Balanced Network Flows. VIII. A Revised Theory of Phase-Ordered Algorithms and the $O(\sqrt{n}m \log(n^2/m)/\log n)$ Bound for the Nonbipartite Cardinality Matching Problem

Christian Fremuth-Paeger, Dieter Jungnickel
Lehrstuhl für Diskrete Mathematik, Optimierung und Operations Research, University of Augsburg, D-86135 Augsburg, Germany

This paper closes some gaps in the discussion of nonweighted balanced network flow problems. These gaps all concern the *phase-ordered* augmentation algorithm, which can be viewed as the matching counterpart of the Dinic max-flow algorithm. We show that this algorithm runs in $O(n^2m)$ time compared to the bound of $O(nm^2)$ derived in Part III of this series and that the bipartiteness requirement can be omitted. The result which deserves attention is the complexity bound for the cardinality matching problem which was shown for the bipartite case by Feder and Motwani. This bound was previously claimed for the general case in a preprint of Goldberg and Karzanov but omitted in later versions of this paper. © 2003 Wiley Periodicals, Inc.

Keywords: cardinality matching problem; network flows; skew-symmetric graphs; antisymmetrical digraphs; graph compression; clique partition

PRELIMINARIES

In Parts I–V of this series [3–7], balanced networks were treated as bipartite graphs, that is, it was required that the node set of a balanced network N splits into two sets $Inner(N)$ and $Outer(N)$ which are independent. In addition, we required for every node $v \in V(N)$ that $v' \in Inner(N)$ if and only if $v \in Outer(N)$.

In Part VI [8], it was shown that this bipartiteness assumption is immaterial at least for the polyhedral characterization of balanced circulations. A careful inspection of Part I [3] reveals that the general theory of BNS algorithms is independent of this bipartiteness assumption. However, the computation of distance labels in nonbipartite networks requires some extra statements which we must supply.

By this extension, balanced network flows become compatible with the idea of graph compression introduced by Feder and Motwani [2], who proved the $O(\sqrt{n}m \log(n^2/m)/\log n)$ bound for the bipartite case of the cardinality matching problem (CMP). We will show how this bound can be reached in the general case. This result heavily depends on the network flow formulation of the CMP since one does not compress the graph but, rather, the transformed balanced flow network. It seems complicated to combine the Micali/Vazirani algorithm [12] and the clique partition algorithm directly in terms of a matching problem.

Even more, we can show the $O(n^2m)$ complexity of the maximum balanced st-flow algorithm which was formulated in Part III [5] and called **phase-ordered.** This algorithm essentially is a combination of the Dinic max-flow method and the Micali/Vazirani method for the CMP. Hence, the results of this paper all indicate that nonweighted balanced network flow (matching) problems are just as hard as are ordinary network flow (bipartite matching) problems.

The article of Feder and Motwani was brought to our attention by Loehnertz [11], who described the application of the graph compression technique to the search procedures of the Blum cardinality matching algorithm [1] and who may claim the improved complexity bound for cardinality matching. We mention that the general idea can already be found in Goldberg and Karzanov [9] without evident proof, and in a later version [10], the corresponding statement is omitted. Unlike [11], we compress balanced flow networks explicitly and then show, with little effort, that the phase-ordered augmentation algorithm [5] has the claimed complexity.

41. DISTANCE LABELS IN NONBIPARTITE GRAPHS

We can find a maximum balanced st-flow by successive balanced augmentation where each augmenting path is a shortest valid st-path, which we refer to as a $d(t)$-path. The

Received January 2002; accepted January 2003
Correspondence to: D. Jungnickel; e-mail: jungnickel@math.uni-augsburg.de

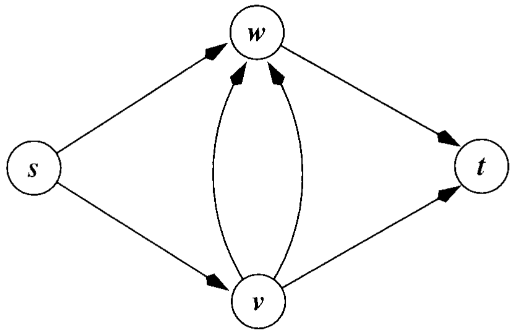

FIG. 1. A nonbipartite balanced flow network.

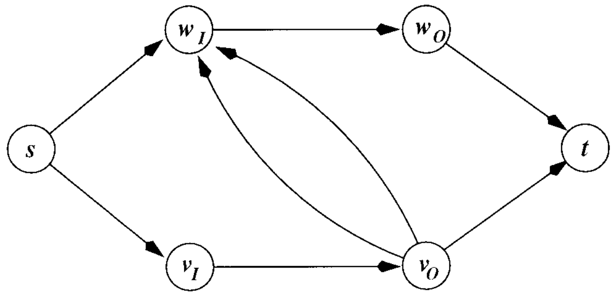

FIG. 2. The transformed balanced flow network.

determination of shortest paths for augmentation has similar advantages as in the case of ordinary maximum flow problems.

Again, let $d(v)$ denote the minimum length of a valid sv-path. Recall that the **tenacity labels** were defined in [3] by

$$t(v) := d(v) + d(v')$$

and, respectively,

$$t(uv) := d(u) + d(v') + 1.$$

The definition of distance labels and tenacity labels can be extended to nonbipartite networks without any modification. We now call every node v with $d(v) \leq d(v')$ a **minlevel node.** This is compatible with the bipartite case and the former definition where we required $d(v) < d(v')$. If we have $d(v) = d(v')$, we call v a **pseudo-petal.**

Consider the balanced flow network N given in Figure 1 with $s' = t$, $v' = w$, $cap(sv) = 3$, $cap(sw) = 1$ and $cap(vw) = 2$. In this example, the successive shortest path algorithm would first augment along the paths $p = (s, v, t)$, p' and then augment along the paths $q = (s, v, w, t)$, q' to achieve a maximum balanced flow.

Note that we have $d(v) = d(w) = 1$ in the first iteration. The BNS algorithm as stated in [3] would have to choose either v or $w = v'$ as a minlevel node. We ask if there is a special tie-breaking rule which makes this algorithm applicable to nonbipartite networks.

We next show how a given balanced network N can be transformed into a bipartite network N^{bip} by splitting the nodes. It turns out that N^{bip} and N are equivalent in a very strong sense.

Let v, $w = v'$ be a complementary node pair. Replace v by nodes v_I, v_O and w by nodes w_I, w_O. Add new arcs v_Iv_O and w_Iw_O with infinite capacity. Put $v_I' = w_O$ and $w_I' = v_O$. Replace the former arcs xy by arcs x_Oy_I without changing the capacity (see Fig. 2).

First notice that this transformation preserves the skew-symmetry and that N^{bip} is indeed bipartite. Furthermore, a valid uv-path p in N corresponds to a valid u_Ov_O-path q in N^{bip} with $|q| = 2|p|$ and vice versa. This implies that props in N map to props in N^{bip} and that bridges in N map to bridges in N^{bip}.

If we denote the distance labels in N^{bip} by $\bar{d}$ and the tenacity labels by $\bar{t}$ and, again, put $w := v'$, we can observe that

$$\begin{aligned}\bar{t}(v_I) &= \bar{d}(v_I) + \bar{d}(w_O)\\ &= 2d(v) - 1 + 2d(w) = 2t(v) - 1 = \bar{t}(v_O)\end{aligned}$$

and

$$\begin{aligned}\bar{t}(u_Ov_I) &= \bar{d}(u_O) + \bar{d}(v_O) + 1\\ &= 2d(u) + 2d(v) + 1 = 2t(uv) - 1.\end{aligned}$$

Apparently, we either have $\bar{d}(v_I) = \bar{d}(v_O) = \infty$ or $\bar{d}(v_O) = \bar{d}(v_I) + 1$. If we have $d(v) = d(w) = i$, then $\bar{d}(v_I) = \bar{d}(w_I) = 2i - 1$ and $\bar{d}(v_O) = \bar{d}(w_O) = 2i$, so that v_Iv_O and w_Iw_O are petals with

$$\bar{t}(v_Iv_O) = \bar{d}(v_I) + \bar{d}(w_I) + 1 = 4i - 1.$$

Otherwise, if $d(v) < d(w)$, then $\bar{d}(v_O) = \bar{d}(v_I) + 1 < \bar{d}(w_I)$ for reasons of parity. In that case, v_Iv_O is a prop.

As in [3] for the bipartite case, we call the props uv with $d(v) \leq i$ the (i)-props and the bridges xy with $t(xy) \leq 2i$ the (i)-bridges. The (i)-props together with their complements and with the (i)-bridges form the network N_i.

We already have shown that the (i)-props in N map to $(2i)$-props in N^{bip} and that the (i)-bridges in N map to $(2i)$-bridges in N^{bip}. Hence, we have

Lemma 41.1. $(N_i)^{bip} = (N^{bip})_{2i}$.

Now, we are in a comfortable situation. If we apply Theorem 12.1 in [3] to the network N^{bip} and a node pair v, v' with $d(v) < d(v')$, we obtain

$$\bar{d}_{2i}(v) = \begin{Bmatrix} \bar{d}(v), & \text{if } \bar{d}(v) \leq 2i \\ \infty, & \text{otherwise} \end{Bmatrix},$$

$$\bar{d}_{2i}(v') = \begin{cases} \bar{d}(v'), & \text{if } \bar{t}(v) \le 4i - 1 \\ \infty, & \text{otherwise} \end{cases}.$$

The translation from N^{bip} to N is straightforward and yields the following extension of Theorem 12.1 in [3] to nonbipartite networks:

Theorem 41.2. *Let* $d(v) \le d(v')$. *Then, the following assertions hold*:

$$d_i(v) = \begin{cases} d(v), & \text{if } d(v) \le i \\ \infty, & \text{otherwise} \end{cases}$$

$$d_i(v') = \begin{cases} d(v'), & \text{if } t(v) \le 2i \\ \infty, & \text{otherwise} \end{cases}.$$

The BNS algorithm given in [3] for bipartite problems starts with the network N_0. At some stage, the network N_{i-1} is extended to N_i, maintaining a corresponding layered auxiliary network. This is done by first adding the props uv with $d(v) = i$ and then adding the bridges a with $t(a) = 2i - 1$.

In the nonbipartite situation, we have several possibilities for applying the bipartite algorithm:

(1) Perform the transformation of N into N^{bip} explicitly.
(2) Collect the pseudopetals v with $d(v) = d(v') = i$ into a queue. After adding the props and the bridges with tenacity $2i - 1$, shrink the petals with tenacity $2i$ and the collected pseudopetals in an arbitrary order.
(3) If $d(v) = i$ has been set by the algorithm before and v is recognized to be a pseudopetal with tenacity $2i$ afterwards, the props of v' are simply treated as bridges with tenacity $2i$ in what follows.

The first option results in much code and is not very attractive. The second option requires a modification of the path expansion rule, but seems more natural than option (3). Nevertheless, we prefer the tie-breaking method (3), since it requires only minor changes of the algorithm provided in Part III of this series [5].

42. AN IMPROVED GENERAL COMPLEXITY BOUND

In Part III, we proved that the phase-ordered algorithm runs in $O(nm^2)$ time. We suggested that this bound might be tight in contrast to the Dinic max-flow algorithm which only needs $O(n^2m)$ time. The key observation was that the double depth-first search procedure (DDFS) searches arcs on which the algorithm does not augment and which are not shrunk immediately.

Reading this again, we found that Lemma 22.3 in [5] can be replaced by the following statement:

Lemma 42.1. *Consider a call of the DDFS which yields* $b = s$. *Let* a *be a prop which is searched by this DDFS, but not on the augmenting paths* p, p'. *Then,* a *is blocked after that augmentation step.*

Proof. Let v be the start node and w be the end node of a in the layered auxiliary network just before the call of DDFS. The assertion follows by induction on $d(v)$. In the case of $d(v) = 0$, we have that a is on p or p' and there is nothing to show. Thus, we can assume that $d(v) = i + 1$, $i \ge 0$ and that the statement is true for every prop $\tilde{a}$ with start node $\tilde{v}$ in the layered network where $d(\tilde{v}) \le i$. We also assume that a is neither on p nor p'.

We first assume that a has been included into one of the DFS trees, say the left one. Since the left DFS has backtracked from v, all props of v have been searched by the left DFS before. Note that these arcs are neither on p nor p' and are blocked by the induction hypothesis. But, then, a is also blocked.

Otherwise, a has not been included into one of the DFS trees, but w is in one of the DFS trees, say the left one. It turns out that v has been included into the left DFS tree and the left DFS has backtracked from v before a was searched. As in the previous case, all props of v have been searched, do not appear on the augmenting paths, and are blocked by the induction hypothesis. But, then, a is blocked likewise. ■

Corollary 42.2. *If we ignore the DSU operations and the elementary operations concerning arcs of the augmenting paths, the time spent for DDFS operations during a whole phase is* $O(m)$.

The time needed for DSU operations is $O(m + n^2)$ even if a very simple data structure is used. The other operations of the phase-ordered augmentation algorithm are a topological erasure, which is the same as for the Dinic algorithm and requires $O(m)$ time per phase, and the augmentations which require $O(nm)$ time per phase and dominate the overall complexity:

Corollary 42.3. *The phase-ordered balanced augmentation algorithm as presented in* [5] *runs in* $O(n^2m)$ *time and each phase takes* $O(nm)$ *operations.*

43. NETWORKS WITH 0–1 CAPACITIES

In the case of 0–1 capacities, the augmenting paths of a single phase are edge-disjoint. It turns out that the DSU operations dominate the complexity of all operations in a single phase:

Corollary 43.1. *If* N *has* 0–1 *capacities, then a phase runs in* $O(m\alpha(m, n))$ *time.*

Here, $\alpha(m, n)$ denotes the so-called **inverse Ackerman function** which is very slow growing. For cardinality matching algorithms (and very likely in our setting), it is known that this factor can be omitted in total. We leave this an open question. If the conjecture is true, all terms of α which occur later can be omitted.

Goldberg and Karzanov [9] showed the following nontrivial bound on the number of phases:

Lemma 43.2. *If N is a balanced flow network with* 0–1 *capacities, then any phase-ordered algorithm requires at most* $\sqrt{m}$ *phases.*

Proof. Let g be a maximum balanced st-flow on N, say with value 2ν, and let $r := \lceil \sqrt{m}/2 \rceil$. Suppose that $\nu - m > 0$, and let f be the balanced st-flow of value $2(\nu - m)$ which is constructed by the algorithm. By Theorem 4.1 in [3], we can write

$$g - f \equiv \sum_{i=1}^{s} \chi_{p_i},$$

where $r \le s$, $p_1, p_2, \ldots, p_r$ are pairwise edge-disjoint valid st-paths in $N(f)$ and $p_{r+1}, p_{r+2}, \ldots, p_s$ are valid cycles in $N(f)$.

If $d(t)$ denotes the minimum length of a valid st-path in $N(f)$ and l denotes the minimum length of a path among $p_1, p_2, \ldots, p_r$, we have $2lr \le m$ and, hence, $d(t) \le l \le \sqrt{m}$. But, at most, $d(t)$ phases occur before f is constructed and, at most, r phases occur after the construction of f. This gives the desired bound. ∎

To obtain better bounds, one may define a **node capacity** for every $w \in V(N)$ by

$$ucap(w) := \min\left\{ \sum_{a^- = w} ucap(a), \sum_{a^+ = w} ucap(a) \right\}.$$

In the same way, one can define the **residual node capacity** and obtain the following:

Observation 43.3. *Let f be a feasible pseudoflow on N and w be a node with* $div_f(w) = 0$. *Then,* $ucap(w) = rescap(w)$ *holds.*

This shows that a unit capacity node can appear on an augmenting path only once during a single phase. Even more, if a flow is decomposed into cycles and st-paths in N, unit capacity nodes can appear only once on these paths.

Remember that the number of phases in the Dinic maxflow algorithm is $O(\sqrt{n})$ if all nodes have unit capacities. Goldberg and Karzanov [10] obtained an even more general bound for skew-symmetric flows:

Lemma 43.4. *Let N be a balanced flow network. Then, any phase-ordered augmentation algorithm requires at most* $\sqrt{ucap(N)}$ *phases, where*

$$ucap(N) := \sum \{ucap(v) : v \in V(N)\}.$$

Although not stated explicitly in [10], this shows that any k-factor problem (k fixed!) is solved in $O(\sqrt{n})$ phases if a phase-ordered algorithm is applied to the transformed matching instances $N_{\mathcal{M}}$.

On the other hand, this statement does not bound the number of phases if some nodes have large capacities. For this reason, we consider slightly different prerequisites:

A **strong** st**-node cut** is a node set W such that the nodes in W have unit capacity and s and t are in different connected components of $N[V(N)\backslash W]$. In a flow decomposition, any augmenting path must meet this potential cut, and, hence,

Observation 43.5. *The value of an st-flow is bounded by the minimum size of a strong st-node cut* (*if one exists*).

Lemma 43.6. *Let N be a balanced flow network with unit arc capacities which admits a strong st-node cut of size k and in which the length of a directed path is bounded from above by a constant l. Then, any phase-ordered augmentation algorithm requires at most* $(l^2 + l + 1)\sqrt{k}$ *phases.*

Proof. Let g be a maximum balanced st-flow on N, say with value 2ν, and $r := \lceil \sqrt{k} \rceil$. Suppose that $\nu - r > 0$, and let f be the balanced st-flow of value $2(\nu - r)$ which is constructed by the algorithm. Write

$$g - f \equiv \sum_{i=1}^{s} \chi_{p_i},$$

where $r \le s$, $p_1, p_2, \ldots, p_r$ are pairwise edge-disjoint, valid st-paths in $N(f)$. Note that $p_1, p'_1, p_2, p'_2, \ldots, p_r, p'_r$ can traverse at most $val(f)l = 2l(\nu - r)$ backward arcs altogether. Thus, at least one of the paths traverses at most $l/r(\nu - r)$ backward arcs. In particular, we have

$$d(t) \le \frac{l(l+1)}{r}(\nu - r) + l = \frac{l(l+1)\nu}{r} - l^2$$
$$\le l(l+1)\sqrt{k} - l^2, \quad (1)$$

since $\nu \le k$. Hence, at most, $l(l+1)\sqrt{k} - l^2$ phases occur before f is constructed and, at most, $r = \lceil \sqrt{k} \rceil \le \sqrt{k} + 1$ phases occur after the construction of f. ∎

The reader may observe that Lemmas 43.4 and 43.6 apply to the transformed 1-matching instances N_G and, hence, generalize Theorem 6.6 in [3]. Lemma 43.6 has another application which is discussed in the next section and which leads to an improvement of the Micali/Vazirani cardinality matching algorithm, at least for dense graphs.

44. GRAPH COMPRESSION

Let G be a simple graph with n nodes and m arcs. A **biclique** is a pair (U, W) with $U, W \subseteq V(G)$ so that u and w are adjacent in G for every pair $u \in U$, $w \in W$. Note that $U \cap W = \emptyset$ since no loops are allowed. A **clique partition** of G is a family $\mathcal{F} = \{(U_1, W_1), (U_2, W_2), \ldots, (U_k, W_k)\}$ of bicliques so that every edge $e \in E(G)$ is spanned by exactly one biclique. If $\prec$ denotes some ordering of $V(G)$, a trivial clique partition of G is given by

$$\{(\{u\}, \{w\}) : \{u, w\} \in E(G),\ u \prec w\}.$$

We can define a balanced flow network $N_G^{\mathcal{F}}$ as follows:

- Take a disjoint copy $V(G)'$ of $V(G)$ and define complementarity for nodes, arcs, node sets, and paths in the obvious way.
- For every biclique $(U, W) \in \mathcal{F}$, introduce a complementary pair of biclique nodes (U, W') and $(U, W')' = (W, U')$.
- Introduce a source node s and a sink node t.
- Connect s to all nodes in $V(G)$ and all nodes in $V(G)'$ to t.
- For every biclique $(U, W) \in \mathcal{F}$ and nodes $u \in U$, $w \in W$, connect u to (U, W'), (U, W') to w', w to (W, U') and (W, U') to u'.
- Put $ucap \equiv 1$ and $lcap \equiv 0$.

The number of nodes in $N_G^{\mathcal{F}}$ is $n^* := 2(n + k + 1)$, and the number of arcs is denoted by m^*. Note that every biclique (U, W) consists of $|U||W|$ edges which are mapped to $2(|U| + |W|)$ arcs in the balanced flow network, that is, m^* is $O(m)$ and n^* is $O(n + k)$.

Note that $N_G^{\mathcal{F}}$ is not bipartite, and the biclique nodes all have $O(m)$ capacities in the worst case. On the other hand, V and V' constitute strong st-cuts and a directed path in $N_G^{\mathcal{F}}$ traverses at most four arcs, that is, we can compute a maximum balanced flow on $N_G^{\mathcal{F}}$ in $O(\sqrt{n}m^*)$ time, which may be considerably less than $O(\sqrt{n^*}m^*)$ time.

It is evident that a balanced flow on $N_G^{\mathcal{F}}$ with value 2ν can be transformed into a matching of G with cardinality ν and vice versa, and this relationship is many-to-many.

Corollary 44.1. *Given any clique partition $\mathcal{F}$ of the graph G, a maximum matching of G can be determined in $O(\sqrt{n}m^*\alpha(m,n))$ time.*

We finally have to describe how the clique partition algorithm devised by Feder and Motwani [2] for bipartite graphs applies to our setting:

One deletes from N_G the source and the sink node, deletes one arc of every complementary pair, and forgets about the orientation of the arcs. The original clique partition algorithm of [2] will give a partition $\mathcal{F}$ of this reduced graph. If one adds the complementary biclique of every biclique in $\mathcal{F}$, a partition of $N_G - \{s, t\}$ results which represents every edge of G twice. This modified graph compression algorithm does not apply to the original matching problem G but rather to the balanced network N_G.

The complexity analysis works just as in the setting for bipartite matching problems. Feder and Motwani obtained the following result:

Theorem 44.2. *Given a bigraph with partitions of equal size and some parameter $\delta \in (0, 1)$, the clique partition algorithm finds a partition $\mathcal{F}$ of cardinality $O(m/n^{1-\delta})$ in time $O(mn^{\delta}\log^2 n)$. The number of edges in the compressed graph is $O[m((\log(n^2/m)/(\delta \log n))]$.*

For our purposes, we must increase the number of edges and cliques only by a factor of 2. If one takes a constant $\delta \in (0, 1/2)$, the computation of $\mathcal{F}$ does not contribute to the overall complexity of the cardinality matching algorithm.

Corollary 44.3. *The cardinality matching problem can be solved in*

$$O\left(\sqrt{n}m\,\frac{\log\dfrac{n^2}{m}}{\log n}\,\alpha(m, n)\right)$$

time.

Note that any future progress in the compression rate would improve the CMP algorithm immediately. If bipartite matching should, after all, be easier than the general CMP, this would have to involve a new algorithmic idea apart from graph compression.

One could think of a modified clique partition algorithm for symmetric bigraphs which computes a biclique of $N_G - \{s, t\}$, then immediately adds the complementary biclique and performs the corresponding clique stripping operations. The stripping of such a reverse biclique is rather expensive if the original Feder/Motwani data structure is used. We do not know how to overcome these difficulties.

45. PROJECT SUMMARY

In this paper, we discussed some open questions and new ideas regarding the phase-ordered augmentation algorithm which was described in Part III [5]. Since this is the concluding part of this series of papers about balanced network flow theory, we would like to take the opportunity and say something about our implementation and the computational experiences:

Up to the graph compression algorithm, we have implemented all methods presented throughout this series of papers. The complete code is available by the C++ open-source software GOBLIN, which also covers many further graph optimization problems. This C++ library can be downloaded from the URL:

`www.math.uni-augsburg.de/opt/goblin.html`.

The implementations have shown that the plain balanced augmentation algorithm cannot compete with the balanced augmentation algorithm which uses the depth-first BNS heuristic and verifies negative results with the exact BNS procedure. We have also tried out a breadth-first BNS heuristic which totally ignores the existence of blossoms, but which performs rather poorly. We mention that the Kameda/Munro heuristic as presented in [4] includes a serious bug but does not discuss the details. Interested readers are referred to our computer code for a corrected version.

The cycle canceling method and the phase-ordered balanced augmentation algorithm in the Micali/Vazirani tradition cannot beat each other, at least if the Dinic method is used for the max-flow computations. It is remarkable that the even cycle canceling procedure leaves only very few odd cycles.

Furthermore, both the Dinic method and the phase-ordered balanced augmentation algorithm require only few phases; indeed, the number of phases is generally much smaller than the bounds $O(\sqrt{n})$ and $O(n)$ may suggest. In the case of the balanced method, one can, in practice, save much time if one checks for maximality with a basic BNS procedure before starting the BNS procedure described in [5]. Otherwise, the bulk of the blossom shrinking operations would occur in the very last BNS, although no augmentation is possible.

In the weighted setting, one should also check for maximality before searching for a minimum-length valid augmenting path and, similarly, the main portion of blossom shrinking and expansion operations occur during the last iterations of the primal-dual method. For this reason, the enhanced PD algorithm is indeed faster but the running times do not decrease as dramatically as expected by us.

For solving matching problems on complete graphs, we used sparse candidate graphs but our price and repair procedure applies to the underlying min-cost flow solver only. Hence, the overall running times depend mainly on the odd-cycle canceling operations on the dense problems which usually occur.

It seems obvious that our primal-dual code can be developed further in analogy with the 1-matching case. But there are additional technical difficulties, at least with the expansion of blossoms (which are no longer simple cycles) and the postoptimality operations (i.e., primal feasible iterations). Concrete efficient implementations of the primal-dual (or the shortest augmenting path) method are not as obvious as suggested in [9].

We have done some performance tests for 1- and 2-factor problems. We report that nonweighted problems on 10,000 node problems are solved within 2–5 seconds and that metric problems with 1000 nodes are solved in 30 minutes on a current PC. Sparse weighted matching problems and nonmetric random instances are much easier to solve. Our code applies to the more general setting of capacitated matching problems. For the case of 1-factor problems, it is not as efficient as are the best available codes; there remains the challenge to close this gap.

REFERENCES

[1] N. Blum, A simplified realization of the Hopcroft–Karp approach to maximum matching in general graphs, Technical report, University of Bonn, 1999.

[2] T. Feder and R. Motwani, Clique partitions, graph compression and speeding up algorithms, JCSS 51 (1995), 261–272.

[3] C. Fremuth-Paeger and D. Jungnickel, Balanced network flows (I): A unifying framework for design and analysis of matching algorithms, Networks 33 (1999), 1–28.

[4] C. Fremuth-Paeger and D. Jungnickel, Balanced network flows (II): Simple augmentation algorithms, Networks 33 (1999), 29–41.

[5] C. Fremuth-Paeger and D. Jungnickel, Balanced network flows (III): Strongly polynomial augmentation algorithms, Networks 33 (1999), 43–56.

[6] C. Fremuth-Paeger and D. Jungnickel, Balanced network flows (IV): Duality and structure theory, Networks 37 (2001), 194–201.

[7] C. Fremuth-Paeger and D. Jungnickel, Balanced network flows (V): Cycle canceling algorithms, Networks 37 (2001), 202–209.

[8] C. Fremuth-Paeger and D. Jungnickel, Balanced network flows (VI): Polyhedral descriptions, Networks 37 (2001), 210–218.

[9] A. Goldberg and A.V. Karzanov, Maximum skew-symmetric flows, preprint, 1995.

[10] A. Goldberg and A.V. Karzanov, Maximum skew-symmetric flows and their applications to b-matchings, preprint, 1999.

[11] M. Loehnertz, Alternating path algorithms for non-bipartite matching using graph compression, Technical report, University of Bonn, 2001.

[12] S. Micali and V.V. Vazirani, An $O(\sqrt{V}\, E)$ algorithm for finding maximum matching in general graphs, Proc 21st Ann IEEE Symp in Foundation of Computer Science, IEEE Press, Piscataway, NJ, 1980, pp. 17–27.