Discrete Applied Mathematics 45 (1993) 169–179
North-Holland 

# On paths with the shortest average arc length in weighted graphs

Shmuel Wimer

*IBM Science and Technology, Matam, Haifa 31905, Israel*

Israel Koren

*Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003, USA*

Israel Cederbaum

*Department of Electrical Engineering Technion – Israel Institute of Technology, Haifa 32000, Israel*

Received 24 July 1990
Revised 17 June 1991

*Abstract*

Wimer, S., I. Koren and I. Cederbaum, On paths with the shortest average arc length in weighted graphs, Discrete Applied Mathematics 45 (1993) 169–179.

The problem of finding the path having the smallest average arc length in an acyclic digraph with a single source and a single sink is considered in this paper. This problem arises in VLSI block placement procedures when spreading the building blocks uniformly over the chip area is attempted. A well-known approximation algorithm to find the path with the minimum weight-ratio in a doubly-weighted graph can solve this problem. It combines a combinatorial algorithm with numerical iterations and its time complexity is $O(|U|^3 \log 1/\varepsilon)$, where $|U|$ is the number of vertices and $\varepsilon$ is the desired accuracy. This paper presents two new algorithms. The first, called the *path length minimization* algorithm, is based on the same principles as the algorithm presented by Karp, and can also be applied to undirected graphs. It is purely combinatorial and has $O(|U|^3)$ time complexity. We show how this algorithm for finding the path with the minimum average arc length can be extended to solve the more general problem of finding the path with the minimum weight-ratio in a doubly-weighted graph for which the secondary arc weights are positive integral or rational numbers. The second algorithm, called the *vertex balancing* algorithm, approximates the minimum average arc length path in any desired accuracy. It also combines a combinatorial algorithm with numerical iterations. Though having an exponential time complexity, it has been used successfully, achieving rapid convergence in all the practical cases which have been encountered.

*Correspondence to:* Professor I. Koren, Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003, USA.

## 1. Introduction

Let $G(U,E)$ be a finite weighted acyclic digraph having one source and one sink, denoted by $s$ and $t$, respectively. This paper studies the problem of finding a path from $s$ to $t$ along which the average arc length is minimum among all paths from $s$ to $t$. The problem of finding the path having the minimum average arc length arises in VLSI block placement. There, in order to avoid block congestion and make the routing of interconnections feasible, the building blocks have to be spread uniformly over the chip area. The uniform spreading problem was modeled and solved in [1] via an acyclic weighted digraph, for which a path with a minimal average arc length is found.

This paper presents two approaches to solve the minimum average arc length path problem, and some extensions to find the path having the minimum weight-ratio in doubly-weighted graphs. One is purely combinatorial, yielding the desired path. This algorithm follows the same principles as the algorithm presented by Karp [4] which finds the minimum cycle mean in a digraph. The second combines a combinatorial algorithm with numerical iterations, and finds a path which solves the problem for any desired accuracy, defined as follows. Let $\Omega$ be a path from $s$ to $t$ along which the average arc length is minimal, and let $\varepsilon$ be any real positive number. We say that a path $\Delta$ from $s$ to $t$ solves the problem with accuracy $\varepsilon$ if its average arc length satisfies

$$\frac{l(\Delta)}{|\Delta|}-\frac{l(\Omega)}{|\Omega|}\leq\varepsilon, \tag{1}$$

where $l(\Omega)$ and $l(\Delta)$ are the lengths of $\Omega$ and $\Delta$, respectively, and $|\Omega|$ and $|\Delta|$ are their cardinalities.

One way to solve the above problem is to apply the well-known approximation algorithm for finding the path with minimum weight-ratio in a doubly-weighted graph [2]. There, two positive weights are assigned to every arc $e\in E$: a *primary weight* denoted by $l(e)$ and a *secondary weight* denoted by $w(e)$. The problem then is to find a path $\Phi$ for which the objective function $z(\Phi)$ given by the ratio $z(\Phi)=\sum_{e\in\Phi} l(e)/\sum_{e\in\Phi} w(e)$ is minimized. It is evident that by setting $w(e)=1$ for every $e\in E$, the above problem turns out to be the minimum average arc length path problem. The algorithm described in [2] comprises a combinatorial algorithm and numerical iterations. Its complexity is $O(|U|^3 \log 1/\varepsilon)$, where $|U|$ is the number of the vertices in $G$ and $\varepsilon$ is the desired accuracy.

The two new algorithms for the minimum average arc length path problem are presented in the following order. The first algorithm is described in Section 2. It is purely combinatorial and has $O(|U|^3)$ time complexity. Section 3 presents the second algorithm combining a combinatorial algorithm with numerical iterations. Though having exponential time complexity, it has been found to be very efficient for all the practical cases for which a solution of the problem has been attempted. Section 4 concludes the discussion.

## 2. The path length minimization algorithm

Let us number the vertices of the acyclic digraph $G$ as follows. First, every vertex is ranked according to the maximal cardinality (number of arcs) of some path from $s$ to the vertex. Clearly, $s$ has rank 0, $t$ has the highest rank, and no two vertices of the same rank lie on a common path. Next we number the vertices according to their rank, starting with $s$, then the vertices of rank 1 (in an arbitrary order), then the vertices of rank 2 and so on. This way $s$ is numbered by 1, $t$ is numbered by $|U|$ and for every arc $e(u,v)$ the vertex $u$ is assigned a smaller number than the vertex $v$ (see for example the *st*-numbering in [3]).

Let $u$ be a vertex on a path from $s$ to $t$. Obviously, among all the paths connecting $s$ with $u$ and having the same cardinality, only the shortest one (if there are several, choose one arbitrarily) can be a part of the shortest average arc length path from $s$ to $t$. With every vertex $u \in U$ we associate a real-valued vector $L(u)$ of length $|U|$ whose elements are defined as follows. The $j$th element of $L(u)$, denoted by $L_j(u)$, $0 \le j \le |U|-1$, is the minimum length of any path from $s$ to $u$ whose cardinality is exactly $j$. Let $\Pi_j(u)$ denote a path yielding that minimum length (there may exist several ones). Clearly, the cardinality of a path cannot exceed $|U|-1$ since $G(U,E)$ is an acyclic digraph. If for some cardinality there exists no path from $s$ to $u$, an infinite length is assigned. We also associate with $u$ a vector $P(u)$ of length $|U|$, whose $j$th element, denoted by $P_j(u)$, indicates the last vertex preceding $u$ on $\Pi_j(u)$, i.e., the vertex $v$ for which $L_{j-1}(v)+l(e(v,u))=L_j(u)$. If $L_j(u)=\infty$, we set $P_j(u)=\phi$.

The algorithm proceeds iteratively. Starting from $s$, in each iteration a new vertex is marked until $t$ is reached. When a new vertex $u$ is marked we know the length of the shortest path from $s$ to $u$ for every cardinality between 0 and $|U|-1$. In the following algorithm $\Gamma^{\text{in}}(u)$ and $\Gamma^{\text{out}}(u)$ denote the sets of arcs entering and leaving $u \in U$, respectively.

*Step* 0: *Initialization.* Set $L_0(s)=0$ and $L_j(s)=\infty$, $1 \le j \le |U|-1$. Mark $s$ and set $T=U-\{s\}$. For every $u \in T$ set $L_j(u)=\infty$, $0 \le j \le |U|-1$. For every $u \in U$ define $P_j(u)=\phi$, $0 \le j \le |U|-1$.

*Step* 1: *New vertex selection.* Find a vertex $u \in T$ for which all the tail vertices of the arcs in $\Gamma^{\text{in}}(u)$ are already marked. Such a vertex must exist since $G(E,U)$ is an acyclic digraph with a single source and a single sink whose vertices are numbered as described above.

*Step* 2: *Updating the minimum path lengths.* Determine the shortest path length vector $L(u)$ by considering every vertex $v$ for which $e(v,u) \in \Gamma^{\text{in}}(u)$ as follows.

$$L_j(u) = \min\{L_{j-1}(v) + l(e(v,u)) \mid e(v,u) \in \Gamma^{\text{in}}(u)\}, \quad 1 \le j \le |U|-1. \quad (2)$$

Let $v^*$ be the vertex obtained when solving (2) for given $u$ and $j$. Then, set $P_j(u)=v^*$.

*Step* 3: *Updating the set of marked vertices.* Mark $u$ and set $T=T-\{u\}$.

*Step* 4: *Termination test.* If $u=t$ then go to Step 5, else go to Step 1.

*Step* 5: *Retrieving the minimum average arc length path.* Upon termination, every $L_j(t)<\infty$ is the length of the shortest path from $s$ to $t$ among all the paths of cardinality $j$. For every $j$ satisfying $L_j(t)=\infty$ there exists no path of cardinality $j$ from $s$ to $t$. Evidently, $\min\{L_j(t)/j \mid 1\le j\le |U|-1\}$ yields the minimum average arc length for any path from $s$ to $t$. Let $j^*$ be the cardinality of the path for which the minimum average arc length was obtained. Then, the desired path is retrieved by traversing backwards from $t$ to $s$ as follows. We start from $t$ and go backwards to the vertex stored in $p_{j^*}(t)$. We then go backwards to the vertex stored in $P_{j^*-1}[P_{j^*}(t)]$ and continue in the same manner until $s$ is reached.

Let us calculate the time complexity of the above algorithm. Notice that in order to find $L(u)$ and $P(u)$ for a vertex $u\in U$, we have to find for each $L_j(u)$, $1\le j\le |U|-1$, the minimum of a set of $|\Gamma^{\text{in}}(u)|$ expressions of type (2). This requires $O(|U||\Gamma^{\text{in}}(u)|)$ operations. Since $\bigcup_{u\in U}\Gamma^{\text{in}}(u)=E$, the total time complexity is $O(|U||E|)$, which in the worst case may be equal to $O(|U|^3)$. Notice that the above algorithm is applicable also to undirected graphs. Let $\Gamma(u)$ denote the set of the edges incident to $u$. Then, one has only to replace $\Gamma^{\text{in}}(u)$ by $\Gamma(u)$ in Step 1.

### *2.1. Minimum weight-ratio in doubly-weighted graphs*

As stated in the introduction, finding the minimum average arc length path is a special case of a more general problem in doubly-weighted graphs, of finding the path from $s$ to $t$ for which the ratio between the primary and the secondary path weights is minimized. In the following we propose an alternative to the algorithm presented in [2], by showing how the previous algorithm can be generalized to handle doubly-weighted graphs for which the secondary weights are nonnegative integral or rational numbers.

Assume first that the secondary weights are integral numbers. Let $\Omega'$ and $\Omega''$ be two different paths connecting $s$ with some vertex $u$, such that the secondary weights satisfy $\sum_{e\in\Omega'} w(e)=\sum_{e\in\Omega''} w(e)$. Obviously, if the corresponding primary weights satisfy $\sum_{e\in\Omega'} l(e)>\sum_{e\in\Omega''} l(e)$, $\Omega'$ cannot be a part of any path from $s$ to $t$ for which the ratio between the primary and the secondary path weights is minimized. If $\sum_{e\in\Omega'} l(e)=\sum_{e\in\Omega''} l(e)$ we can arbitrarily discard one of $\Omega'$ and $\Omega''$, since their effect on the weight-ratio of the paths passing through $u$ is identical. Let $W$ be the maximal secondary weight of any arc, i.e., $W=\max\{w(e)\mid e\in E\}$. Clearly, the total secondary weight of any path from $s$ to any vertex cannot exceed $(|U|-1)W$. The above observations lend themselves to an extension of the former algorithm in which we associate with every vertex $u\in U$ a real-valued vector $L(u)$ and a vector $P(u)$ both of length $(|U|-1)W+1$. For any index $j$, $0\le j\le(|U|-1)W$, $L_j(u)$ is the minimum primary weight of any path $\Omega$ from $s$ to $u$ satisfying $\sum_{e\in\Omega} w(e)=j$. Let $\Pi_j(u)$ denote the path yielding the minimum primary weight. If for some $j$ there exists no path from $s$ having that secondary weight, an infinite primary weight is assigned. The $j$th element of $P(u)$ indicates the last vertex preceding $u$ on $\Pi_j(u)$, i.e., the vertex $v$ for which $L_j(u) = L_{j-w(e(v,u))}(v) + l(e(v,u))$.

It is clear now how to apply the path length minimization algorithm to this problem. We will not elaborate on that and only remark that instead of dealing with path cardinalities one should consider their secondary weights, and change the range of the index $j$ from $0 \le j \le |U|-1$ to $0 \le j \le (|U|-1)W$ accordingly. In particular, the retrieval of the optimal path in Step 5 is started from $t$ and goes back to the vertex stored in $P_{j^*}(t)$. Then, we go backwards to the vertex stored in $P_{j^*-w(P_{j^*}(t),t)}[P_{j^*}(t)]$ and so forth, until $s$ is reached. Since in every iteration of the algorithm we have to consider all the possible secondary weights, it turns out that its worst-case time complexity is $O(W|U|^3)$.

Notice that the above algorithm is applicable also if all the secondary arc weights are a product of an integral positive number and a real positive constant, since such a multiplication preserves the path with the minimum weight-ratio. Consequently, the algorithm works also for the case of rational positive secondary weights. One has only to find the minimal common denominator of all the secondary arc weights and then specify it as the multiplicative factor.

## 3. The vertex balancing algorithm

This section describes an algorithm that yields an approximated solution to the shortest average arc length path problem. Although it has an exponential time complexity, in practice, it solves the problem in $O(|U|^2)$ steps, as compared to the $O(|U|^3)$ steps required by the first algorithm. This algorithm proceeds iteratively, where in every iteration the length of $G$'s arcs is modified by considering its vertices one by one, in such a way that the length of every path from the source $s$ to the sink $t$ is invariant. We call this operation a *balancing cycle*. Let $G_n$ denote the graph obtained after the $n$th iteration, and let $l_n(e)$ be the length of an arc $e$ in $G_n$. We shall prove that the series $\{l_n(e)\}$ converges uniformly for every arc $e \in E$, thus resulting in a limit graph denoted by $G_\infty$. Moreover, $G_\infty$ possesses the property that for each of its vertices except $s$ and $t$, the length of the shortest entering arc is equal to the length of the shortest leaving arc. We then show how this property yields the path having the minimum average arc length.

We first present some notations and definitions. Let $\alpha(u)$ and $\beta(u)$ denote the shortest length of any entering and any leaving arc of a vertex $u$, respectively, i.e.,

$$\alpha(u) = \min\{l(e) \mid e \in \Gamma^{\text{in}}(u)\}, \quad u \in U - \{s\}, \tag{3a}$$

$$\beta(u) = \min\{l(e) \mid e \in \Gamma^{\text{out}}(u)\}, \quad u \in U - \{t\}. \tag{3b}$$

We define $\mu(u)$ to be the *imbalance* of the vertex $u$,

$$\mu(u) = \beta(u) - \alpha(u), \quad u \in U - \{s, t\}. \tag{3c}$$

$\alpha_n(u)$, $\beta_n(u)$ and $\mu_n(u)$ are defined similarly for $G_n$. The graph $G$ is said to be *balanced* if

$$\mu(u) = 0, \quad \forall u \in U - \{s, t\}. \tag{4}$$

The iterative algorithm described below transforms the original $G$ into an infinite series of isomorphic graphs $\{G_n\}$ converging to a balanced graph denoted by $G_\infty$. In contrast to the previous algorithm which required the vertices to be numbered, their order in the following algorithm can be arbitrary. However, to simplify the convergence proof we first assume that they are numbered as before, and then we show that the algorithm is valid for any ordering.

*Step* 0: *Initialization.* Set $G_0 = G$ and $n = 0$.

*Step* 1: *Defining a new iteration.* Set $n = n + 1$.

*Step* 2: *Performing a balancing cycle.* For every $u \in U - \{s, t\}$ repeat the following process in the order defined by the vertex numbering. Calculate first the imbalance $\mu(u)$ and then update the length of all its entering and leaving arcs as follows:

$$\begin{aligned} l(e) &= l(e) + \tfrac{1}{2}\mu(u), \quad \forall e \in \Gamma^{\text{in}}(u); \\ l(e) &= l(e) - \tfrac{1}{2}\mu(u), \quad \forall e \in \Gamma^{\text{out}}(u). \end{aligned} \tag{5}$$

Denote by $G_n$ the resultant graph after processing all the vertices in the $n$th iteration, $n = 0, 1, 2, \ldots$, and by $\mu_n(u)$ the imbalance of a vertex $u \in U - \{s, t\}$ in $G_n$.

*Step* 3: *Termination test.* Let $\delta$ be a real positive parameter controlling the accuracy of the solution. Then, if $\max\{|\mu_n(u)| \mid u \in U - \{s, t\}\} < \delta$ go to Step 4, else go to Step 1.

*Step* 4: *Retrieving the minimum average arc length path $\Delta$.* For each $u \in U - \{s\}$ let $R(u)$ denote the vertex in $G_n$ for which the length of the arc $(R(u), u)$ is minimal among all the arcs $e \in \Gamma^{\text{in}}(u)$, i.e., $l_n(R(u), u) = \alpha_n(u)$. Then, the desired path is retrieved by traversing backwards from $t$ to $s$ as follows. We start from $t$ and go backwards to the vertex $R(t)$. Then we go backwards to the vertex $R(R(t))$ and continue in the same manner, until $s$ is reached.

### *3.1. Convergence of the algorithm*

We still have to show that the convergence assumption of the infinite series $\{G_n\}$ in Step 3 is always true, and that the path retrieved in Step 4 achieves the minimum average arc length for any desired accuracy. We prove first the convergence.

**Lemma 3.1.** *The infinite series of graphs $\{G_n\}$ resulting from the vertex balancing algorithm by ignoring the termination test of Step* 3 *converges to a graph $G_\infty$, satisfying*

$$\mu_\infty(u) = 0, \quad \forall u \in U - \{s, t\}. \tag{6}$$

**Proof.** Define

$$\mu_n = \max\{|\mu_n(u)| \mid u \in U - \{s,t\}\}. \tag{7}$$

Let $q$ be the maximal number of vertices along a path from $s$ to $t$ (excluding $s$ and $t$}. We show next that there exists a real nonnegative number $0 \le \gamma \le 1 - 1/2^q$ such that

$$\mu_{n+1} \le \gamma\mu_n, \quad n = 0, 1, 2, \ldots. \tag{8}$$

The validity of (8) implies that the imbalance of each vertex uniformly converges to zero, since $\mu_{n+1} \le \gamma\mu_n \le \gamma^2\mu_{n-1} \le \cdots \le \gamma^{n+1}\mu_0$.

To prove (8) notice that the vertex balancing operation in Step 2 of the algorithm equally shortens (lengthens) the length of every entering arc, and equally lengthens (shortens) the length of every leaving arc. Also, recall that during a balancing cycle the imbalance of every vertex is reset to zero once, and later on in this cycle it may become unbalanced when an adjacent vertex is balanced. In principle, the balancing of a vertex $u$ may affect only the imbalance of its adjacent vertices, which may increase in the worst case by half of $u$'s imbalance. Recall that the vertices are numbered and let $e(u,v) \in \Gamma^{\text{out}}(u)$. Then, the imbalance of $v$ immediately after balancing $u$ is increased by at most $\frac{1}{2}\mu_n$, i.e., its imbalance is bounded by $\mu_n + \frac{1}{2}\mu_n = 1\frac{1}{2}\mu_n$. Let $e(v,w) \in \Gamma^{\text{out}}(v)$. Then, the imbalance of $w$ immediately after the balancing of $v$ takes place, is increased by at most $\frac{3}{4}\mu_n$, i.e., it is bounded by $\mu_n + \frac{1}{2}(\mu_n + \frac{1}{2}\mu_n) = 1\frac{3}{4}\mu_n$. The effect of balancing a vertex on the remaining vertices propagates along the paths passing through it. Consequently, only those vertices lying on paths passing through $u$ may be affected by the balancing of $u$. Moreover, this effect decreases in integral powers of $\frac{1}{2}$ with the distance from $u$.

When the imbalance of a vertex $u$ is considered, one entering and one leaving arc are determined (see equation (3)). Suppose that the vertices along the maximal cardinality path are numbered $u_1, u_2, \ldots, u_q$ ($s$ and $t$ are excluded). Then, the maximal number of balancing operations during a balancing cycle that may affect the imbalance of $u_q$ is $q-1$. Therefore, the maximal quantity that can be added to the imbalance of $u_q$ during cycle $n+1$ is

$$\mu_n(\tfrac{1}{2} + \tfrac{1}{4} + \cdots + 1/2^{q-1}) = \mu_n(1 - 1/2^{q-1}), \tag{9}$$

and the total imbalance of $u_q$ prior to its balancing in cycle $n+1$ is bounded by $\mu_n(2 - 1/2^{q-1})$. Thus, after the imbalance of $u_q$ was reset to zero in this cycle, the imbalance of $u_{q-1}$ is bounded by $(1 - 1/2^q)\mu_n$. The inequality (8) follows by setting $\gamma = 1 - 1/2^q$.

The convergence of the graph series $\{G_n\}$ follows now immediately. First, the topology of the graph is invariant during the whole algorithm. Secondly, during a balancing cycle the length of each arc can be changed at most twice, once upon the balancing of its tail vertex and once upon the balancing of its head vertex. Since the maximal magnitude of a single change in the length of any arc during the $(n+1)$th balancing cycle is not greater than half of the actual imbalance, which is bounded by $\mu_n$, the overall change in any arc length cannot exceed $\mu_n$, which has been proven to converge to zero. □

We show now that the path $\Delta$ retrieved in Step 4 achieves the minimum average arc length for any desired accuracy.

**Lemma 3.2.** *Let $\varepsilon$ be the desired accuracy of the solution to the problem of finding the minimum average arc length path. The parameter $\delta$ in Step 3 can be chosen sufficiently small such that the path $\Delta$ retrieved in Step 4 of the algorithm solves the problem with accuracy $\varepsilon$, i.e., inequality (1) is satisfied.*

**Proof.** Let $\Omega$ be the path obtained by applying Step 4 of the algorithm to $G_\infty$ (Lemma 3.1 proved that $G_\infty$ exists). According to Lemma 3.1 all the imbalances in $G_\infty$ are zero. Therefore, the arc lengths along $\Omega$ in $G_\infty$ are the same. Moreover, their length is minimal among the arc lengths in $G_\infty$. Otherwise, there would exist an arc $e \notin \Omega$ having smaller length than the length of the arcs along $\Omega$. Since the imbalance of a vertex is determined by its shortest entering and leaving arcs, and in $G_\infty$ the imbalances are zero, one could then construct a distinct path from $s$ to $t$ for which the arc lengths are all equal, and are smaller than those of $\Omega$. This contradicts the fact that the construction of $\Omega$ in Step 4 started with the shortest arc in $\Gamma^{\text{in}}(t)$. Consequently, $\Omega$ solves the problem in $G_\infty$. Since the lengths of isomorphic paths from $s$ to $t$ are identical in $G$ and $G_\infty$, it follows that the average arc length along paths from $s$ to $t$ is invariant, and hence $\Omega$ solves the problem in $G$, too.

We now set the termination parameter $\delta = \varepsilon/2^{|U|-1}$ and show that in $G_\infty$ (and as stated above, in $G$, too) inequality (1) is satisfied. It has been shown in the proof of Lemma 3.1 that the magnitude in which any arc length can be changed during a balancing cycle is not greater than the product of $1-1/2^q$ and the maximal imbalance after the previous balancing cycle. Therefore, the difference between the length of any arc when the algorithm terminates (due to the condition stated in Step 3), and its length in $G_\infty$ cannot exceed

$$\delta \sum_{r=0}^{\infty} (1-1/2^q)^r = 2^q \delta \le 2^{|U|-2}\delta = \frac{\varepsilon}{2}. \tag{10}$$

Let the path $\Delta$ that was retrieved in Step 4 of the balancing algorithm consist of the arcs $e_1, \dots, e_k$, and the minimum average arc length path $\Omega$ consist of the arcs $a_1, \dots, a_m$. The fact that when the algorithm terminates the imbalance is smaller than $\delta$, and the way $\Delta$ was constructed, imply that the lengths of the arcs $e_k$, $e_{k-1}$, $e_{k-2}, \dots, e_1$, in $G_n$, do not exceed the values $l_n(e_k), l_n(e_k)+\delta$, $l_n(e_k)+2\delta, \dots, l_n(e_k)+(k-1)\delta$, respectively. Therefore,

$$\frac{l(\Delta)}{|\Delta|} = \frac{l_n(\Delta)}{|\Delta|} \le \sum_{i=1}^{k} \frac{l_n(e_k)+(k-i)\delta}{k} < l_n(e_k) + \frac{k}{2}\delta. \tag{11}$$

Since in Step 4 of the balancing algorithm the arc $e_k$ was selected as the arc of minimal length among all the arcs in $\Gamma^{\text{in}}(t)$ (to which $a_m$ belongs too), the following is true:

$$l_n(e_k) \le l_n(a_m). \tag{12}$$

According to (10), the difference between the length of an arc in $G_n$ and its length in $G_\infty$ is bounded by $\varepsilon/2$. Hence,

$$l_n(a_m) \le l_\infty(a_m) + \frac{\varepsilon}{2} = \frac{l_\infty(\Omega)}{|\Omega|} + \frac{\varepsilon}{2} = \frac{l(\Omega)}{|\Omega|} + \frac{\varepsilon}{2}. \tag{13}$$

Finally, the minimality of $\Omega$ implies that

$$\frac{l(\Omega)}{|\Omega|} \le \frac{l(\Delta)}{|\Delta|}. \tag{14}$$

Combining equations (11)–(14) yield the following inequalities

$$\frac{l(\Delta)}{|\Delta|} \le l_n(e_k) + \frac{k}{2}\delta \le l_n(a_m) + \frac{k}{2}\delta \le \frac{l(\Omega)}{|\Omega|} + \frac{k}{2}\delta + \frac{\varepsilon}{2} \le \frac{l(\Delta)}{|\Delta|} + \frac{k}{2}\delta + \frac{\varepsilon}{2}, \tag{15a}$$

which in turn implies that

$$\frac{l(\Delta)}{|\Delta|} - \frac{k}{2}\delta - \frac{\varepsilon}{2} \le \frac{l(\Omega)}{|\Omega|} \le \frac{l(\Delta)}{|\Delta|} + \frac{k}{2}\delta + \frac{\varepsilon}{2}. \tag{15b}$$

Consequently,

$$\left| \frac{l(\Delta)}{|\Delta|} - \frac{l(\Omega)}{|\Omega|} \right| \le \frac{k}{2}\delta + \frac{\varepsilon}{2} < \frac{|U|}{2}\delta + \frac{\varepsilon}{2} < \varepsilon, \tag{15c}$$

which proves that the path $\Delta$ solves the problem in accuracy $\varepsilon$. □

It can be easily verified that any order of balancing the vertices within the balancing cycle yields convergence. This follows from the fact that the coefficient $\gamma = 1 - 1/2^q \le 1 - 1/2^{|U|-2}$ is an upper bound on the reduction of the imbalance from cycle to cycle, and it is independent of the order in which the vertices are considered during the cycles. Moreover, this order can vary from cycle to cycle, as long as each vertex is balanced once in every cycle. In general, convergence is guaranteed for an arbitrary balancing scheme, as long as the period between two consecutive treatments of any vertex is uniformly bounded.

### 3.2. *Complexity of the algorithm*

The complexity of the balancing algorithm is exponential in the number of vertices in $G$. Consider first a single balancing cycle. There, every arc is treated exactly twice: once upon the balancing of its tail vertex, and once upon the balancing of its head vertex. Therefore, the time complexity of a balancing cycle is $\Theta(|E|)$ which is $\Theta(|U^2|)$ in the worst case.

Table 1. Number of iterations as a function of the maximal path cardinality and the desired accuracy for uniformly distributed arc lengths

| $\delta$ | $q$ | | | |
|---|---|---|---|---|
| | 20 | 50 | 100 | 200 |
| $10^{-4}$ | $1.8\times10^2$ | $6.2\times10^2$ | $1.8\times10^3$ | $4.0\times10^3$ |
| $10^{-7}$ | $5.0\times10^2$ | $2.4\times10^3$ | $8.6\times10^3$ | $3.1\times10^4$ |
| $10^{-10}$ | $7.2\times10^2$ | $4.2\times10^3$ | $1.6\times10^4$ | $6.2\times10^4$ |

To calculate the number of balancing cycles, let $\varepsilon$ be the desired accuracy and let $n$ denote the number of balancing cycles required to achieve this accuracy. Any imbalance during the execution of the balancing algorithm is bounded by the largest arc length in $G$. Assume for convenience that this is a unit magnitude. It has been shown in Lemma 3.1 that the factor $(1-1/2^{|U|-2})^{-1}$ is a lower bound on the reduction of the imbalance from cycle to cycle. Also, to obtain accuracy $\varepsilon$, we set in Lemma 3.2 the termination parameter $\delta$ to be equal to $\varepsilon/2^{|U|-1}$. Therefore, the following equation determines the number of iterations needed to terminate the algorithm,

$$(1-1/2^{|U|-2})^n = \frac{\varepsilon}{2^{|U|-1}}, \tag{16}$$

which after some manipulations yields $n=\Theta[2^{|U|}(|U|+\log(1/\varepsilon))]$.

Surprisingly, in practice, for problems where $|U|=O(10^2)$, the algorithm converges very fast. In contrast to the above worst-case analysis, the balancing of a vertex may in practice, reduce the imbalance of some of its adjacent vertices, which in turn reduces substantially the number of iterations needed to achieve a given accuracy. Table 1 provides some information on the measured efficiency of the vertex balancing algorithm. There, the number of balancing cycles is shown as a function of the required accuracy and the largest cardinality of a path in the given graph. The arc lengths were drawn from a uniform distribution in the interval [0, 1]. One can see that the number of balancing iterations as a function of $q$ grows quadratically rather than exponentially.

One may expect that if a zero length would be assigned to all the arcs in $G$ except

Table 2. Number of iterations as a function of the maximal path cardinality and the desired accuracy for 0-1 arc lengths

| $\delta$ | $q$ | | | |
|---|---|---|---|---|
| | 20 | 50 | 100 | 200 |
| $10^{-4}$ | $2.1\times10^2$ | $8.3\times10^2$ | $1.9\times10^3$ | $2.5\times10^3$ |
| $10^{-7}$ | $4.9\times10^2$ | $2.6\times10^3$ | $8.9\times10^3$ | $3.0\times10^4$ |
| $10^{-10}$ | $7.7\times10^2$ | $4.3\times10^3$ | $1.6\times10^4$ | $5.7\times10^4$ |

one arc in $\Gamma^{\text{in}}(t)$ which would be assigned a unit length, then a much larger number of balancing iterations will be required. This follows from the unit imbalance that we start with and its propagation to other vertices. Table 2 shows the results for this case for a graph isomorphic to that of Table 1. Surprisingly, the results are almost the same as those for the uniformly distributed arc lengths.

## 4. Discussion

Although the worst-case analysis of the vertex balancing algorithm yields exponential complexity, in all the practical examples which have been encountered, a quadratic growth in the number of balancing iterations was observed. It is interesting to find out why the performance, in practice, is extremely better than the theoretical one. Also, it might be worthwhile to construct an example for which the number of balancing iterations approaches the worst-case complexity.

## Acknowledgement

The authors would like to thank an anonymous referee for his helpful comments and suggestions, and the correction of the proof of Lemma 3.2.

## References

[1] I. Cederbaum, I. Koren and S. Wimer, Balanced block spacing for VLSI layout, Discrete Appl. Math. 40 (1992) 303–318.
[2] N. Christofides, Graph Theory–An Algorithmic Approach (Academic Press, New York, 1975) Ch. 8.
[3] S. Even, Graph Algorithms (Computer Science Press, Rockville, MD, 1979).
[4] R.M. Karp, A characterization of the minimum cycle mean in a digraph, Discrete Math. 23 (1978) 309–311.