ELSEVIER

Discrete Applied Mathematics 49 (1994) 77–93

DISCRETE APPLIED MATHEMATICS

Balancing problems in acyclic networks*

Endre Boros[a, b, **], Peter L. Hammer[b], Mark E. Hartmann[c], Ron Shamir[d, ***]

[a] *DIMACS, Rutgers University, New Brunswick, NJ 08903, USA*
[b] *RUTCOR, Rutgers University, New Brunswick, NJ 08903, USA*
[c] *Department of Operations Research, University of North Carolina, Chapel Hill, NC 27599, USA*
[d] *Department of Computer Science, Sackler Faculty of Exact Sciences, Tel Aviv University, Tel-Aviv 69978, Israel*

Received 15 August 1991; revised 17 March 1992

Abstract

A directed acyclic network with nonnegative integer arc lengths is called balanced if any two paths with common endpoints have equal lengths. In the buffer assignment problem such a network is given, and the goal is to balance it by increasing arc lengths by integer amounts (called buffers), so that the sum of the amounts added is minimal. This problem arises in VLSI design, and was recently shown to be polynomial for rooted networks. Here we give simple procedures which solve several generalizations of this problem in strongly polynomial time, using ideas from network flow theory. In particular, we solve a weighted version of the problem, extend the results to nonrooted networks, and allow upper bounds on buffers. We also give a strongly polynomial algorithm for solving the min-max buffer assignment problem, based on a strong proximity result between fractional and integer balanced solutions. Finally, we show that the problem of balancing a network while minimizing the number of arcs with positive buffers is NP-hard.

1. Introduction

The buffer assignment problem can be stated as follows: Given a directed acyclic graph with integer costs associated to the arcs, find additional integer arc costs so that the sums of arc costs along any two paths with common endpoints are equal, and the total sum of the costs added is minimum.

* Research partially supported by AFOSR grant 90-0008, by NSF grants DMS 89-06870 and STC88-09648, and by ONR grant N00014-92-71375.
** Corresponding author.
*** This work was done while the author was a visitor at RUTCOR.

SSDI 0166-218X(92)00131-V

The problem arises in VLSI design of special-purpose parallel machines, when one wishes to transform a data flow machine [7] into a systolic machine [9, 15]. In both architectures a network of processors is set up, and the computational task is decomposed into smaller tasks to be performed by the individual processors. The processors' network is modeled by a graph, where vertices correspond to processors and arcs indicate the flow of data in the network. The computation time of each processor is indicated by a length (or cost) attached to all the arcs emanating from its vertex. If for each vertex all inputs of the same computation arrive simultaneously, then we say that the network is *balanced*. Balanced networks have the advantage that they can process "pipelined" inputs and thus have higher utilization and throughput. A common way to make a network balanced is to introduce delays (or *buffers*) along certain arcs. It is easy to see that such balancing is always achievable in acyclic graphs. To minimize hardware costs, it is desirable to achieve this goal by using a minimum number of buffer units. The buffer assignment problem is thus how to convert a directed acyclic data flow network into a balanced one, by adding a minimum number of buffer units.

Because of its importance, the buffer assignment problem has been addressed in the past by several authors (see [5] for a list of references). Chang and Lee described an integer programming decomposition (nonpolynomial) procedure for this problem [5]. In [4] we have shown that this problem can in fact be solved in strongly polynomial time, provided that the graph has a root (a vertex such that either there exists a directed path from every vertex to it, or there exists a path from it to every other vertex). Using ideas from duality and network flow theory, an $O(mn \log n)$ algorithm was given in [4] for solving the problem on a rooted network with n vertices and m arcs.

In this paper we extend the investigation on buffer assignment problems in several directions:

- The restriction that the graph is rooted is removed, and it is shown that the buffer assignment problem is polynomial for any acyclic graph. A solution of the buffer assignment problem on such graphs in $O(mn \log n)$ operations is given.
- It is shown how to solve in $O(mn \log n + n^2 \log^2 n)$ operations a weighted version of the problem, where buffers on different arcs may have different weights and minimization of the weighted sum is required.
- An equivalence between the weighted buffer assignment problem and the feasible uncapacitated minimum cost flow (transshipment) problem on acyclic rooted graphs is proved.
- It is shown how to solve the unweighted and the weighted problems with additional upper bounds on buffer sizes, without an increase in the solution complexity.
- Polynomial and strongly polynomial solutions are given to the min-max version of the balancing problem, in which the minimization of the maximum buffer size is required.
- The problem of balancing a network while minimizing the number of arcs with positive buffers is shown to be NP-hard.

The paper is organized as follows: Section 2 addresses the weighted problem under the assumption that the network is rooted. The analysis is along the lines of [4], with

appropriate generalizations. The equivalence to transshipment on acyclic rooted graphs is also proved. Section 3 addresses the problem when the network has no root. To achieve synchronization in such networks, it is shown that a stronger definition of balance is necessary. Using the results of Section 2, it is shown how to solve the buffer assignment problem for nonrooted graphs using the algorithms for the unweighted case. Section 4 shows how to handle upper bounds on the size of buffers. Section 5 gives two algorithms for the min-max balancing problem, as well as a theorem which shows a strong proximity between fractional and integer balanced solutions. Section 6 shows that the problem of minimizing the number of buffered arcs is NP-hard. Section 7 contains concluding remarks and open questions.

2. Weighted buffer assignment problems

Let $G = (V, E)$ be a directed acyclic graph with n vertices and m arcs. Associated with each arc (i, j) are a nonnegative integer *length* w_{ij} and a nonnegative *cost* (or weight) q_{ij}. The length of a directed path in G is the sum of the lengths of the arcs along that path. Two distinct paths are called *parallel paths* if they have the same endpoints. A graph is *balanced with respect to w* if every two parallel paths have equal length. Given an unbalanced graph, the *Weighted Buffer Assignment Problem* is to increase the length of each arc (i, j) by a nonnegative integer number x_{ij} so that the graph is balanced with respect to the length function $w + x$, and the total cost added $\sum_{(i,j)\in E} q_{ij}x_{ij}$ is minimum. Because of the origin of the problem is in VLSI design, the x_{ij} are also called *buffers*.

A straightforward integer programming formulation of the problem is the following:

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} q_{ij}x_{ij}, \\ \text{s.t.} \quad & \sum_{(i,j)\in P_k} (w_{ij} + x_{ij}) = \sum_{(i,j)\in P_l} (w_{ij} + x_{ij}) \quad \forall \text{ parallel paths } P_k, P_l, \\ & x_{ij} \geqslant 0 \text{ integer.} \end{aligned} \tag{2.1}$$

A *root* in G is a vertex t such that either there exists a directed path from every vertex to t, or there exists a path from t to every vertex. G is called *rooted* if it contains such a root. We shall first analyze the weighted buffer assignment problem under the assumption that the graph is rooted. (In Section 3 we shall show how to remove this assumption.) We assume without loss of generality that the root t is an *output vertex*, i.e., there is a path from every vertex to t.

Since G has an output vertex t, G is balanced if and only if each two parallel paths ending at t have equal costs. Hence, instead of enumerating in the constraints of (2.1) all parallel paths, it suffices to consider only paths ending at t. (Note that even so the number of equations in (2.1) may grow *exponentially* with the number of vertices.) This is essentially the formulation proposed by Chang and Lee [5], who subsequently describe a decomposition method which solves several smaller integer programming problems instead of the original one.

A simple yet crucial first step in reformulating the problem in [4] was to introduce additional vertex variables: For a graph with arc lengths $\bar{w}_{ij}$, define a variable d_v for each vertex v, and write the following set of *distance equations*:

$$d_i = d_j + \bar{w}_{ij} \quad \text{for every } (i,j) \in E. \tag{2.2}$$

Proposition 2.1 [4]. *For a rooted directed acyclic graph G, the system of distance equations* (2.2) *is consistent if and only if G is balanced with respect to* $\bar{w}$.

The proposition is not true if we omit the requirement that the graph is rooted. This will be discussed further in Section 3. Note that the values of all the d_i are determined up to a common additive constant. If we set $d_t = 0$ then all d_v will be integers, and the value of d_v obtained in a solution to the system (2.2) is just the distance along any path from v to t in the corresponding balanced graph.

By the proposition, the following problem is equivalent to (2.1):

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} q_{ij}x_{ij}, \\ \text{s.t.} \quad & d_i = d_j + w_{ij} + x_{ij}, \quad (i,j)\in E, \\ & x_{ij} \geqslant 0 \text{ integer.} \end{aligned} \tag{2.1'}$$

Next, eliminate the x_{ij} variables from (2.1') by substituting $x_{ij} = d_i - d_j - w_{ij}$. The weighted objective function becomes

$$\begin{aligned} \sum_{(i,j)\in E} q_{ij}x_{ij} &= \sum_{(i,j)\in E} q_{ij}(d_i - d_j - w_{ij}) \\ &= \sum_{v\in V} d_v \left(\sum_{\{j\,|\,(v,j)\in E\}} q_{vj} - \sum_{\{i\,|\,(i,v)\in E\}} q_{iv} \right) - \sum_{(i,j)\in E} q_{ij}w_{ij}. \end{aligned}$$

Define now $\rho_v := \sum_{\{j\,|\,(v,j)\in E\}} q_{vj} - \sum_{\{i\,|\,(i,v)\in E\}} q_{iv}$. The constant term $\sum_{(i,j)\in E} q_{ij}w_{ij}$ does not affect the set of optimal solutions, so our problem is equivalent to

$$\begin{aligned} \min \quad & \sum_{i\in V} \rho_i d_i, \\ \text{s.t.} \quad & d_i - d_j \geqslant w_{ij}, \quad (i,j)\in E, \\ & d_i \text{ integer.} \end{aligned} \tag{2.3}$$

The dual problem to (2.3) is

$$\begin{aligned} \max \quad & \sum_{(i,j)\in E} w_{ij}y_{ij}, \\ \text{s.t.} \quad & \sum_{\{i\,|\,(k,i)\in E\}} y_{ki} - \sum_{\{j\,|\,(j,k)\in E\}} y_{jk} = \rho_k, \quad k\in V, \\ & y_{ij} \geqslant 0 \text{ integer.} \end{aligned} \tag{2.4}$$

But (2.4) is a network flow problem: Interpreting ρ as excesses (supplies and demands) at the vertices, and y_{ij} as the flow along arc (i,j), the constraints in (2.4) are flow conservation constraints (see, e.g., [22]). Here w_{ij} is the per unit shipping cost

along arc (i, j), and the problem is to find a maximum cost flow. (Note that $\sum_{v\in V}\rho_v = 0$ is satisfied as required.) Since the constraint matrix for that problem is totally unimodular (see, e.g., [19]), the integrality condition can be relaxed without changing the solution value, thus we can immediately conclude that the weighted buffer assignment problem for rooted acyclic graphs is polynomial.

To describe efficient solutions for the problem, we use network flow theory. The reader is referred to [1, 12, 22] for terminology and algorithms on network flows. The approach developed in [4] for the unweighted case carries over to the weighted case, with appropriate changes in the procedures and in the complexity. For the sake of completion, and for the discussion in later sections, we repeat them briefly here.

To use existing minimum cost flow algorithms, we recast the problem as a minimization problem with nonnegative variables: For every vertex v let π_v be the length of the longest path from v to the root t. Hence, $\pi_i \geqslant \pi_j + w_{ij}$, so $\bar{w}_{ij} = w_{ij} - \pi_i + \pi_j \leqslant 0$ for all $(i, j)\in E$. Replacing the arc costs w_{ij} by the reduced costs $\bar{w}_{ij}$ does not change the set of optimal flows (cf. [12, 4]). To replace maximization by minimization, define $c_{ij} := -\bar{w}_{ij}$ for all $(i, j)\in E$. Hence, the sets of optimal flows for (2.4) and for the following minimization problem are identical:

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} c_{ij}y_{ij}, \\ \text{s.t.} \quad & \sum_{\{i\,|\,(k,i)\in E\}} y_{ki} - \sum_{\{j\,|\,(j,k)\in E\}} y_{jk} = \rho_k, \quad k\in V, \\ & y_{ij} \geqslant 0 \text{ integer.} \end{aligned} \tag{2.4'}$$

Problem (2.4′) is a transshipment (uncapacitated minimum cost flow) problem, with nonnegative costs, which can be solved directly by well-known algorithms. For example, the *excess scaling algorithm* of Edmonds and Karp [8] (see also [20]) requires $O(n \log U)$ flow augmentations on an uncapacitated problem with maximum absolute excess U. We shall mention other algorithms later.

To obtain an optimal solution to (2.3) from an optimal solution y to (2.4′), in the residual graph R_y, compute the shortest distance δ_v from each vertex v to t, with respect to the costs c_{ij}. Since y is an optimal flow, R_y contains no negative cost cycles, and all distances are well defined. By the properties of the shortest path distances, $\delta_j + c_{ij} \geqslant \delta_i$ for all arcs (i, j) in R_y. Setting $\bar{d}_v := \pi_v - \delta_v$, we get

$$\begin{aligned} \bar{d}_i - \bar{d}_j &= \pi_i - \delta_i - \pi_j + \delta_j \geqslant -c_{ij} + \pi_i - \pi_j \\ &= \bar{w}_{ij} + \pi_i - \pi_j = w_{ij}, \quad (i, j)\in R_y. \end{aligned} \tag{2.5}$$

Using the fact that y is also optimal for (2.4), and complementary slackness (see, e.g., [1]), the feasibility of y together with (2.5) imply that $\bar{d}$ is optimal for (2.3). The buffers are now readily given by $x_{ij} = \bar{d}_i - \bar{d}_j - w_{ij}$.

The following algorithm and theorem summarize the procedure for solving the weighted buffer assignment problem, and its complexity:

algorithm Buffer Assignment;
begin
1. For each $i\in V$, find π_i, the longest distance from i to t.
2. For each $(i, j)\in E$, set $c_{ij} := \pi_i - \pi_j - w_{ij}$.

3. Solve the minimum cost flow problem (2.4′) to obtain an optimal flow y for (2.4′) and (2.4).
4. For each $i \in V$, find δ_i, the shortest distance from i to t in R_y.
5. For each $(i, j) \in E$, set $x_{ij} := \pi_i - \delta_i - \pi_j + \delta_j - w_{ij}$.

end

Theorem 2.2. *The above procedure solves the buffer assignment problem for weighted rooted graphs in* $O(n \log n(m + n \log n))$ *arithmetic operations.*

Proof. Validity follows from the arguments above. As to the complexity, steps 2 and 5 take $O(m)$ time. Step 1 requires the solution of a longest path problem on an acyclic graph, which can be done in $O(m)$ time (see, e.g., [22]). Step 4 can be done, for example, in $O(m + n \log n)$ [10]. The bottleneck computation is solving problem (2.4′). Orlin's strongly polynomial algorithm [20] solves the problem in $O(n \log n)$ augmentations. Each augmenting path can be found in $O(m + n \log n)$ operations by using Fredman and Tarjan's shortest path algorithm [10]. □

Note that in case the cost coefficients are small, other weakly polynomial algorithms may provide better complexity.

If there are some arcs on which adding buffers is forbidden, the problem may be infeasible. To check this, assign sufficiently high weights to these arcs and use the algorithm above to solve the resulting problem. If in the solution buffers are introduced on any of the large weight arcs, then there is no solution which avoids putting buffers on forbidden arcs, and the original problem is infeasible. An alternative solution to this problem will be described in Section 4.

In unweighted buffer assignment problems, $q_{ij} = 1$ for all arcs. Hence $\rho_v = out(v) - in(v)$. The corresponding uncapacitated minimum cost flow problem (2.4′) has a special supply and demand structure, which was utilized in [4] to obtain more efficient algorithms than are known for the general transshipment problem. The weighted problem discussed above generates a more general supply and demand structure. A natural question is whether the weighted problem is as general as the transshipment problem on rooted graphs. In other words, under what conditions does a minimum cost flow problem on a rooted graph have an equivalent weighted buffer assignment formulation? This question is answered by the following theorem.

Theorem 2.3. *Every feasible minimum cost flow problem on an acyclic rooted network has a weighted buffer assignment formulation.*

Proof. The input to the minimum cost flow problem includes an excess value b_v for each node v. By the discussion above, an equivalent weighted buffer assignment formulation exists only if there are values q_{ij} such that

$$b_v = \sum_{\{j \mid (v,j) \in E\}} q_{vj} - \sum_{\{i \mid (i,v) \in E\}} q_{iv}, \quad v \in V, \quad q_{ij} \geqslant 0 \text{ integer}, \ (i,j) \in E.$$

Since these constraints are flow conservation constraints, and are *the same* as the flow constraints in the original problem, there exist weights for a corresponding weighted

buffer assignment formulation if and only if the flow problem is feasible. By total unimodularity, there will always be integral weights whenever the flow problem is feasible.

To obtain the corresponding weighted buffer assignment problem as a minimization problem with nonnegative costs, we use again the "trick" of replacing costs by more convenient reduced costs: Let π_v be the longest distance from v to t, which is well defined since the graph is rooted. Replace each cost coefficient c_{ij} in the minimum cost flow problem by the reduced cost $c_{ij} - \pi_i + \pi_j$. Hence, the resulting problem is equivalent to the original one. Since its cost coefficients are all nonpositive, reverse the signs of all coefficients and replace minimization by maximization to get another equivalent problem which has exactly the form of (2.4), the dual of a buffer assignment problem. □

3. Nonrooted graphs

Graphs which do not have a root may be balanced even though they cannot be realized as a synchronous system. The reason is that the definition of a balanced graph sets conditions only with respect to parallel paths. Consider the graph in Fig. 1. This graph is trivially balanced, since it contains no parallel paths, even though it does not correspond to a synchronous network: A signal leaving vertex a at time 0 will reach c and d at time 1. For the synchronous operation, all signals reaching a vertex should arrive simultaneously. Hence, the signal from b to c should leave b at time 0, but the same signal from b to d should leave b at time -1.

This example demonstrates that for a nonrooted network to be synchronous, a stronger condition is necessary. Namely, one should be able to assign times to the vertices which will reflect in a consistent way the time points at which any one signal passes through them. Formally, we define an acyclic graph G with arc lengths w to be *time consistent* (with respect to w) if there is an assignment of real numbers t to the vertices such that for every path P from v to u, $t_v - t_u = \sum_{(i,j)\in P} w_{ij}$. The buffer assignment problem for an acyclic (not necessarily rooted) graph is to assign a minimum number of buffer units to the arcs so that the resulting graph is time consistent. Clearly, a rooted graph is balanced if and only if it is time consistent. For the nonrooted case we have the stronger version of Proposition 2.1.

Proposition 3.1. *An acyclic graph G is time consistent if and only if the system of equations* (2.2) *is consistent.*

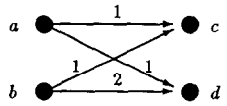

Fig. 1. A "balanced" graph which is not time consistent. The number above each arc is its length.

By Proposition 3.1, (2.1) is a valid formulation of the more general buffer assignment problem on acyclic graphs, and the analysis in Section 2 applies to it. In particular, the polynomiality result extends to nonrooted graphs.

As it turns out, we can solve nonrooted problems by combining the algorithms of [4] for the rooted case with the tools developed in Section 2 for the weighted case: Add an artificial root (output) vertex t to G. For every vertex v with outdegree zero add an artificial arc (v, t) with length $w_{vt} = 0$. The resulting graph $\bar{G} = (V \cup \{t\}, \bar{E})$ is now rooted, and if $\bar{G}$ is balanced then G is time consistent. We can thus solve the problem on G, but we want to preclude from the objective function the cost of the buffers on the artificial arcs. We can solve this problem as a *weighted* buffer assignment problem. Give every original arc (i, j) cost $q_{ij} = 1$, give every artificial arc (v, t) cost $q_{vt} = 0$, and solve the resulting weighted problem. Steps 1 and 2 in the procedure of Section 2 remain unchanged. In step 3, we need to solve the resulting minimum cost flow problem, where the supply at the root t, $\rho_t = 0$ and for each vertex $v \neq t$, its supply or demand is

$$\rho_v = \sum_{\{j|(v,j)\in\bar{E}\}} q_{vj} - \sum_{\{i|(i,v)\in\bar{E}\}} q_{iv} = out(v) - in(v),$$

where $in(v)$ and $out(v)$ are the indegree and outdegree of vertex v in the *original graph* G. In other words, in step 3 we can solve the minimum cost flow problem (2.4) for the *unweighted* original graph G, disregarding t and the artificial arcs. In particular, this means that this step can be done in the lower complexity of the unweighted case algorithms. Finally, in step 4, we introduce t and the artificial arcs again, and find a shortest path from each vertex to t in the residual graph for the final flow, where the flow on all artificial arcs is zero. The bottleneck for the modified procedure is still step 3, so we can draw the following conclusion.

Theorem 3.2. *The buffer assignment problem for acyclic (not necessarily rooted) networks is solvable in* $O(mn \log n)$ *arithmetic operations.*

Clearly, the same reasoning applies to the weighted, unrooted case, with the resulting complexity matching that in Theorem 2.2.

4. Upper bounds on buffers

Suppose there are upper bounds on buffer sizes on individual arcs. Denote these constraints by $x_{ij} + w_{ij} \leqslant u_{ij}$ for each arc $(i, j) \in E$, where $w_{ij} \leqslant u_{ij}$ is assumed. For simplicity, we assume that the problem is rooted. By Proposition 2.1, the problem is equivalent to

$$\min\left\{\sum_{(i,j)\in E} q_{ij}x_{ij} : d_i = d_j + w_{ij} + x_{ij},\ x_{ij} \geqslant 0 \text{ integer},\ x_{ij} + w_{ij} \leqslant u_{ij}\right\}. \tag{4.1}$$

As in Section 2, eliminate the x_{ij} variables from (4.1) by substituting $x_{ij} = d_i - d_j - w_{ij}$. The problem becomes

$$\min\left\{\sum_{i\in V}\rho_i d_i : w_{ij} \leqslant d_i - d_j \leqslant u_{ij},\ d_i \text{ integer}\right\}. \tag{4.2}$$

The dual problem to (4.2) is

$$\begin{aligned}\max\quad & \sum_{(i,j)\in E}(w_{ij}y_{ij} - u_{ij}z_{ij}),\\ & \sum_{\{i\,|\,(k,i)\in E\}}(y_{ki} - z_{ki}) - \sum_{\{j\,|\,(j,k)\in E\}}(y_{jk} - z_{jk}) = \rho_k, \quad k\in V, \\ & z_{ij}, y_{ij} \geqslant 0 \text{ integer.}\end{aligned} \tag{4.3}$$

Extend the original network by adding for each arc $(i,j)\in E$ whose length is w_{ij} a reverse arc (j,i) with length $w_{ji} = -u_{ij}$. Call the extended set of arcs E'. The new network has no upper bounds. Define $f_{ij} = y_{ij}, f_{ji} = z_{ij}$. Then (4.3) is equivalent to

$$\begin{aligned}\min\quad & \sum_{(i,j)\in E'} w_{ij}f_{ij},\\ & \sum_{\{i\,|\,(k,i)\in E'\}} f_{ki} - \sum_{\{j\,|\,(j,k)\in E'\}} f_{jk} = \rho_k, \quad k\in V, \\ & f_{ij} \geqslant 0 \text{ integer, } (i,j)\in E'.\end{aligned} \tag{4.3'}$$

In other words, (4.3′) is an uncapacitated maximum cost flow problem. Let us remark that the transformation above implies $w_{ij} + w_{ji} \leqslant 0$ for all arcs $(i,j)\in E'$. Hence, we can restrict the search for an optimal solution to solutions in which no two antiparallel arcs carry positive flows.

To formulate (4.3′) as a minimum cost flow problem, simply reverse the signs of the costs. The resulting problem can be solved by uncapacitated minimum cost flow algorithms. Note that the resulting problem contains a negative cost cycle if and only if the problem is unbounded, which implies that the original problem is infeasible. The solution to the primal problem can be obtained from the dual solution using the same method as in Section 2. In summary, our conclusion is as follows.

Proposition 4.1. *The weighted buffer assignment problem with upper bounds is solvable in* $O(n\log n(m + n\log n))$ *steps. The unweighted problem with upper bounds is solvable in* $O(mn\log n)$ *steps.*

Finally, suppose we only wish to determine if there exists an integer solution subject to the upper bounds. In that case, setting $q \equiv 0$, we get that problem (4.1) is feasible if and only if the corresponding dual (4.3′) with $\rho = 0$ has a bounded solution. But this is true if and only if the extended network $G(V, E')$ with costs $-w$ contains no negative cost cycle. Determining if such a cycle exists can be done, for example, using shortest path algorithms which can handle negative arc costs (cf. [2, Section 5.5]) in $O(mn)$ steps.

Corollary 4.2. *Deciding the feasibility of a buffer assignment problem with upper bounds can be done in* $O(mn)$ *steps.*

5. Minimization of the maximum buffer size

Let us now consider the buffer assignment problem, in which, instead of the sum of the buffers, the *maximum buffer size* has to be minimized. In fact, we shall address a more general question. Let $F \subseteq E$ be any subset of the arcs. The problem requires balancing G by nonnegative integer buffers, such that the maximum buffer size on the arcs in F must be as small as possible. By Proposition 2.1, a straightforward integer programming formulation of this problem is the following:

$$\begin{aligned} \min \quad & z, \\ \text{s.t.} \quad & d_i = d_j + w_{ij} + x_{ij} \quad \text{for } (i,j) \in E, \\ & x_{ij} \leqslant z \qquad\qquad\quad \text{for } (i,j) \in F, \\ & x_{ij} \geqslant 0 \text{ integer}. \end{aligned} \tag{5.1}$$

The problem can be solved using the results of Section 4, by applying binary search to the size of the upper bound on the buffer size, as follows. Define

$$X(\alpha) = \{x \in \mathbb{R}^m \mid \exists d \in \mathbb{R}^n \text{ s.t. } d_i - d_j = w_{ij} + x_{ij},\ x_{ij} \leqslant \alpha,\ (i,j) \in F,\ x_{ij} \geqslant 0 \text{ integer}\}.$$

Clearly, $X(\alpha) \neq \emptyset$ if and only if there is a balancing buffer assignment in which the maximum buffer size does not exceed α. We assume that $X(0) = \emptyset$, since otherwise the initial network is balanced without adding any buffers. Let U be the length of the longest path in the network. Clearly, $X(U) \neq \emptyset$, i.e. U is an upper bound on the min-max solution. Define $k = \lceil \log_2 U \rceil$. The procedure is as follows:

algorithm Bin Search Minmax;
begin
Step 1. $u \leftarrow 2^k$, $l \leftarrow 0$.
Step 2. **if** $u - l = 1$ **then** output u and stop.
Step 3. Set $\alpha \leftarrow (u + l)/2$; check whether $X(\alpha)$ is empty.
Step 4. **if** $X(\alpha) = \emptyset$ **then** $l \leftarrow \alpha$ **else** $u \leftarrow \alpha$ **endif.**
go to Step 2.
end

Proposition 5.1. *The above algorithm solves the min-max buffer assignment problem in* $O(mn \log U)$ *steps.*

Proof. Since 2^k and 0 are upper and lower bounds, respectively, on the solution value, validity follows by standard binary search arguments. The number of iterations is clearly $k = \lceil \log_2 U \rceil$, where each iteration requires determining the feasibility of a buffer assignment problem with upper bounds. Hence, the complexity follows from Proposition 4.2. □

Since $U \leqslant n \max_{(i,j)\in E} w_{ij}$, the algorithm is polynomial but not strongly polynomial. We now describe a different, strongly polynomial algorithm for the problem. In the process, we shall prove a strong proximity result between fractional and integer solutions of the min-max balancing problem.

Omitting integrality, (5.1) becomes a linear programming problem, which can be solved in polynomial time. However, the optimal solution of this linear program may be nonintegral. We now show how to get an optimal integral solution to (5.1) from the fractional one. In fact, we show that, whenever a fractional balanced solution exists, there is also an integer balanced solution in which every integer buffer differs by less than one unit from the corresponding fractional buffer. The proof uses an argument similar to the one used in [3].

Lemma 5.2. *If x and d are vectors satisfying the equations*

$$\begin{aligned} &d_i = d_j + w_{ij} + x_{ij}, \quad (i,j)\in E, \\ &x_{ij} \geqslant 0, \qquad\qquad (i,j)\in E, \end{aligned} \tag{5.2}$$

where w_{ij} are integers, then the integer vectors $d'_i = \lceil d_i \rceil$, $i \in V$, and $x'_i = d'_i - d'_j - w_{ij}$, $(i,j)\in E$, satisfy the same equations.

Proof. Rewrite (5.2) as $d_i - d_j \geqslant w_{ij}$, $(i,j)\in E$. Since for every real $r, s, \lceil r \rceil - \lceil s \rceil \geqslant \lfloor r - s \rfloor$, we get from the integrality of w_{ij},

$$d'_i - d'_j = \lceil d_i \rceil - \lceil d_j \rceil \geqslant \lfloor d_i - d_j \rfloor \geqslant w_{ij}.$$

Together with $x'_{ij} = d'_i - d'_j - w_{ij} \geqslant 0$, $(i,j)\in E$, this gives a feasible integer solution to (5.2). □

Note that the proof also implies that the integer buffers satisfy $\lfloor x_{ij} \rfloor \leqslant x'_{ij} \leqslant \lceil x_{ij} \rceil$ for all $(i,j)\in E$. Note also that in the proof we have not used the fact that the graph is acyclic or the fact that $w \geqslant 0$ and $x \geqslant 0$, so the theorem holds under those more general conditions.

Corollary 5.3. *If x, d, z^* are an optimal solution to the linear relaxation of* (5.1), *then $d'_i = \lceil d_i \rceil$, $i \in V$, $x'_{ij} = d'_i - d'_j - w_{ij}$, $(i,j)\in E$, and $z' = \lceil z^* \rceil$ form an optimal integer solution to* (5.1).

Proof. Clearly $d_t = 0$ can be assumed, since all d_i are determined up to an additive constant. By Lemma 5.2, x' and d' are a feasible solution to the subsystem (5.2) and therefore also $x'_i \leqslant \lceil x_{ij} \rceil \leqslant \lceil z^* \rceil$ for every $(i,j)\in F$. Since $\lceil z^* \rceil$ is a lower bound on the optimal solution value, the result follows. □

The linear program relaxation of (5.1) can also be solved in strongly polynomial time, using recent algorithms for the *minimum cost to time ratio problem.* In that problem a directed graph $D = (V, H)$ is given, and associated with each arc e are an arbitrary cost c_e and a nonnegative integer *transit time* t_e. The goal is to find a cycle

C minimizing the cost to time ratio $\lambda(C) = \sum_{e\in C} c_e / \sum_{e\in C} t_e$. Let A be the vertex–arc incidence matrix of D. A linear programming formulation of the problem (see [6]) is

$$\begin{aligned} \min \quad & \sum_{e\in H} c_e x_e, \\ \text{s.t.} \quad & \sum_{e\in H} t_e x_e = 1 \\ & Ax = 0, \\ & x \geqslant 0, \end{aligned} \tag{5.3}$$

whose dual is

$$\begin{aligned} \max \quad & \lambda, \\ \text{s.t.} \quad & \lambda t_{(i,j)} - \pi_i + \pi_j \leqslant c_{(i,j)}, \quad (i,j)\in H. \end{aligned} \tag{5.4}$$

Going back to the linear programming relaxation of (5.1), by substituting $\lambda = -z$, the problem can be reformulated as

$$\begin{aligned} \max \quad & \lambda, \\ \text{s.t.} \quad & -d_i + d_j \leqslant -w_{ij} \quad \text{for } (i,j)\in E, \\ & \lambda + d_i - d_j \leqslant w_{ij} \quad \text{for } (i,j)\in F. \end{aligned} \tag{5.1'}$$

Define a graph $\hat{G} = (V, E \cup \hat{F})$ by adding to the graph G of the min–max balancing problem additional arcs $\hat{F} = \{(j,i) \mid (i,j)\in F\}$, with costs $c_{ij} = -w_{ij}$ for $(i,j)\in E$ and $c_{ij} = w_{ji}$ for $(i,j)\in \hat{F}$, and transit times $t_e = 0$ for $e\in E$ and $t_e = 1$ for $e\in \hat{F}$. The dual of the minimum cost to time ratio for this problem is exactly the problem (5.1'). The optimal value λ^* and the quantities d_v for $v\in V$ can be obtained from the algorithm of Young, Orlin and Tarjan [23] in $O(mn + n^2 \log n)$ steps. Since G is an acyclic graph the faster algorithm of Hartmann and Orlin [13] can be used with a total complexity of $O(mn)$ steps. (The algorithm of [13] can also be modified to take advantage of the fact that it suffices to compute $\lceil z^* \rceil$ and a vector d which satisfies the constraints of (5.1) for $\lceil z^* \rceil$ to allow for earlier termination.) Together with Corollary 5.3 we can therefore conclude as follows.

Theorem 5.4. *The min–max buffer assignment problem can be solved in* $O(mn)$ *steps.*

For networks with smaller arc lengths, the binary search algorithm may be faster in practice. Similar transformations to that used to obtain (5.1') from (5.1) are given by Orlin and Rothblum [21] in the context of matrix scaling.

We complete this section with a comment on the problem of minimizing the weighted maximum buffer: determine $\min \max\{q_{ij} x_{ij}\}$, subject to the balancing constraints, where $q_{ij} \geqslant 0$. The problem can be formulated in a similar fashion to (5.1), where individual upper bounds $q_{ij}^{-1} z$ replace the uniform upper bound z in the set of inequalities for F. Hence, the binary search algorithm can be used (with the upper bound $U = n \max_{(i,j)\in E} w_{ij} \max_{(i,j)\in E} q_{ij}$) and the problem is polynomial. For

a strongly polynomial algorithm, the algorithm of Megiddo [17] solves the fractional version of (5.1) – with z replaced by $q_{ij}^{-1}z$ – in $O(mn^2 \log n)$. However, the rounded solution provided by Lemma 5.2 is not necessarily optimal. Obtaining a strongly polynomial algorithm for the weighted min–max problem is thus an open problem.

6. Minimizing the number of buffered arcs

In this section we discuss the balancing problem in which the number of arcs with positive buffers is to be minimized. We shall show that the decision version of this problem is NP-complete. The decision problem can be stated as follows.

MINIMUM ARC-COST BALANCING.

Instance: An acyclic digraph $G = (V, E)$, weights $w_{ij} \geqslant 0$ for $(i, j) \in E$, and a positive integer $B \leqslant |E|$.

Question: Is there a time-consistent weighting $\bar{w} \geqslant w$ of G with at most B positive buffers (arcs e with $\bar{w}_e > w_e$)?

We have said that a weighting $w \geqslant 0$ of an acyclic digraph $G = (V, E)$ is time consistent if there is an assignment of real numbers t to the vertices such that for every directed path P from v to u, $t_v - t_u = \sum_{(i,j)\in P} w_{ij}$. After defining some new notation, we give an alternative characterization.

For a subset $U \subseteq V$, let $\delta^+(U) = \{(u, v): u \in U, v \notin U\}$ denote the set of arcs leaving the set U, and analogously, let $\delta^-(U) = \{(u, v): u \notin U, v \in U\}$ denote the set of arcs entering U. The set $\delta^+(U)$ is called a *directed cutset* if $\delta^-(U) = \emptyset$. A nonnegative vector $x = (x_e \mid e \in E)$ is a *circulation* for a digraph $G = (V, E)$ if $\sum_{e\in\delta^+(U)} x_e = \sum_{e\in\delta^-(U)} x_e$ for all subsets $U \subset V$, or equivalently, if x can be expressed as a nonnegative linear combination of incidence vectors of directed cycles (see, e.g., [2, Theorem 3.5]).

If $G = (V, E)$ is a planar digraph, then a *planar dual* $G^* = (V^*, E^*)$ of G is a planar digraph formed by associating vertices of G^* with faces of a planar embedding of G. For each arc $e \in E$, there is a *dual arc* $e^* \in E^*$ which intuitively is obtained by rotating e counterclockwise until it is incident to the two vertices of G^* corresponding to the faces of G which were previously separated by e. We will make use of the fact that directed cutsets and directed cycles are exchanged under this geometric planar duality (see, e.g. [16, Theorem 2.8.1]).

Lemma. *A weighting $w \geqslant 0$ of an acyclic digraph $G = (V, E)$ is time consistent if and only if it can be expressed as a nonnegative linear combination of incidence vectors of directed cutsets.*

Proof. First suppose that $w \geqslant 0$ is a time-consistent weighting for G and that $t_1 \geqslant t_2 \geqslant \cdots \geqslant t_n$ are the certifying vertex numbers. Let $d_k = t_k - t_{k+1}$ for $k = 1, \ldots, n-1$ and $U_k = \{1, 2, \ldots, k\}$ for $k \in S = \{k: d_k > 0\}$. If $(i, j) \in \delta^-(U_k)$, then $t_j > t_i = w_{ij} + t_j$, contradicting the fact that $w_{ij} \geqslant 0$. Thus $\delta^+(U_k)$ is a directed cutset

for every $k \in S$. Since an arc (i, j) belongs to $\delta^+(U_k)$ if and only if $i \leqslant k < j$ and

$$w_{ij} = t_i - t_j = \sum_{k=i}^{j-1} d_k = \sum_{k \in S,\, i \leqslant k < j} d_k,$$

the vector w is the linear combination of the incidence vectors of $\delta^+(U_k)$ for $k \in S$ with corresponding multipliers $d_k > 0$.

Conversely, suppose that $w \geqslant 0$ is a nonnegative linear combination of incidence vectors of directed cutsets $\delta^+(V_k)$ for $k \in T$, and let $\mu_k \geqslant 0$ be the corresponding multipliers. For $j \in V$, let $T_j = \{k \in T : j \in V_k\}$. Since $\delta^+(V_k)$ is a directed cutset, $\delta^-(V_k) = \emptyset$ for $k \in T$, which implies that $T_j \subseteq T_i$ for every $(i, j) \in E$. Let $t_j = \sum_{k \in T_j} \mu_k$. Then, since

$$\begin{aligned} w_{ij} &= \sum_{k \in T,\, (i,j) \in \delta^+(V_k)} \mu_k = \sum_{k \in T,\, i \in V_k,\, j \notin V_k} \mu_k \\ &= \sum_{k \in T_i \setminus T_j} \mu_k = \sum_{k \in T_i} \mu_k - \sum_{k \in T_j} \mu_k = t_i - t_j, \end{aligned}$$

w is a time-consistent weighting for G. □

Note that we have not used the fact that G is acyclic in the proof, so the characterization holds for arbitrary digraphs. However, when $w \geqslant 0$ this is the only interesting case.

Corollary. *Let $G = (V, E)$ be a planar digraph and let G^* be a planar dual of G. For a weighting $w \geqslant 0$ of G define edge weights w^* on G^* by $w^*_{e^*} = w_e$ for the dual arc e^* of each $e \in E$. Then w is time consistent for G if and only if w^* is a circulation for G^*.*

We will use this to show that minimizing the number of positive buffers is NP-hard.

Theorem. MINIMUM ARC-COST BALANCING is NP-*complete, even when restricted to planar digraphs.*

Proof. Given certifying vertex numbers t for $\bar{w}$, we can easily verify that $t_v - t_u = \bar{w}_{vu}$ for all $(v, u) \in E$, so MINIMUM ARC-COST BALANCING is in NP. To show that it is NP-hard, we give a reduction from STEINER TREE IN GRAPHS, which is known to be NP-hard for planar graphs [11]. Given an undirected graph $G = (V, E)$, a subset $R \subseteq V$ and a positive integer $K \leqslant |V| - 1$, is there a subtree of G that includes all the vertices of R and contains no more than K edges? Since this problem concerns undirected graphs, we first transform it to the equivalent problem of finding a minimum cardinality feasible circulation in a related digraph.

Given an instance of STEINER TREE IN GRAPHS for which $G = (V, E)$ is planar, construct a planar digraph $D = (V, A)$ by replacing each edge $\{u, v\} \in E$ by a pair of arcs (u, v) and (v, u) in A. Note that this ensures that there are no subsets $W \subset V$ with $\delta^-(W) = \emptyset$ in D. Let s be any vertex in R, and find a feasible solution x to the transshipment problem in D with demands $b_v = -1$ for $v \in R \setminus \{s\}$, $b_s = |R| - 1$ and $b_v = 0$ for $v \notin R$. If $\bar{x}$ is a circulation for D satisfying $\bar{x} \geqslant x$, then $\bar{x} - x$ is a feasible

solution to the transshipment problem in D with demands $b_v = +1$ for $v \in R \setminus \{s\}$, $b_s = -(|R| - 1)$ and $b_v = 0$ for $v \notin R$ and so identifies a directed tree rooted at s containing each $v \in R$. Intuitively, finding a Steiner tree reduces to finding a circulation $\bar{x}$ in D satisfying $\bar{x} \geqslant x$ for which the number of arcs $a \in A$ with $\bar{x}_a > x_a$ is as small as possible.

Now let $D^* = (V^*, A^*)$ be a planar dual of D. Since directed cycles and directed cutsets are exchanged under planar duality, D^* will be acyclic. Define $w_{a^*} = x_a$ for the dual arc a^* which corresponds to each arc $a \in A$. Let $\bar{w} \geqslant w$ be a time-consistent weighting for D^* with B positive buffers. By the corollary, $\bar{x} \geqslant x$, defined by $\bar{x}_a = \bar{w}_{a^*}$ for the arc a corresponding to each dual arc $a^* \in A^*$, is a circulation in D. Clearly the set of arcs a with $\bar{x}_a > x_a$ contains a subtree of G that includes all the vertices of R and has no more than B edges.

Conversely, let T be a subtree of G which includes all the vertices of R and contains B edges. By assigning directions to the edges of T, we can find a directed tree in D rooted at s with B arcs and a corresponding feasible solution $y \geqslant 0$ to the transshipment problem in D with demands $b_v = 1$ for $v \in R \setminus \{s\}$, $b_s = -(|R| - 1)$, and $b_v = 0$ for $v \notin R$. It follows that $\bar{x} = x + y$ is a circulation and thus by the corollary the corresponding weighting $\bar{w} \geqslant w$ for D^* is time consistent and has B positive buffers.

Since a planar dual of a directed planar graph can be found in polynomial time, the above reduction from STEINER TREE IN GRAPHS is a polynomial time reduction, proving thus the theorem. □

7. Concluding remarks

We have given efficient solution procedures for several balancing problems. Three of them were generalizations of the min-sum buffer assignment problem: weighted problems, unrooted problems and capacitated problems. The fourth one was the min–max buffer assignment problem. We have shown an equivalence between the weighted min-sum problem and the transshipment problem on acyclic graphs. We have also presented a tight proximity result for the distance between a fractional and an integer solution for balancing problems. On the other hand, we have shown that the problem of balancing a network while minimizing the number of arcs containing positive buffers is NP-hard.

Interestingly, similar balancing problems arise in the area of project management, better known as PERT/CPM: The Critical Path Method (CPM) has been used since the early sixties as a tool for planning and scheduling projects (see, e.g., [14, 18]). In CPM, a graph models the project where vertices correspond to events and arcs correspond to activities. The graph is acyclic with both an origin and an output vertex. An activity cannot start before all the preceding events have been completed. There are lower and upper bounds on the possible duration of each activity, and a non-decreasing linear utility function is assigned to the duration of each activity. The goal is to find durations for the activities and consistent times for the events so that the total utility is maximized.

Once an assignment of durations to activities has been determined, there is still flexibility in terms of timing the events. More precisely, the starting (or ending) time of

any activity which is not on one of the longest paths from the origin to the output vertex is not uniquely determined. The extra time available from the time point an activity is complete to the succeeding event is called the *float* on that activity. Hence, the "floats" in PERT are exactly the "buffers" in the language we use here, and minimizing the sum of buffer lengths corresponds to minimizing the total amount of idle times in a project. The weighted buffer assignment and the min–max problem also have obvious interpretations in the context of project management. Minimizing the sum of the buffers corresponds to making the schedule of the project as tight as possible, to save idle time costs. In other situations it may be desirable to maximize the total idle time (without increasing the total project length), in order to account for unforeseeable delays and to increase flexibility during the operation. This problem can also be dealt with by the techniques described in Section 2. In fact, its dual is immediately a minimization problem, so the transformation of costs in steps 1 and 2 of the algorithm of Section 2 is unnecessary.

One question about balancing which is still open is whether one can solve the weighted version of the min–max problem in strongly polynomial time. Another possible research direction is generalizing the study of balancing problems to other algebraic structures, e.g., by replacing the sum and max operations by more abstract and more general algebraic operations (cf. [24]).

Acknowledgement

We thank Bruno Simeone for raising the question of handling capacities, at the Viewpoint on Optimization meeting in Grimentz in September 1990, and Rainer Burkard for the idea of algebraic generalizations. We thank Maurice Queyranne for pointing out to us a connection between the buffer assignment problem and the PERT/CPM model.

References

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, Network flows, in: G.L. Nemhauser, A.H.G. Rinnooy Kan and M.S. Todd, eds., Handbooks in Operations Research and Management Science, Vol. I (Elsevier, Amsterdam, 1989) 211–369.

[2] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, Network Flows: Theory, Algorithms and Applications (Prentice Hall, Englewood Cliffs, NJ, 1992).

[3] J.J. Bartholdi, J.B. Orlin and H.D. Ratliff, Cyclic scheduling via integer programs with circular ones, Oper. Res. 28 (1980) 1073–1085.

[4] E. Boros, P.L. Hammer and R. Shamir, A polynomial algorithm for balancing acyclic data flow graphs, IEEE Trans. Comput. 41 (1992) 1380–1385.

[5] P.R. Chang and C.S.G. Lee, A decomposition approach for balancing large-scale acyclic data flow graphs, IEEE Trans. Comput. 39 (1990) 34–46.

[6] G.B. Dantzig, W. Blattner and M.R. Rao, Finding and cycle in a graph with minimum cost to time ratio with application to a ship routing problem, in: P. Rosensthiehl, ed., Theory of Graphs (Dunod, Paris and Gordon and Breach, New York, 1967) 77–84.

[7] J.B. Dennis, Data flow supercomputers, IEEE Comput. (Nov. 1980) 48–56.

[8] J. Edmonds and R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, J. ACM 19 (1972) 248–264.

[9] A.L. Fisher and S.Y. Kung, Special-purpose VLSI architectures: general description and a case study, in: Kung, Whitehouse and Kailath, eds., VLSI and Modern Signal Processing (Prentice Hall, Englewood Cliffs, NJ, 1985) 153–169.
[10] M.L. Fredman and R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, J. ACM 34 (1987) 596–615.
[11] M.R. Garey and D.S. Johnson, The rectilinear Steiner tree problem is NP-complete, SIAM J. Appl. Math. 32 (1977) 826–834.
[12] A.V. Goldberg, É. Tardos and R.E. Tarjan, Network flow algorithms, in: B. Korte, L. Lovasz, H.J. Prömel and A. Schrijver, eds., Paths, Flows and VLSI-Layout (Springer, Berlin, 1990) 101–164.
[13] M. Hartmann and J.B. Orlin, Finding minimum cost to time ratio cycles with small integral transit times, Tech. Rept. No. UNC/TR/91-19, University of North Carolina at Chapel Hill (1991).
[14] J.E. Kelley Jr, Critical-path planning and scheduling: mathematical basis, Oper. Res. 9 (1961) 296–320.
[15] S.Y. Kung, Why systolic architectures?, IEEE Comput. (Jan. 1982) 37–46.
[16] E. Lawler, Combinatorial Optimization: Networks and Matroids (Holt, Rinehart and Winston, New York, 1976).
[17] N. Megiddo, Combinatorial optimization with rational objective functions, Math. Oper. Res. 3 (1979) 313–323.
[18] J.J. Moder, C.R. Phillips and E.W. Davis, Project Management with CPM, PERT and Precedence Diagrams, (Van Nostrand Reinhold, New York, 3rd ed., 1983).
[19] G.L. Nemhauser and L.A. Wolsey, Integer and Combinatorial Optimization (Wiley, New York, 1988).
[20] J.B. Orlin, A faster strongly polynomial minimum cost flow algorithm, in: Proceedings 20th ACM Symposium on Theory of Computing (1988) 377–387; revised version: Sloan W.P. No. 3060-89-MS, Sloan School of Management, M.I.T. (1989).
[21] J.B. Orlin and U.G. Rothblum, Computing optimal scaling by parametric network algorithms, Math. Programming 32 (1985) 1–10.
[22] R.E. Tarjan, Data Structures and Network Algorithms (Society for Industrial and Applied Mathematics, Philadelphia, PA, 1983).
[23] N.E. Young, R.E. Tarjan and J.B. Orlin, Faster parametric shortest path and minimum balance algorithms, Networks 21 (1991) 205–221.
[24] U. Zimmermann, Linear and Combinatorial Optimization in Ordered Algebraic Structures, Annals of Discrete Mathematics 10 (North-Holland, Amsterdam, 1981).