subtraction is meant modulo n, so $v_1 = v_{n+1}$, and we use intervals $[i, j]$ that denotes $i, i+1, \ldots, j$. We also use half-open and open intervals.

The non-negative *integral* capacity of edge e_i is given by $g(e_i) = g_i$ and the non-negative symmetric *integral* demands are given by $h(v_i, v_j) = h(v_j, v_i) = h_{ij}$, the amount must be routed between vertices v_i and v_j. (Throughout we assume $h_{ii} = 0$.) Let $h_i = \sum_{j=1}^{n} h_{ij}$.

Let P_{ij} denote the path going to the right from v_i to v_j, i.e. $v_i, v_{i+1}, \ldots, v_j$. The expected solution is a routing of the demands, namely assigning non-negative values r_{ij} with the following properties.

$$r_{ij} + r_{ji} = h_{ij} \ \ \forall\, 1 \le i, j \le n,$$

$$\sum_{[i,j)\ni k} r_{ij} \le g_k \ \ \forall\, 1 \le k \le n.$$

[⋆]Department of Computer Science and Communication Networks Laboratory, Eötvös University, Pázmány Péter sétány 1/C, Budapest, Hungary. Research supported by EGRES group (MTA-ELTE) and OTKA grants T 037547 and T046234.

In other words, r_{ij} amount of demand h_{ij} is routed to the right from v_i, i.e. along path P_{ij}, and the remaining $r_{ji} = h_{ij} - r_{ij}$ amount is routed to the left from v_i, i.e. along path P_{ji}. Necessarily the total routed amount through any edge must not exceed its capacity.

We call a routing integral, if every value r_{ij} is an integer, half-integral, if they are halves of an integer and fractional otherwise. The problem is called Eulerian, if $g_{i-1} + g_i + h_i$ is even for all i. For a simple cut determined by edges e_i and e_j, let $L_h(i, j)$ denote the total demand through this cut, i.e. the sum of h_{kl} where $k \in (i, j]$ and $l \in (j, i]$. The following condition, called *cut-condition* (shortly c-c) is clearly necessary for having even a fractional routing:

$$L_h(i, j) \le g_i + g_j \ \ \forall\, 1 \le i \ne j \le n.$$

The next theorem states that it is sufficient (this part follows from Farkas' lemma), moreover, if the problem is Eulerian then c-c is enough for having an integer solution.

Theorem 1.1 (Okamura and Seymour [2]). *The ring routing problem has*

(i) a fractional solution iff the cut-condition holds,

(ii) a half-integer solution iff it has a fractional solution,

(iii) an integer solution, if the problem is Eulerian and the cut-condition holds.

We call a cut $\{e_i, e_j\}$ tight, if the c-c is satisfied (for i, j) with equality, and call a pair of edges, e_i and e_j a tight pair if they determine a tight cut. Observe that the third part of the theorem is not an "iff" statement, the necessary and sufficient condition for having an integral routing was given by Frank. Let us call cuts $\{e_i, e_j\}$ and $\{e_k, e_l\}$ *crossing*, if $k \in (i, j)$ and $l \in (j, i)$. Furthermore a crossing pair of cuts is called odd, if $L_h(i, k) + g_i + g_k$ is an odd number.

Theorem 1.2 (Frank [1]). *For a ring routing problem suppose that the c-c is satisfied. There is an integral routing iff no odd crossing pair of tight cuts exists.*

We call a demand h_{ij} *uniquely routable*, if one of P_{ij} and P_{ji} contains both determining edges of a tight cut (in this case we say that the path *double-crosses* the tight cut). If wlog. P_{ji} double-crosses a tight cut then the phrase "we route the uniquely routable demand h_{ij}" means that we set $r_{ij} = h_{ij}$, $r_{ji} = 0$, set $h_{ij} = 0$ and decrease every g_k by h_{ij} for every $k \in [i, j)$.

Corollary 1.3 (of Theorem 1.1). *If the c-c was valid before this procedure then it remains valid after it, as well. Moreover no g_k is decreased to a negative number, and every tight cut remains tight.*

Proof. Let the tight cut double-crossed by P_{ji} is determined by e_k and e_l. If the c-c was valid before this procedure then, by Theorem 1.1, a routing exists. If in this routing $r_{ji} > 0$ then after routing only demand h_{ij}, the values g_k and g_l are decreased by $r_{ji} > 0$ but $L_h(k, l)$ is not changed, resulting that the cut $\{e_k, e_l\}$ violates the c-c.

However in this case the other demands cannot be routed, a contradiction. If in the routing given by the theorem $r_{ji} = 0$ then after routing this demand alone, clearly a routing exists for the remaining demands as well (and the capacities remain non-negative). A tight cut was either double-crossed by P_{ji} when neither the capacities nor the L_h value was changed, or simply-crossed by P_{ij} when both side was decreased by h_{ij}. $\square$

We modify the algorithm of Schrijver, Seymour and Winkler [4]. Our algorithm also consists of four phases. They claim running time of $O(n^4)$, we guarantee running time of $O(n^2)$. The main differences are the following. They route uniquely routable demands one-by-one, while we first calculate which are the uniquely routable demands and then route them simultaneously. Instead of their fourth phase (which is not exactly defined in their paper and the form presented it is faulty) we calculate a simple explicit routing. And we implement all phases carefully to achieve $O(n^2)$ total running time. (We should mention that the purpose of their paper was not to solve this problem, but to give an approximate solution for the unsplittable routing problem, where for each i, j one of r_{ij} and r_{ji} must be zero.)

2 Algorithm

The algorithm consists of four phases:

I Calculate tight cuts and ensure that every edge belongs to a tight cut.

II For every vertex determine the “opposite vertex”, which will have the following property: every demand from the actual vertex to a vertex left to the opposite vertex must be routed to the left and every demand from the actual vertex to a vertex right to the opposite vertex must be routed to the right.

III Route each demand that is uniquely routable, that is go from one vertex to a vertex that is not the opposite one.

IV For the remaining demands calculate locally how big part of it must go to the right and to the left.

2.1 Initialization

Define $P_{i,j} = \sum_{k=1}^{j} h_{ik}$ and $P_{i,0} = 0$. Moreover define for $1 \le j_1, j_2 \le n$

$$\Sigma_i(j_1, j_2) = \sum_{j=j_1}^{j_2} h_{ij} = \begin{cases} P_{i,j_2} - P_{i,j_1-1} & \text{if } j_2 \ge j_1 \\ h_i - P_{i,j_1-1} + P_{i,j_2} & \text{if } j_2 < j_1 \end{cases}$$

At the initialization we calculate and store the values of $P_{i,j}$ and h_i. Using these values we can evaluate any $\Sigma_i(j_1, j_2)$ in $O(1)$ time.

2.2 Phase I

We calculate the slackness $S(i,j) = g_i + g_j - L_h(i,j)$ for every pair $1 \le i,j \le n$ and decrease the capacities g_i in order to have $\min_j S(i,j) = 0$. (If the cut condition is not satisfied somewhere we stop with an error). Moreover for each i we calculate right_i that is the first number j in interval $(i, i-1]$ such that $S(i,j) = 0$.

```
for $i = 1$ to $n$ {
    $S(i, i+1) := g_i + g_{i+1} - h_{i+1}$
    $\min_i := S(i, i+1)$; $\text{right}_i := i+1$
    for $j = i+2$ to $i-1$ {
        $S(i,j) := S(i, j-1) - g_{j-1} + g_j + \Sigma_j(i+1, j-1) - \Sigma_j(j+1, i)$
        if $S(i,j) < \min_i$ then $\min_i := S(i,j)$; $\text{right}_i := j$
    }
    if $\min_i < 0$ then print "c-c is violated at" $e_i, e_{\text{right}_i}$ ; stop
    $g_i := g_i - \min_i$
}
```

Phase I clearly runs in time $O(n^2)$. If the c-c was not satisfied then we observe this fact, print a faulty cut and stop. It is not hard to see that the converse is also true, if at the beginning the c-c was satisfied then it remains satisfied along the procedure. Thus, from now on, we assume that the c-c is satisfied.

If during Phase I any g_i becomes 0 the routing problem becomes trivial. So from now on we also assume $g_i > 0$ for each $1 \le i \le n$. Remember that after this phase each edge has a tight pair.

2.3 Phase II

For notational convenience we put virtual vertices in the middle of each edge. The virtual vertex on edge e_i is called $v_{i+\frac{1}{2}}$.

Definition 2.1. The opposite vertex of v_i is v_j (where $i \ne j$ are integers; notation: $\text{op}(i) = j$), if there are $k \in [i,j)$ and $l \in [j,i)$ so that the pairs (e_k, e_j) and (e_l, e_{j-1}) are tight, but neither P_{ij} nor P_{ji} double-crosses any tight cut. The opposite vertex of v_i is the virtual vertex $v_{j+\frac{1}{2}}$, if there are $k_1, k_2 \in [i,j]$ and $l \in (j,i)$ so that the pairs (e_{k_1}, e_{k_2}) and (e_l, e_j) are tight.

We will show that every vertex has an opposite vertex and this is almost unique.

Phase II:

Repeat the following for each $i = 1, \ldots, n$. For $j' = i-1,\ i-2, \ldots, i+1$, define j as the first value j' satisfying

$$\text{right}_{j'-1} \in [j', i).$$

Define first $\text{op}(i) = j$, and if $j \ne i+1$ check the following. For $j' = i,\ i+1, \ldots, j-2$, check whether

$$\text{right}_{j'} \in [i, j).$$

If such a j' exists redefine $\text{op}(i) = j - \frac{1}{2}$.

Theorem 2.2. *Phase II finds* $\mathrm{op}(i)$ *for each* i*, in total running time* $O(n^2)$*, and* $\mathrm{op}(i)$ *is the opposite vertex of* v_i*. If* $\mathrm{op}(i) = j$ *is an integer then it is the unique opposite vertex of* i*, and for each* $k \in (i,j)$*,* h_{ik} *is uniquely routable to the right, for each* $l \in (j,i)$*,* h_{il} *is uniquely routable to the left, and* h_{ij} *is not uniquely routable. Moreover* $\mathrm{op}(\mathrm{op}(i)) = i$*.*

If $\mathrm{op}(i) = j - \frac{1}{2}$ *is not an integer then for each* $k \in (i, j-1]$*,* h_{ik} *is uniquely routable to the right, for each* $l \in [j,i)$*,* h_{il} *is uniquely routable to the left.*

Proof. Fix an arbitrary i and first suppose that there exists a $j \neq i$ such that h_{ij} is not uniquely routable. First we prove that in this case j is unique. Taking any $j_1 \neq i$ and $j_2 \in (j_1, i)$ the edge e_{j_1} has a tight pair e_k. If $k \in [i, j_1)$ then h_{ij_2}, otherwise h_{ij_1} is uniquely routable.

Next we show that the algorithm above terminates with $\mathrm{op}(i) = j$. As h_{ij} is not uniquely routable, clearly $\mathrm{right}_{j-1} \in [j,i)$. If for some $j' \in [j+1, i)$ we have $\mathrm{right}_{j'-1} \in [j', i)$ then h_{ij} would be uniquely routable (to the right). So after the first j'-loop the algorithm defines $\mathrm{op}(i) = j$. If in the second j'-loop the algorithm found a j' with $\mathrm{right}_{j'} \in [i,j)$ then h_{ij} would be uniquely routable (to the left), so no redefinition of $\mathrm{op}(i)$ is done.

Since e_{j-1} and e_j both have a tight pair on the other side of j (looking from i), clearly every demand h_{ik} is uniquely routable for $i \neq k \neq j$.

As h_{ji} is not uniquely routable iff h_{ij} is not uniquely routable, we immediately get that $\mathrm{op}(\mathrm{op}(i)) = i$.

Now suppose h_{ij} is uniquely routable for all $j \neq i$. The first j'-loop clearly finds some j (at the latest $j' = i+1$ satisfies the property). Since h_{ij} is uniquely routable and P_{ji} does not double-cross any tight cuts (because j was the first j' satisfying the property), P_{ij} must double-cross a tight cut, consequently $j \neq i+1$ and the second j'-loop finds such a cut and redefines $\mathrm{op}(i) = j - \frac{1}{2}$.

If $k \in (i, j-1]$ then the cut determined by e_{j-1} and its tight pair found in the first j'-loop shows that h_{ik} is uniquely routable (to the right). If $l \in [j,i)$ then the cut determined by the tight pair found in the second j'-loop shows that h_{il} is uniquely routable (to the left). □

2.4 Phase III

We are going to route every demand h_{ij} if $j \neq \mathrm{op}(i)$. We must take care about not to route any h_{ij} twice and do the update for each g_i in time $O(n)$.

```
for  $i = 1$  to  $n$  {
  $j^* := \lceil \mathrm{op}(i) - 1 \rceil$
  if  $j^* \neq i$ then
  for  $j = i+1$  to  $j^*$  {
    $r_{ij} := h_{ij}$;  $r_{ji} := 0$
    $g_{j-1} := g_{j-1} - \Sigma_i(j, j^*)$
    $h_{ij} = 0$
  }
}
```

As $\Sigma_i(j,j^*)$ can be calculated in constant time, the total time for Phase III is really $O(n^2)$, if we prove that we do not need to recalculate the values $P_{i,j}$. It is enough to see that the $\Sigma_i(j,j^*)$ values used had not been changed during this phase before. In the inner loop, due to the proper ordering of the j values, they have not been changed. In the outer loop the used h_{ij} value has not been changed before because when using this value we want to route uniquely routable demand h_{ij} to the right, so previously we did not route h_{ji} to the right.

Since h_{ij} is uniquely routable to the left iff h_{ji} is uniquely routable to the right, we routed all uniquely routable demands exactly once. (By the previous theorem h_{ij} is uniquely routable to the right iff $j \in (i, j^*]$.)

By Corollary 1.3 the c-c remains satisfied, every g_j remains non-negative end every tight cut remains tight. By Theorem 2.2 for every i at most one h_{ij} remains positive. Recalculating h_i values we have $h_{ij} = h_i$ if $j = \mathrm{op}(i)$ and 0 otherwise.

2.5 Phase IV

For each i route

$$r_{ij} = \frac{h_i + g_i - g_{i-1}}{2}$$

from demand h_i to the right. (And consequently $r_{ji} = \frac{h_i - g_i + g_{i-1}}{2}$ to the left, but we actually route this part to the right from $\mathrm{op}(i)$.)

Theorem 2.3. *The algorithm described finds a fractional solution if it exists in time $O(n^2)$. If the problem is Eulerian then the solution found is integral.*

Proof. By Theorem 1.1 we know that a routing exists. As every edge is in a tight cut, in any routing we must use all capacity of every edge. As routing any demand differing from h_{ij} uses the same amount of capacity on edges e_{i-1} and e_i, the routing of h_{ij} in any solution must be the same as defined in Phase IV. Clearly the routing we got is half-integral. If the original problem was Eulerian then this property remains true throughout the algorithm resulting an integral routing. It is obvious that Phase IV runs in time $O(n^2)$. □

3 Integer routing, unsplittable routing

If the input is integer but is not Eulerian we can get an integer solution by using the previous algorithm twice. Let $1 \le i_1 \le i_2 \le \ldots \le i_k \le n$ be the indices, where $g_{i_j - 1} + g_{i_j} + h_{i_j}$ is odd. First subtract one from each g_j where $j \in [i_l, i_{l+1})$ for an l odd. Clearly this modified problem is Eulerian. Use the previous algorithm, either it gives an integer solution or it gives a violating pair of edges, say e_{k_1} and e_{k_2}. Then restore all the original g_is and subtract one from each g_j where $j \in [i_l, i_{l+1})$ for an l even (also for $j \in [i_k, i_1)$). Use the previous algorithm again, either it gives an integer solution or it gives a violating pair of edges, say e_{l_1} and e_{l_2}.

By parity reasoning it is easy to see, that if there is an integral routing for the original problem, it is also a valid routing either for the first or for the second modified problem, so we surely find it.

It is also true that $\{e_{k_1}, e_{k_2}\}$ and $\{e_{l_1}, e_{l_2}\}$ form an odd crossing pair of tight cuts. We only sketch the proof (which also proves Theorem 1.2). If the two cuts do not cross, then one of the four cuts determined by $\{e_{k_i}, e_{l_j}\}$ would violate the c-c, because by parity reasons these cuts cannot be tight. If they cross then clearly they are odd (in all four "quarters" there are odd numbers of non-Eulerian vertices).

If we want to approximate the unsplittable routing problem, or the ring loading problem, then after the algorithm we use Theorem 5.1 of Schrijver, Seymour and Winkler. The algorithm describes there for unsplitting also runs in $O(n^2)$.

References

[1] A. Frank, *Edge-disjoint paths in planar graphs*, J. of Combinatorial Theory, Ser. B. No. 2 (1985), pp. 164-178.

[2] H. Okamura and P.D. Seymour, *Multicommodity flows in planar graphs,* J. Combinatorial Theory, Ser. B, 31 (1981) pp. 75-81.

[3] H. Ripphausen-Lipa, D. Wagner and K. Weihe, *Survey on Efficient Algorithms for Disjoint Paths Problems in Planar Graphs*, DIMACS-Series in Discrete Mathematics and Theoretical Computer Science, Volume 20 on the "Year of Combinatorial Optimization" (W. Cook and L. Lovász and P.D. Seymour eds.), AMS (1995) pp. 295-354.

[4] A. Schrijver, P. Seymour, and P. Winkler, *The ring loading problem*, SIAM J. Discrete Math. Vol. 11, No 1. (1998) pp. 1-14.

[5] D. Wagner and K. Weihe, *A linear-time algorithm for edge-disjoint paths in planar graphs*, Combinatorica 15 (1995) pp. 135-150.