Andrew V. Goldberg[1] and Alexander V. Karzanov[2]

Maximum skew-symmetric flows and matchings

December 2003

Abstract. The maximum integer skew-symmetric flow problem (MSFP) generalizes both the maximum flow and maximum matching problems. It was introduced by Tutte [28] in terms of self-conjugate flows in antisymmetrical digraphs. He showed that for these objects there are natural analogs of classical theoretical results on usual network flows, such as the flow decomposition, augmenting path, and max-flow min-cut theorems. We give unified and shorter proofs for those theoretical results.

We then extend to MSFP the shortest augmenting path method of Edmonds and Karp [7] and the blocking flow method of Dinits [4], obtaining algorithms with similar time bounds in general case. Moreover, in the cases of unit arc capacities and unit "node capacities" the blocking skew-symmetric flow algorithm has time bounds similar to those established in [8, 21] for Dinits' algorithm. In particular, this implies an algorithm for finding a maximum matching in a nonbipartite graph in $O(\sqrt{n}m)$ time, which matches the time bound for the algorithm of Micali and Vazirani [25]. Finally, extending a clique compression technique of Feder and Motwani [9] to particular skew-symmetric graphs, we speed up the implied maximum matching algorithm to run in $O(\sqrt{n}m \log(n^2/m)/\log n)$ time, improving the best known bound for dense nonbipartite graphs.

Also other theoretical and algorithmic results on skew-symmetric flows and their applications are presented.

Key words. skew-symmetric graph – network flow - matching – b-matching

Mathematics Subject Classification (1991): 90C27, 90B10, 90C10, 05C85

[1]A.V. Goldberg: InterTrust Technologies Corp., 4750 Patrick Henry Drive, Santa Clara, CA 95054 email: avg@acm.org. Part of this research was done while this author was at NEC Research Institute, Inc., Princeton, NJ.

[2]A.V. Karzanov: Corresponding author. Institute for System Analysis of the Russian Academy of Sciences, 9, Prospect 60 Let Oktyabrya, 117312 Moscow, Russia, email: sasha@cs.isa.ac.ru. This author was supported in part by a grant from the Russian Foundation of Basic Research.

1 Introduction

By a *skew-symmetric flow* we mean a flow in a skew-symmetric directed graph which takes equal values on any pair of "skew-symmetric" arcs. This is a synonym of Tutte's *self-conjugate flow in an antisymmetrical digraph* [28]. This paper is devoted to the maximum integer skew-symmetric flow problem, or, briefly, the *maximum IS-flow problem*. We study combinatorial properties of this problem and develop fast algorithms for it.

A well-known fact [10] is that the bipartite matching problem can be viewed as a special case of the maximum flow problem. The combinatorial structure of nonbipartite matchings revealed by Edmonds [5] involves blossoms and is more complicated than the structure of flows. This phenomenon explains, to some extent, why general matching algorithms are typically more intricate relative to flow algorithms. The maximum IS-flow problem is a generalization of both the maximum flow and maximum matching (or b-matching) problems. Moreover, this generalization appears to be well-grounded for two reasons. First, the basic combinatorial and linear programming theorems for usual flows have natural counterparts for IS-flows. Second, when solving problems on IS-flows, one can use intuition, ideas and technical tools well-understood for usual flows, so that the implied algorithms for matchings become more comprehensible.

As the maximum flow problem is related to certain path problems, the maximum IS-flow problem is related to certain problems on so-called *regular paths* in skew-symmetric graphs. We use some theoretical and algorithmic results on the *regular reachability* and *shortest regular path* problems from [16].

Tutte [28] originated a mini-theory of IS-flows (in our terms) to bridge theoretical results on matchings and their generalizations (b-factors, b-matchings, degree-constrained subgraphs, Gallai's track packings, and etc.) and results on usual flows. This theory parallels Ford and Fulkerson's flow theory [10] and includes as basic results the decomposition, augmenting path, and max-flow min-cut theorems. Subsequently, some of those results were re-examined in different, but equivalent, terms by other authors, e.g., in [3, 15, 22].

Recall that the flow decomposition theorem says that a flow can be decomposed into a collection of source-to-sink paths and cycles. The augmenting path theorem says that a flow is maximum if and only if it admits no augmenting path. The max-flow min-cut theorem says that the maximum flow value is equal to the minimum cut capacity. Their skew-symmetric analogs are, respectively, that an IS-flow can be decomposed into a collection of pairs of symmetric source-to-sink paths and pairs of symmetric cycles, that an IS-flow is maximum if and only if it admits no regular augmenting path, and that the maximum IS-flow value is equal to the minimum *odd-barrier* capacity. We give unified and shorter proofs for these skew-symmetric flow theorems.

There is a relationship between skew-symmetric flows and *bidirected flows* introduced by Edmonds and Johnson [6] in their combinatorial study of a natural class of integer linear programs generalizing usual flow and matching problems. In particular, they established a linear programming description for integer bidirected flows. We finish the theoretical part by showing how to obtain a linear programming description for maximum IS-flows directly, using the max-IS-flow min-barrier theorem.

The second, larger, part of this paper is devoted to efficient methods to solve the maximum IS-flow problem (briefly, *MSFP*) in general and special cases, based on the theoretical ground given in the first part. First of all we explain how to adapt the idea of Anstee's elegant methods [1, 2] for b-matchings in which standard flow algorithms are used to construct an optimal half-integer solution and then, after rounding, the "good pre-solution" is transformed into an optimal b-matching by solving $O(n)$ certain path problems. We devise an $O(M(n,m)+nm)$-time algorithm for MSFP in a similar fashion, using a regular reachability algorithm with linear complexity to improve a good pre-solution. Hereinafter n and m denote the numbers of nodes and arcs of the input graph, respectively, and $M(n,m)$ is the time needed to find an integer maximum flow in a usual network with n nodes and m arcs. Without loss of generality, we assume $n = O(m)$.

The next approach is the core of this paper. The purpose is to extend to MSFP the well-known shortest augmenting path algorithm of Edmonds and Karp [7] with complexity $O(nm^2)$, and its improved version, the blocking flow algorithm of Dinits [4] with complexity $O(n^2m)$, so as to preserve the complexity bounds. Recall that the blocking flow algorithm consists of $O(n)$ *phases*, each finding a blocking flow in a layered network representing the union of currently shortest augmenting paths. We introduce concepts of shortest blocking and totally blocking IS-flows and show that an optimal solution to MSFP is also constructed in $O(n)$ phases, each finding a shortest blocking IS-flow in the residual skew-symmetric network. In its turn a phase is reduced to finding a totally blocking IS-flow in an *acyclic* (though not necessarily layered) skew-symmetric network.

The crucial point is to perform the latter task in time comparable with the phase time in Dinits' algorithm (which is $O(nm)$ in general case). We reduce it to a certain auxiliary problem in a usual acyclic digraph. A fast algorithm for this problem provides the desired time bound for a phase.

The complexity of our blocking IS-flow algorithm remains comparable with that of Dinits' algorithm in important special cases where both the number of phases and the phase time significantly decrease. More precisely, Dinits' algorithm applied to the maximum matching problem in a bipartite graph runs in $O(\sqrt{n}m)$ time [18, 20]. Extending that result, it was shown in [8, 21] that for arbitrary nonnegative integer capacities, Dinits' algorithm has $O(\min\{n, \sqrt{\Delta}\})$ phases and each phase runs in $O(\min\{nm, m+\Delta\})$ time, where Δ is the sum of transit capacities of inner nodes. Here the transit capacity of a node (briefly, the *node capacity*) is the maximum flow value that can be pushed through this node. We show that both bounds remain valid for the blocking IS-flow algorithm.

When the network has unit arc capacities (resp. unit inner node capacities), the number of phases turns into $O(\sqrt{m})$ (resp. $O(\sqrt{n})$); in both cases the phase time turns into $O(m)$. The crucial auxiliary problem (that we are able to solve in linear time for unit arc capacities) becomes the following *maximal balanced path-set problem*:

MBP: *Given an acyclic digraph in which one sink and an even set of sources partitioned into pairs are distinguished, find an (inclusion-wise) maximal set of pairwise arc-disjoint paths from sources to the sink such that for each pair $\{z, z'\}$ of sources in the partition, the number of paths from z is equal to that from z'.*

As a consequence, the implied algorithm solves the maximum matching problem in a general graph in the same time, $O(\sqrt{n}m)$, as the algorithm of Micali and Vazirani [25, 29] (cf. [3, 14]) and solves the b-factor or maximum degree-constrained subgraph problem in $O(m^{3/2})$ time, similarly to Gabow [12]. The logical structure of our algorithm differs from that of [25] and sophisticated data structures (incremental trees for set union [13]) are used only in the regular reachability and shortest regular path algorithms of linear complexity from [16] (applied as black-box subroutines) and once in the algorithm for MBP.

Finally, we show that a clique compression technique of Feder and Motwani [9] can be extended to certain skew-symmetric graphs. As a result, our maximum matching algorithm in a general graph is speeded up to run in $O(\sqrt{n}m \log(n^2/m)/\log n)$ time. This matches the best bound for bipartite matching [9].

Fremuth-Paeger and Jungnickel [11] developed an algorithm for MSFP (stated in terms of "balanced flows") which combines Dinits' approach with ideas and tools from [25, 29]; it runs in $O(nm^2)$ time for general capacities and in time slightly slower than $O(\sqrt{n}m)$ in the nonbipartite matching case.

This paper is organized as follows. Basic definitions and facts are given in Section 2. Sections 3 and 4 contain theoretical results on combinatorial and linear programming aspects of IS-flows, respectively. Section 5 describes Anstee's type algorithm for MSFP. The shortest regular augmenting path algorithm and a high level description of the blocking IS-flow method are developed in Section 6. Section 7 gives a short review on facts and algorithms in [16] for regular path problems. Using it, Section 8 explains the idea of implementation of a phase in the blocking IS-flow method. It also bounds the number of phases for special skew-symmetric networks. Section 9 completes the description of the blocking IS-flow algorithm by reducing the problem of finding a totally blocking IS-flow in an acyclic skew-symmetric network to the above-mentioned auxiliary problem in a usual acyclic digraph and devising a fast procedure to solve the later. The concluding Section 10 discusses implications for matchings and their generalizations, and explains how to speed up the implied maximum matching algorithm by use of the clique compression.

This paper is self-contained up to several quotations from [16]. Main results presented in this paper were announced in extended abstract [15]. Subsequently, the authors found a flaw in the original fast implementation of a phase in the blocking IS-flow method. It was corrected in a revised version of this paper (circulated in 2001) where problem MBP and its weighted analog were introduced and efficiently solved, whereas the original version (identical to preprint [17]) embraced only the content of Sections 3–8.

2 Preliminaries

By a *skew-symmetric graph* we mean a digraph $G = (V, E)$ with a mapping (involution) σ of $V \cup E$ onto itself such that: (i) for each $x \in V \cup E$, $\sigma(x) \neq x$ and $\sigma(\sigma(x)) = x$; (ii) for each $v \in V$, $\sigma(v) \in V$; and (iii) for each $a = (v, w) \in E$, $\sigma(a) = (\sigma(w), \sigma(v))$. Although parallel arcs are allowed in G, an arc leaving a node x and entering a node y is denoted by (x, y) when it is not confusing. We assume that σ is fixed (when there are several such mappings) and explicitly

included in the description of G. The node (arc) $\sigma(x)$ is called *symmetric* to a node (arc) x (using, for brevity, the term *symmetric* rather than skew-symmetric). Symmetric objects are also called *mates*, and we usually use notation with primes for mates: x' denotes the mate $\sigma(x)$ of an element x. Note that G can contain an arc a from a node v to its mate v'; then a' is also an arc from v to v'.

Unless mentioned otherwise, when talking about paths (cycles), we mean directed paths (cycles). The symmetry σ is extended in a natural way to paths, subgraphs, and other objects in G; e.g., two paths (cycles) are symmetric if the elements of one of them are symmetric to those of the other and go in the reverse order. Note that G cannot contain self-symmetric paths or cycles. Indeed, if $P = (x_0, a_1, x_1, \dots, a_k, x_k)$ is such a path (cycle), choose arcs a_i and a_j such that $i \le j$, $a_j = \sigma(a_i)$ and $j - i$ is minimum. Then $j > i + 1$ (as $j = i$ would imply $\sigma(a_i) = a_i$ and $j = i + 1$ would imply $\sigma(x_i) = x_{j-1} = x_i$). Now $\sigma(a_{i+1}) = a_{j-1}$ contradicts the minimality of $j - i$.

We call a function h on E *symmetric* if $h(a) = h(a')$ for all $a \in E$.

A *skew-symmetric network* is a quadruple $N = (G, \sigma, u, s)$ consisting of a skew-symmetric graph $G = (V, E)$ with symmetry σ, a nonnegative integer-valued symmetric function u (of *arc capacities*) on E, and a *source* $s \in V$. The mate s' of s is the *sink* of N. A *flow* in N is a function $f : E \to \mathbb{R}_+$ satisfying the capacity constraints

$$f(a) \le u(a) \qquad \text{for all } \ a \in E$$

and the conservation constraints

$$\mathrm{div}_f(x) := \sum_{(x,y)\in E} f(x,y) - \sum_{(y,x)\in E} f(y,x) = 0 \qquad \text{for all } \ x \in V - \{s, s'\}.$$

The value $\mathrm{div}_f(s)$ is called the *value* of f and denoted by $|f|$; we usually assume that $|f| \ge 0$. Now *IS-flow* abbreviates *integer symmetric flow*, the main object that we study in this paper. The *maximum IS-flow problem (MSFP)* is to find an IS-flow of maximum value in N.

The integrality requirement is important: if we do not require f to be integral, then for any integer flow f in N, the flow f', defined by $f'(a) := (f(a) + f(a'))/2$ for $a \in E$, is a flow of the same value as f, which is symmetric but not necessarily integral. Therefore, the *fractional* skew-symmetric flow problem is equivalent to the ordinary flow problem.

Note that, given a digraph $D = (V(D), A(D))$ with two specified nodes p and q and nonnegative integer capacities of the arcs, we can construct a skew-symmetric graph G by taking a disjoint copy D' of D with all arcs reversed, adding two extra nodes s and s', and adding four arcs $(s, p), (s, q'), (q, s'), (p', s')$ of infinite capacity, where p', q' are the copies of p, q in D', respectively. Then there is a natural one-to-one correspondence between integer flows from p to q in D and the IS-flows from s to s' in G. This shows that MSFP generalizes the classical (integer) max-flow problem.

Remark. Sometimes it is useful to consider a sharper version of MSFP in which double-sided capacity constraints $\ell(a) \le f(a) \le u(a)$, $a \in E$, are imposed, where $\ell, u : E \to \mathbb{Z}_+$ and $\ell \le u$ (*problem DMSFP*). Similarly to the max-flow problem with upper and lower capacities [10], DMSFP

is reduced to MSFP in the skew-symmetric network N' obtained from N by subdividing each arc $a = (x, y)$ into three arcs $(x, v), (v, w), (w, y)$ with (upper) capacities $u(a), u(a) - \ell(a), u(a)$, respectively, and adding extra arcs (s, w) and (v, s') with capacity $\ell(a)$ each. It is not difficult to show (e.g., using Theorem 3.5) that DMSFP has a solution if and only if all extra arcs are saturated by a maximum IS-flow f' for N', and in this case f' induces a maximum IS-flow for N in a natural way. For details, see [11].

In our study of IS-flows we rely on results for regular paths in skew-symmetric graphs. A *regular path*, or an *r-path*, is a path in G that does not contain a pair of symmetric arcs. Similarly, an *r-cycle* is a cycle that does not contain a pair of symmetric arcs. The *r-reachability problem (RP)* is to find an r-path from s to s' or a proof that there is none. Given a symmetric function of *arc lengths*, the *shortest r-path problem (SRP)* is to find a minimum length r-path from s to s' or a proof that there is none.

A criterion for the existence of a regular s to s' path is less trivial than that for the usual path reachability; it involves so-called barriers. We say that

$$\mathcal{B} = (A; X_1, \ldots, X_k)$$

is an *s-barrier* if the following conditions hold.

(B1) $A, X_1, \ldots, X_k$ are pairwise disjoint subsets of V, and $s \in A$.

(B2) For $A' = \sigma(A)$, $A \cap A' = \emptyset$.

(B3) For $i = 1, \ldots, k$, X_i is self-symmetric, i.e., $\sigma(X_i) = X_i$.

(B4) For $i = 1, \ldots, k$, there is a unique arc, e^i, from A to X_i.

(B5) For $i, j = 1, \ldots, k$ and $i \neq j$, no arc connects X_i and X_j.

(B6) For $M := V - (A \cup A' \cup X_1 \cup \ldots \cup X_k)$ and $i = 1, \ldots, k$, no arc connects X_i and M.

(B7) No arc goes from A to $A' \cup M$.

(Note that arcs from A' to A, from X_i to A, and from M to A are possible.) Figure 1 illustrates the definition. Tutte proved the following (see also [3, 16]).

Theorem 2.1 [28] *There is an r-path from s to s' if and only if there is no s-barrier.*

This criterion will be used in Section 3 to obtain an analog of the max-flow min-cut theorem for IS-flows. RP is efficiently solvable.

Theorem 2.2 [3, 16] *The r-reachability problem in G can be solved in $O(m)$ time.*

The methods for the maximum IS-flow problem that we develop apply, as a subroutine, the r-reachability algorithm of linear complexity from [16], which finds either a regular s to s' path or an s-barrier. Another ingredient used in our methods is the shortest r-path algorithm for the case of nonnegative symmetric lengths, which runs in $O(m \log n)$ time in general, and in $O(m)$ time for all-unit lengths [16]. The necessary results on RP and SRP are outlined in Section 7.

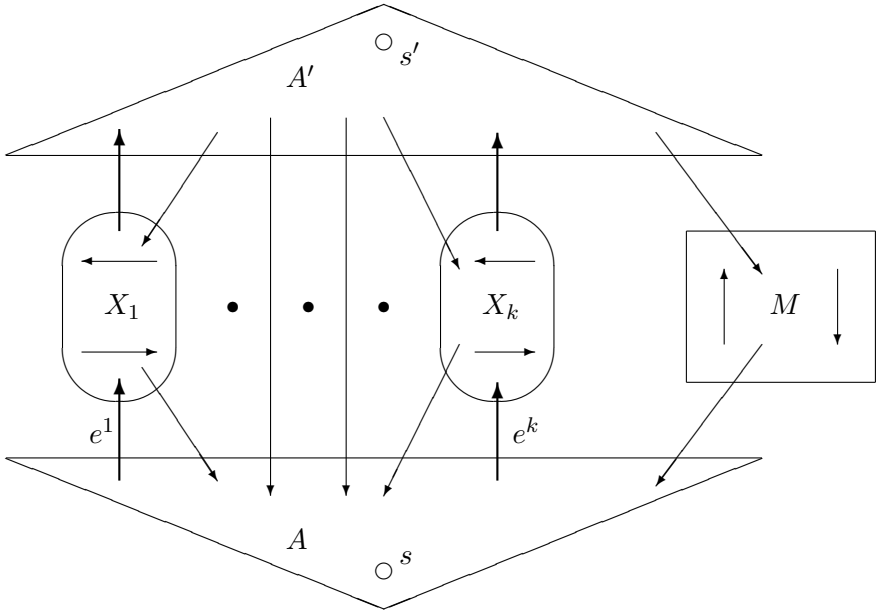


Figure 1: A barrier

In the rest of this paper, σ and s will denote the symmetry map and the source, respectively, regardless of the network in question, which will allow us to use the shorter notation (G, u) for a network (G, σ, u, s). Given a simple path P, the number of arcs on P is denoted by $|P|$ and the incidence vector of its arc set in $\mathbb{R}^E$ is denoted by χ^P, i.e., $\chi^P(a) = 1$ if a is an arc of P, and 0 otherwise.

2.1 Relationships to Matchings

Given an undirected graph $G' = (V', E')$, a *matching* is a subset $M \subseteq E'$ such that no two edges of M have a common endnode. The *maximum matching problem* is to find a matching M whose cardinality $|M|$ is as large as possible.

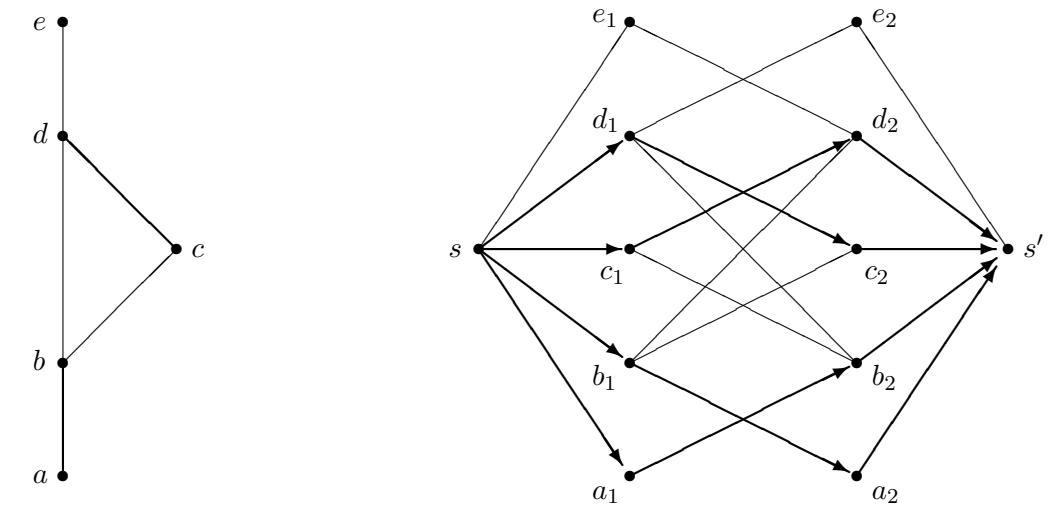
There are well-known generalizations of matchings; for a survey see [23, 24, 26]. Let $u_0, u : E' \to \mathbb{Z}_+ \cup \{\infty\}$ and $b_0, b : V' \to \mathbb{Z}_+$ be functions such that $b_0 \leq b$ and $u_0 \leq u$. A (u_0, u)-*capacitated* (b_0, b)-*matching* is a function $h : E' \to \mathbb{Z}_+$ satisfying the capacity constraints

$$u_0(e) \leq h(e) \leq u(e) \qquad \text{for all } \ e \in E',$$

and the supply constraints

$$b_0(v) \leq \sum_{e=\{v,w\} \in E'} h(e) \leq b(v) \qquad \text{for all } \ v \in V'.$$

The *value* of h is defined to be $h(E')$. Hereinafter, for a numerical function g on a set S and a subset $S' \subseteq S$, $g(S')$ denotes $\sum_{e \in S'} g(e)$. Popular special cases are: a *u-capacitated b-matching*



Matching Skew-symmetric flow

Figure 2: *Reduction example for maximum cardinality matching*

(when $b_0 = 0$); a *degree-constrained subgraph* (when $u \equiv 1$); a *perfect b-matching* (when $u \equiv \infty$ and $b_0 = b$); a *b-factor* (when $u \equiv 1$ and $b_0 = b$). In these cases one assigns $u_0 = 0$. Typically, in unweighted versions, one is asked for maximizing the value of h (in the former two cases) or for finding a feasible h (in the latter two cases).

The general maximum (u_0, u)-capacitated (b_0, b)-matching problem is reduced to the maximum IS-flow problem (MSFP or DMSFP, depending on whether both u_0, b_0 are zero functions or not) without increasing the problem size by more than a constant factor. The construction of the corresponding capacitated skew-symmetric graph $G = (V, E)$ is straightforward (and close to that in [28]):

(i) for each $v \in V'$, V contains two symmetric nodes v_1 and v_2;

(ii) also V contains two additional symmetric nodes s and s' (the source and the sink);

(iii) for each $e = \{v, w\} \in E'$, E contains two symmetric arcs (v_1, w_2) and (w_1, v_2) with lower capacity $u_0(e)$ and upper capacity $u(e)$;

(iv) for each $v \in V'$, E contains two symmetric arcs (s, v_1) and (v_2, s') with lower capacity $b_0(v)$ and upper capacity $b(v)$.

There is a natural one-to-one correspondence between the (u_0, u)-capacitated (b_0, b)-matchings h in G' and the IS-flows f from s to s' in G, and the value of f is twice the value of h. Figure 2 illustrates the correspondence for matchings.

In case of the b-factor or degree-constrained subgraph problem, one may assume that b does not exceed the node degree function of G. Therefore, one can make a further reduction to MSFP in a network with $O(|E'|)$ nodes, $O(|E'|)$ arcs, and unit arc capacities (by getting rid of lower capacities as in the Remark above and then splitting each arc a with capacity $q(a) > 1$ into $q(a)$ parallel arcs of capacity one). In Section 10 we compare the time bounds of our methods for

MSFP applied to the matching problem and its generalizations above with known bounds for these problems.

Edmonds and Johnson [6] studied the class of integer linear programs in which the constraint matrix entries are integers between –2 and +2 and the sum of absolute values of entries in each column (without including entries from the box constraints) does not exceed two. Such a problem is often stated in terms of bidirected graphs (for a survey, see [26, Chapter 36]). Recall that a *bidirected graph* $H = (X, B)$ may contain, besides usual directed edges going from one node to another, edges directed *from* both of its endnodes, and *to* both of them. A particular problem on such an object is the *maximum bidirected flow problem*: given a capacity function $c : B \to \mathbb{Z}_+$ and a *terminal* $p \in X$, find a function (*biflow*) $g : B \to \mathbb{Z}_+$ maximizing the value $\mathrm{div}_g(p)$. (Reasonable versions with more terminals are reduced to this one.) Here $g \le c$ and $\mathrm{div}_g(x) = 0$ for all $x \in X - \{p\}$, where $\mathrm{div}_g(x)$ is the total biflow on the edges directed from x minus the total biflow on the edges directed to x (a loop e at x contributes 0, $2g(e)$ or $-2g(e)$).

The maximum IS-flow problem admits a linear time and space reduction to the maximum biflow problem (in fact, both are equivalent). More precisely, given an instance $N = (G = (V, E), \sigma, u, s)$ of MSFP, take a partition (X, X') of V such that $X' = \sigma(X)$ and $s \in X$. For each pair $\{a, a'\}$ of symmetric arcs in E and nodes $x, y \in X$, assign an edge from x to y if a or a' goes from x to y; an edge from both x, y if a or a' goes from x to $\sigma(y)$; an edge to both x, y if a or a' goes from $\sigma(x)$ to y. This produces a bidirected graph $H = (X, B)$. We set $p := s$ and assign the capacity $c(e)$ of each edge $e \in B$ to be the capacity of the arc from which e is created. There is a one-to-one correspondence between the IS-flows in N and the biflows in (H, c, p), and the values of corresponding flows are equal. A reverse reduction is also obvious. Using these reductions, one can try to derive results for IS-flows from corresponding results on biflows, and vice versa. In this paper we give direct proofs and algorithms for IS-flows.

3 Mini-Theory of Skew-Symmetric Flows

This section extends the classical flow decomposition, augmenting path, and max-flow min-cut theorems of Ford and Fulkerson [10] to the skew-symmetric case. The *support* $\{e \in S : f(e) \neq 0\}$ of a function $f : S \to \mathbb{R}$ is denoted by $\mathrm{supp}(f)$.

Let h be a nonnegative integer symmetric function on the arcs of a skew-symmetric graph $G = (V, E)$. A path (cycle) P in G is called *h-regular* if $h(a) > 0$ for all arcs a of P and each arc $a \in P$ such that $a' \in P$ satisfies $h(a) \ge 2$. Clearly when h is all-unit on E, the sets of regular and h-regular paths (cycles) are the same. We call an arc a of P *ordinary* if $a' \notin P$ and define the *h-capacity* $\delta_h(P)$ of P to be the minimum of all values $h(a)$ for ordinary arcs a on P and all values $\lfloor h(a)/2 \rfloor$ for nonordinary arcs a on P.

To state the symmetric flow decomposition theorem, consider an IS-flow f in a skew-symmetric network $N = (G = (V, E), u)$. An IS-flow g in N is called *elementary* if it is representable as $g = \delta\chi^P + \delta\chi^{P'}$, where P is a simple cycle or a simple path from s to s' or a simple path from s' to s, $P' = \sigma(P)$, and δ is a *positive integer*. Since g is feasible, P is u-regular and $\delta \le \delta_u(P)$. We denote g by (P, P', δ). By a *symmetric decomposition* of f we mean a set D of elementary

flows such that $f = \sum(g : g \in D)$. The following *symmetric decomposition theorem* (see [11, 15]) slightly generalizes a result by Tutte [28] that there exists a symmetric set of $|f|$ paths from s to s' such that any arc a is contained in at most $f(a)$ paths.

Theorem 3.1 *For an IS-flow f in G, there exists a symmetric decomposition consisting of at most m elementary flows.*

Proof. We build up an f-regular path Γ in G until this path contains a simple cycle P or a simple path P connecting s and s'. This will determine a member of the desired flow decomposition. Then we accordingly decrease f and repeat the process for the resulting IS-flow f', and so on until we obtain the zero flow.

We start with Γ formed by a single arc $a \in \mathrm{supp}(f)$. First we grow Γ forward. Let $b = (v, w)$ be the last arc on the current (simple) path Γ. Suppose that $w \neq s, s'$. By the conservation for f at w, $\mathrm{supp}(f)$ must contain an arc $q = (w, z)$. If q' is not on Γ or $f(q) \geq 2$, we add q to Γ.

Suppose q' is on Γ and $f(q) = 1$. Let Γ_1 be the part of Γ between w' and w. Then Γ_1 contains at least one arc since $w \neq w'$. Suppose there is an arc $\widetilde{q} \in \mathrm{supp}(f)$ leaving w and different from q. Then we can add $\widetilde{q}$ to Γ instead of q, forming a longer f-regular path. (Note that since the path Γ is simple, $\widetilde{q}'$ is not on Γ). Now suppose that such a $\widetilde{q}$ does not exist. Then exactly one unit of the flow f leaves w. Hence, exactly one unit of the flow f enters w, implying that $b = (v, w)$ is the only arc entering w in $\mathrm{supp}(f)$, and that $f(b) = 1$. But $\sigma(d)$ also enters w, where d is the first arc on Γ_1. The fact that $\sigma(d) \neq b$ (since Γ_1 is f-regular) leads to a contradiction.

Let (w, z) be the arc added to Γ. If z is not on Γ, then Γ is a simple f-regular path, and we continue growing Γ. If z is on Γ, we discover a simple f-regular cycle P.

If Γ reaches s' or s, we start growing Γ backward from the initial arc a in a way similar to growing it forward. We stop when an f-regular cycle P is found or one of s, s' is reached. In the latter case $P = \Gamma$ is either an f-regular path from s to s' or from s' to s, or an f-regular cycle (containing s or s').

Form the elementary flow $g = (P, P', \delta)$ with $\delta = \delta_f(P)$ and reduce f to $f' := f - \delta\chi^P - \delta\chi^{P'}$. Since P is f-regular, $\delta > 0$. Moreover, there is a pair e, e' of symmetric arcs of P such that either $f'(e) = f'(e') = 0$ or $f'(e) = f'(e') = 1$; we associate such a pair with g. In the former case e, e' vanish in the support of the new IS-flow f', while in the latter case e, e' can be used in further iterations of the decomposition process at most once. Therefore, each pair of arc mates of G is associated with at most two members of the constructed decomposition D, yielding $|D| \leq m$. ■

The above proof gives a polynomial time algorithm for symmetric decomposition. Moreover, the above decomposition process can be easily implemented in $O(nm)$ time, which matches the complexity of standard decomposition algorithms for usual flows.

The decomposition theorem and the fact that the network has no self-symmetric cycles imply the following useful property noticed by Tutte as well.

Corollary 3.2 [28] *For any self-symmetric set $S \subseteq V$ and any IS-flow in G, the total flow on the arcs entering S, as well as the total flow on the arcs leaving S, is even.*

Remark. Another consequence of Theorem 3.1 is that one may assume that G has no arc entering s. Indeed, consider a maximum IS-flow f in G and a symmetric decomposition D of f. Putting together the elementary flows from s to s' in D, we obtain an IS-flow f' in G with $|f'| \geq |f|$, so f' is a maximum flow. Since f' uses no arc entering s or leaving s', deletion of all such arcs from G produces an equivalent problem in a skew-symmetric graph.

Next we state a skew-symmetric version of the augmenting path theorem. It is convenient to consider the graph $G^+ = (V, E^+)$ formed by adding a reverse arc (y, x) to each arc (x, y) of G. For $a \in E^+$, a^R denotes the corresponding reverse arc. The symmetry σ is extended to E^+ in a natural way. Given a (nonnegative integer) symmetric capacity function u on E and an IS-flow f on G, define the *residual capacity* $u_f(a)$ of an arc $a \in E^+$ to be $u(a) - f(a)$ if $a \in E$, and $f(a^R)$ otherwise. An arc $a \in E^+$ is called *residual* if $u_f(a) > 0$, and *saturated* otherwise. Given an IS-flow g in the network (G^+, u_f), we define the function $f \oplus g$ on E by setting $(f \oplus g)(a) := f(a) + g(a) - g(a^R)$. Clearly $f \oplus g$ is a feasible IS-flow in (G, u) whose value is $|f| + |g|$.

By an *r-augmenting path* for f we mean a u_f-regular path from s to s' in G^+. If P is an r-augmenting path and if $\delta \in \mathbb{N}$ does not exceed the u_f-capacity of P, then we can push δ units of flow through a (not necessarily directed) path in G corresponding to P and then δ units through the path corresponding to P'. Formally, f is transformed into $f \oplus g$, where g is the elementary flow (P, P', δ) in (G^+, u_f). Such an augmentation increases the value of f by 2δ.

Theorem 3.3 [28] *An IS-flow f is maximum if and only if there is no r-augmenting path.*

Proof. The direction that the existence of an r-augmenting path implies that f is not maximum is obvious in light of the above discussion.

To see the other direction, suppose that f is not maximum, and let f^* be a maximum IS-flow in G. For $a \in E$ define $g(a) := f^*(a) - f(a)$ and $g(a^R) := 0$ if $f^*(a) \geq f(a)$, while $g(a^R) := f(a) - f^*(a)$ and $g(a) := 0$ if $f^*(a) < f(a)$. One can see that g is a feasible symmetric flow in (G^+, u_f). Take a symmetric decomposition D of g. Since $|g| = |f^*| - |f| > 0$, D has a member (P, P', δ), where P is a u_f-regular path from s to s'. Then P is an r-augmenting path for f. ■

In what follows we will use a simple construction which enables us to reduce the task of finding an r-augmenting path to the r-reachability problem. For a skew-symmetric network (H, h), split each arc $a = (x, y)$ of H into two parallel arcs a_1 and a_2 from x to y (the *first* and *second split-arcs* generated by a). These arcs are endowed with the capacities $[h](a_1) := \lceil h(a)/2 \rceil$ and $[h](a_2) := \lfloor h(a)/2 \rfloor$. Then delete all arcs with zero capacity $[h]$. The resulting capacitated graph is called the *split-graph* for (H, h) and denoted by $S(H, h)$. The symmetry σ is extended to the arcs of $S(H, h)$ in a natural way, by defining $\sigma(a_i) := (\sigma(a))_i$ for $i = 1, 2$.

For a path P in $S(H, h)$, its image in H is denoted by $\omega(P)$ (i.e., $\omega(P)$ is obtained by replacing each arc a_i of P by the original arc $a =: \omega(a_i)$). It is easy to see that if P is regular, then $\omega(P)$ is h-regular. Conversely, for any h-regular path Q in H, there is a (possibly not unique) r-path P in $S(H, h)$ such that $\omega(P) = Q$. Indeed, replace each ordinary arc a of Q by the first split-arc a_1 (existing as $h(a) \geq 1$) and replace each pair a, a' of arc mates in Q by a_i, a'_j for $\{i, j\} = \{1, 2\}$

(taking into account that $h(a) = h(a') \geq 2$). This gives the required r-path P. Thus, Theorem 3.3 admits the following reformulation in terms of split-graphs.

Corollary 3.4 *An IS-flow f in (G, u) is maximum if and only if there is no regular path from s to s' in $S(G^+, u_f)$.*

Finally, the classic max-flow min-cut theorem states that the maximum flow value is equal to the minimum cut capacity. A skew-symmetric version of this theorem involves a more complicated object which is close to an s-barrier occurring in the solvability criterion for the r-reachability problem given in Theorem 2.1. We say that $\mathcal{B} = (A; X_1, \ldots, X_k)$ is an *odd s-barrier* for (G, u) if the following conditions hold.

(O1) $A, X_1, \ldots, X_k$ are pairwise disjoint subsets of V, and $s \in A$.

(O2) For $A' = \sigma(A)$, $A \cap A' = \emptyset$.

(O3) For $i = 1, \ldots, k$, X_i is self-symmetric, *i.e.,* $\sigma(X_i) = X_i$.

(O4) For $i = 1, \ldots, k$, the total capacity $u(A, X_i)$ of the arcs from A to X_i is odd.

(O5) For $i, j = 1, \ldots, k$ and $i \neq j$, no positive capacity arc connects X_i and X_j.

(O6) For $M := V - (A \cup A' \cup X_1 \cup \ldots \cup X_k)$ and $i = 1, \ldots, k$, no positive capacity arc connects X_i and M.

Compare with (B1)–(B7) in Section 2. Define the *capacity* $u(\mathcal{B})$ of $\mathcal{B}$ to be $u(A, V - A) - k$. Since the source is denoted by s throughout, we refer to an odd s-barrier as *odd barrier*.

The following is the *maximum IS-flow minimum barrier theorem.*

Theorem 3.5 [28] *The maximum IS-flow value is equal to the minimum odd barrier capacity.*

Proof. To see that the capacity of an odd barrier $\mathcal{B} = (A; X_1, \ldots, X_k)$ is an upper bound on the value of an IS-flow f, consider a symmetric decomposition D of f. For each member $g = (P, P', \delta)$ of D, where P is a path from s to s', take the *last* arc $a = (x, y)$ of the *first* path P such that $x \in A$. If $y \in A'$, then the symmetric arc a' (which is in P') also goes from A to A' (by (O2)), and therefore, g uses at least 2δ units of the capacity of arcs from A to A'. Associate g with the pair a, a'. Now let $y \notin A'$. Since $y \notin A$, y is either in $Y := M$ or in $Y := X_i$ for some i. The choice of a and (O1),(O5),(O6) imply that P leaves Y by an arc b from Y to A'. Then the symmetric arc b' (which is in P') goes from A to Y (since Y is self-symmetric), and therefore, g uses at least 2δ units of the capacity $u(A, Y)$. Associate g with the pair a, b' (possibly $a = b'$). Note that at least one unit of each capacity $u(A, X_i)$ is not used under the canonical way we associate the elementary s to s' flows of D with arcs from A to $V - A$ (since $u(A, X_i)$ is odd, by (O4)). By these reasonings, $|f| \leq u(\mathcal{B})$.

Next we show that the two values in the theorem are equal. Let f be a maximum IS-flow. By Corollary 3.4, the split-graph $S = S(G^+, u_f)$ contains no s to s' r-path, so it must contain an s-barrier $\mathcal{B} = (A; X_1, \ldots, X_k)$, by Theorem 2.1.

Let e^i be the (unique) arc from A to X_i in S (see (B4) in Section 2). By the construction of S, it follows that the residual capacity u_f of every arc from A to X_i in G^+ is zero except for the arc $\omega(e^i)$, whose residual capacity is one. Hence,

(i) if e^i was formed by splitting an arc $a \in E$, then a goes from A to X_i, and $f(a) = u(a) - 1$;

(ii) if e^i was formed by splitting a^R for $a \in E$, then a goes from X_i to A, and $f(a) = 1$;

(iii) all arcs from A to X_i in G, except a in case (i), are saturated by f;

(iv) all arcs from X_i to A in G, except a in case (ii), are free of flow.

Furthermore, comparing arcs in S and G, we observe that:

(v) property (B7) implies that the arcs from A to $A' \cup M$ are saturated and the arcs from $A' \cup M$ to A are free of flow;

(vi) property (B5) implies (O5) and (B6) implies (O6).

Properties (i)–(iv),(O5),(O6) together with Corollary 3.2 provide (O4). So $\mathcal{B}$ is an odd s-barrier in G. We have $|f| = f(A, V - A) - f(V - A, A) = u(A, V - A) - k$ (in view of (i)–(v)). Hence, $|f| = u(\mathcal{B})$. ■

4 Integer and Linear Programming Formulations

Although methods of solving MSFP developed in subsequent sections will not use explicitly linear programming aspects exhibited in this section, such aspects help to understand more about the structure of IS-flows.

MSFP is stated as an integer program in a straightforward way. We use function rather than vector notation. For functions g, h on a set S, $g \cdot h$ denotes the inner product $\sum_{x \in S} g(x)h(x)$. Assuming that no arc of G enters the source s (see the Remark in the previous section), MSFP can be written as follows:

$$\textbf{maximize } |f| = \sum\nolimits_{(s,v) \in E} f(s, v) \qquad \textbf{subject to} \tag{1}$$

$$f(a) \geq 0 \qquad \forall a \in E \tag{2}$$

$$f(a) \leq u(a) \qquad \forall a \in E \tag{3}$$

$$-\sum\nolimits_{(u,v) \in E} f(u, v) + \sum\nolimits_{(v,w) \in E} f(v, w) = 0 \qquad \forall v \in V - \{s, s'\} \tag{4}$$

$$f(a) - f(\sigma(a)) = 0 \qquad \forall a \in E \tag{5}$$

$$f(a) \quad \text{integer} \qquad \forall a \in E \tag{6}$$

A linear programming formulation for MSFP is obtained by replacing the integrality condition (6) by linear constraints related to certain objects that we call odd fragments in G. The correctness of the resulting linear program will be shown by use of the max-min relation between IS-flow and odd barriers in Theorem 3.5. Alternatively, one can try to derive it from a linear programming characterization of integer bidirected flows in [6] (using the reduction as in Section 2).

An *odd fragment* is a pair $\rho = (V_\rho, U_\rho)$, where V_ρ is a *self-symmetric* set of nodes with $s \notin V_\rho$, and U_ρ is a subset of arcs entering V_ρ such that the total capacity $u(U_\rho)$ is odd. The *characteristic function* χ_ρ of ρ is the function on E defined by

$$\chi_\rho(a) := \begin{cases} 1 & \text{if } a \in U_\rho \cup \sigma(U_\rho), \\ -1 & \text{if } a \in \delta(V_\rho) - (U_\rho \cup \sigma(U_\rho)), \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Here $\delta(V_\rho)$ is the set of arcs with one end in V_ρ and the other in $V - V_\rho$. We denote the set of odd fragments by Ω.

Let f be a (feasible) IS-flow and $\rho \in \Omega$. By (7) and the symmetry of u, we have $f \cdot \chi_\rho \leq u(U_\rho) + u(\sigma(U_\rho)) = 2u(U_\rho)$. Moreover, $f \cdot \chi_\rho$ is at most $2u(U_\rho) - 2$; this immediately follows from Corollary 3.2 and the fact that $u(U_\rho)$ is odd. This gives new linear constraints for MSFP:

$$f \cdot \chi_\rho \leq 2u(U_\rho) - 2 \quad \text{for each } \rho \in \Omega. \tag{8}$$

Addition of these constraints enables us to drop off the symmetry constraints (5) and the integrality constraints (6) without changing the optimum value of the linear program. This fact is implied by the following theorem.

Theorem 4.1 *Every maximum IS-flow is an optimal solution to the linear program* (1)–(4), (8).

Proof. Assign a dual variable $\pi(v) \in \mathbb{R}$ (a *potential*) to each node $v \in V$, $\gamma(a) \in \mathbb{R}_+$ (a *length*) to each arc $a \in E$, and $\xi(\rho) \in \mathbb{R}_+$ to each odd fragment $\rho \in \Omega$. Consider the linear program:

$$\textbf{minimize } \psi(\pi, \gamma, \xi) := \sum_E u(a)\gamma(a) + \sum_\Omega (2u(U_\rho) - 2)\xi(\rho) \qquad \textbf{subject to} \tag{9}$$

$$\gamma(a) \geq 0 \qquad \forall a \in E \tag{10}$$

$$\xi(\rho) \geq 0 \qquad \forall \rho \in \Omega \tag{11}$$

$$\pi(s) = 0 \tag{12}$$

$$\pi(s') = 1 \tag{13}$$

$$\pi(v) - \pi(w) + \gamma(a) + \sum_\Omega \xi(\rho)\chi_\rho(a) \geq 0 \qquad \forall a = (v, w) \in E. \tag{14}$$

In fact, (9)–(14) is dual to linear program (1)–(4),(8). (To see this, introduce an extra arc (s', s), add the conservation constraints for s and s', and replace the objective (1) by $\max\{f(s', s)\}$. The latter generates the dual constraint $\pi(s') - \pi(s) \geq 1$. We can replace it by the equality or impose (12)–(13).) Therefore,

$$\max |f| = \min \psi(\pi, \gamma, \xi), \tag{15}$$

where the maximum and minimum range over the corresponding feasible solutions.

We assert that every maximum IS-flow f achieves the maximum in (15). To see this, choose an odd barrier $\mathcal{B} = (A; X_1, \ldots, X_k)$ of minimum capacity $u(\mathcal{B})$. For $i = 1, \ldots, k$, let U_i be the set of arcs from A to X_i; then $\rho_i = (X_i, U_i)$ is an odd fragment for G, u. Define $\pi(v)$ to be 0 for $v \in A$, 1 for $v \in A'$, and 1/2 otherwise. Define $\gamma(a)$ to be 1 for $a \in (A, A')$, 1/2 for $a \in (A, M) \cup (M, A')$, and 0 otherwise, where $M = V - (A \cup A' \cup X_1 \cup \ldots \cup X_k)$. Define $\xi(\rho_i) = 1/2$ for $i = 1, \ldots, k$, and $\xi(\rho) = 0$ for the other odd fragments in (G, u).

One can check that (14) holds for all arcs a (e.g., both values $\pi(w) - \pi(v)$ and $\gamma(a) + \sum_\Omega \xi(\rho)\chi_\rho(a)$ are equal to 1 for $a = (v, w) \in (A, A')$, and 1/2 for $a = (v, w) \in (A, L) \cup (L, A')$, where $L := V - (A \cup A')$). Thus π, γ, ξ are feasible.

Using the fact that $u(A, M) = u(M, A')$, we observe that $u \cdot \gamma = u(A, A') + u(A, M)$. Also

$$\sum_\Omega (2u(U_\rho) - 2)\xi(\rho) = \sum_{i=1}^{k} \frac{1}{2}(2u(U_i) - 2) = \left(\sum_{i=1}^{k} u(A, X_k)\right) - k.$$

This implies $\psi(\pi, \gamma, \xi) = u(\mathcal{B})$, and now the result follows from Theorem 3.5. ■

5 Algorithm Using a Good Pre-Solution

Anstee [1, 2] developed efficient methods for b-factor and b-matching problems (unweighted or weighted) based on the idea that a good pre-solution can easily be found by solving a corresponding flow problem. In this section we adapt his approach to solve the maximum IS-flow problem in a skew-symmetric network $N = (G = (V, E), u)$. The algorithm that we devise is relatively simple; it finds a "nearly optimal" IS-flow and then makes $O(n)$ augmentations to obtain a maximum IS-flow. The algorithm consists of four stages.

The *first* stage ignores the fact that N is skew-symmetric and finds an integer maximum flow g in N by use of a max-flow algorithm. Then we set $h(a) := (g(a) + g(a'))/2$ for all arcs $a \in E$. Since $\mathrm{div}_h(s) = \mathrm{div}_g(s)/2 - \mathrm{div}_g(s')/2 = \mathrm{div}_g(s)$, h is a maximum flow as well. Also h is symmetric and *half-integer*. Let Z be the set of arcs on which h is not integer. If $Z = \emptyset$, then h is already a maximum IS-flow; so assume this is not the case.

The *second* stage applies simple transformations of h to reduce Z. Let $H = (X, Z)$ be the subgraph of G induced by Z. Obviously, for each $x \in V$, $\mathrm{div}_h(x)$ is an integer, so x is incident to an even number of arcs in Z. Therefore, we can decompose H into simple, not necessarily directed, cycles $C_1, \ldots, C_r$ which are pairwise arc-disjoint. Moreover, we can find, in linear time, a decomposition in which each cycle C_i is either self-symmetric ($C_i = \sigma(C_i)$) or symmetric to another cycle C_j ($C_i = \sigma(C_j)$).

To do this, we start with some node $v_0 \in X$ and grow in H a simple (undirected) path $P = (v_0, e_1, v_1, \ldots, e_q, v_q)$ such that the mate v_i' of each node v_i is not in P. At each step, we choose in H an arc $e \neq e_q$ incident to the last node v_q (e exists since H is eulerian); let x be the other end node of e. If none of x, x' is in P, then we add e to P. If some of x, x' is a node of P, v_i say, then we shorten P by removing its end part from e_{i+1} and delete from H the arcs $e_{i+1}, \ldots, e_q, e$ and their mates. One can see that the arcs deleted induce a self-symmetric cycle (when $x' = v_i$) or two disjoint symmetric cycles (when $x = v_i$). We also remove the isolated nodes created by the arc deletions and change the initial node v_0 if needed. Repeating the process for the new current graph H and path P, we eventually obtain the desired decomposition $\mathcal{C}$, in $O(|Z|)$ time.

Next we examine the cycles in $\mathcal{C}$. Each pair C, C' of symmetric cycles is canceled by sending a half unit of flow through C and through C', i.e., we increase (resp. decrease) $h(e)$ by 1/2 on each forward (resp. backward) arc e of these cycles. The resulting function h is symmetric, and $\mathrm{div}_h(x)$

is preserved at each node x, whence h is again a maximum symmetric flow. Now suppose that two self-symmetric cycles C and D meet at a node x. Then they meet at x' as well. Concatenating the x to x' path in C and the x' to x path in D and concatenating the rests of C and D, we obtain a pair of symmetric cycles and cancel these cycles as above. These cancellations result in $\mathcal{C}$ consisting of pairwise *node-disjoint* self-symmetric cycles, say $C_1, \ldots, C_k$. The second stage takes $O(m)$ time.

The *third* stage transforms h into an IS-flow f whose value $|f|$ is at most k units below $|h|$. For each i, fix a node t_i in C_i and change h on C_i by sending a half unit of flow through the t_i to t'_i path in C_i and through the reverse to the t'_i to t_i path in it. The resulting function h is integer and symmetric and the divergences preserve at all nodes except for the nodes t_i and t'_i where we have $\mathrm{div}_h(t_i) = -\mathrm{div}_h(t'_i) = 1$ for each i (assuming, without loss of generality, that all t_i's are different from s'). Therefore, h is, in essence, a multiterminal IS-flow with sources $s, t_1, \ldots, t_k$ and sinks $s', t'_1, \ldots, t'_k$. A genuine IS-flow f from s to s' is extracted by reducing h on some h-regular paths. More precisely, we add to G artificial arcs $e_i = (s, t_i)$, $i = 1, \ldots, k$ and their mates, extend h by ones to these arcs and construct a symmetric decomposition $\mathcal{D}$ (defined in Section 3) for the obtained function h' in the resulting graph G' (clearly h' is an IS-flow of value $|h| + k$).

Let $\mathcal{D}'$ be the set of elementary flows in $\mathcal{D}$ formed by the paths or cycles which contain artificial arcs. Then $\delta = 1$ for each $(P, P', \delta) \in \mathcal{D}'$. Define $f' := h' - \sum(\chi^P + \chi^{P'} : (P, P', 1) \in \mathcal{D}')$. Then f' is an IS-flow in G', and $|f'| \geq |h'| - 2k \geq |h| - k$. Moreover, since $f(e_i) = 0$ for $i = 1, \ldots, k$, the restriction f of f' to E is an IS-flow in G, and $|f| = |f'|$. Thus, $|f| \geq |h| - k$, and now the facts that $k \leq n/2$ (as the nodes $t_1, \ldots, t_k, t'_1, \ldots, t'_k$ are different) and that h is a maximum flow in N imply that the value of f differs from the maximum IS-flow value by $O(n)$. The third stage takes $O(nm)$ time (the time needed to construct a symmetric decomposition of h').

The final, *fourth*, stage transforms f into a maximum IS-flow. Each iteration applies the r-reachability algorithm (RA) mentioned in Section 2 to the split-graph $S(G^+, u_f)$ in order to find a u_f-regular s to s' path P in G^+ and then augment the current IS-flow f by the elementary flow $(P, P', \delta_{u_f}(P))$ as explained in Section 3. Thus, a maximum IS-flow in N is constructed in $O(n)$ iterations. Since the RA runs in $O(m)$ time (by Theorem 2.2), the fourth stage takes $O(nm)$ time.

Summing up the above arguments, we conclude with the following.

Theorem 5.1 *The above algorithm finds a maximum IS-flow in N in $O(M(n, m) + nm)$ time, where $M(n, m)$ is the running time of the max-flow procedure it applies.*

6 Shortest R-Augmenting Paths and Blocking IS-Flows

Theorem 3.3 and Corollary 3.4 prompt an alternative method for finding a maximum IS-flow in a skew-symmetric network $N = (G, u)$, which is analogous to the method of Ford and Fulkerson for usual flows. It starts with the zero flow, and at each iteration, the current IS-flow f is augmented by an elementary flow in (G^+, u_f) (found by applying the r-reachability algorithm to $S(G^+, u_f)$). Since each iteration increases the value of f by at least two, a maximum IS-flow is constructed in pseudo-polynomial time. In general, this method is not competitive to the method of Section 5.

More efficient methods involve the concepts of shortest r-augmenting paths and shortest blocking IS-flows that we now introduce. Let g be an IS-flow in a skew-symmetric network $(H = (V, W), h)$. We call $g(W)$ $(= \sum_{e \in W} g(e))$ the *volume* of g. Considering a symmetric decomposition $D = \{(P_i, P_i', \delta_i) : i = 1, \ldots, k)$ of g, we have

$$g(W) = \sum(\delta_i |P_i| + \delta_i |P_i'| : i = 1, \ldots, k) \geq |g| \min\{|P_i| : i = 1, \ldots, k\}.$$

This implies

$$g(W) \geq |g| \text{r-dist}_{S(H,h)}(s, s'), \tag{16}$$

where $\text{r-dist}_{H'}(x, y)$ denotes the minimum length of a regular x to y path in a skew-symmetric graph H' (the *regular distance* from x to y). We say that an IS-flow g is

(i) *shortest* if (16) holds with equality, i.e., some (equivalently, any) symmetric decomposition of g consists of shortest h-regular paths from s to s';

(ii) *totally blocking* if there is no $(h - g)$-regular path from s to s' in H, i.e., we cannot augment g using only residual capacities in H itself;

(iii) *shortest blocking* if g is shortest (as in (i)) and

$$\text{r-dist}_{S(H,h-g)}(s, s') > \text{r-dist}_{S(H,h)}(s, s'). \tag{17}$$

Note that a shortest blocking IS-flow is not necessarily totally blocking, and vice versa.

Given a skew-symmetric network $N = (G, u)$, the *shortest r-augmenting path method (SAPM)*, analogous to the method of Edmonds and Karp [7] for usual flows, starts with the zero flow, and each iteration augments the current IS-flow f by a shortest elementary flow $g = (P, P', \delta_{u_f}(P))$.

The *shortest blocking IS-flow method (SBFM)*, analogous to Dinits' method [4], starts with the zero flow, and each *phase* (big iteration) augments the current IS-flow f by performing the following two steps:

(P1) Find a shortest blocking IS-flow g in (G^+, u_f).

(P2) Update $f := f \oplus g$.

Both methods terminate when f no longer admits r-augmenting paths (i.e., g becomes the zero flow). The following observation is crucial for our methods.

Lemma 6.1 *Let g be a shortest IS-flow in (G^+, u_f), and let $f' := f \oplus g$. Let k and k' be the minimum lengths of r-augmenting paths for f and f', respectively. Then $k' \geq k$. Moreover, if g is a shortest blocking IS-flow, then $k' > k$.*

Proof. Take a shortest $u_{f'}$-regular path P from s to s' in G^+. Then $|P| = k'$ and $g' = (P, P', 1)$ is an elementary flow in $(G^+, u_{f'})$.

Note that $\text{supp}(g)$ does not contain opposed arcs $a = (x, y)$ and $b = (y, x)$. Otherwise decreasing g by one on each of a, b, a', b' (which are, obviously, distinct), we would obtain the

IS-flow $\widetilde{g}$ in (G^+, u_f) such that $|\widetilde{g}| = |g|$ and $\widetilde{g}(E^+) < g(E^+)$, which is impossible because $\widetilde{g}(E^+) \geq k|\widetilde{g}|$ and $g(E^+) = k|g|$. This implies that each arc a in the set $Z := \{a \in E^+ : g(a^R) = 0\}$ satisfies

$$u_{f'}(a) = u_f(a) - g(a). \tag{18}$$

If $\mathrm{supp}(g') \subseteq Z$, then g' is a feasible IS-flow in (G^+, u_f) (by (18)), whence $k' = g'(E^+)/|g'| \geq k$. Moreover, if, in addition, g is a shortest blocking IS-flow, then (17) and the fact that $g' \leq u_f - g$ (by (18)) imply $k' > k$.

Now suppose there is an arc $e \in E^+$ such that $g'(e) > 0$ and $g(e^R) > 0$. For each $a \in E^+$, put $\lambda(a) := \max\{0, g(a) + g'(a) - g(a^R) - g'(a^R)\}$. One can check that $\lambda(a) \leq u_f(a)$ for all arcs a and that $\mathrm{div}_\lambda(v) = 0$ for all nodes $v \neq s, s'$. Therefore, λ is an IS-flow in (G^+, u_f) with $|\lambda| = |g| + |g'| = |g| + 2$. Also $\lambda(E^+) < g(E^+) + g'(E^+)$ since for the e above, $\lambda(e) + \lambda(e^R) < g'(e) + g(e^R)$. We have

$$2k' = g'(E^+) > \lambda(E^+) - g(E^+) \geq k(|g| + 2) - k|g| = 2k,$$

yielding $k' > k$. ■

Thus, each iteration of SAPM does not decrease the minimum length of an r-augmenting path, and each phase of SBFM increases this length. This gives upper bounds on the numbers of iterations.

Corollary 6.2 *SAPM terminates in at most* $(n-1)m$ *iterations.*

(This follows by observing, in the proof of Lemma 6.1, that on the iterations with the same length of shortest r-augmenting paths, the subgraph of G^+ induced by the arcs contained in such paths is monotone nonincreasing, and each iteration reduces the capacity of some arc of this subgraph, as well as the capacity of its mate, to zero or one.)

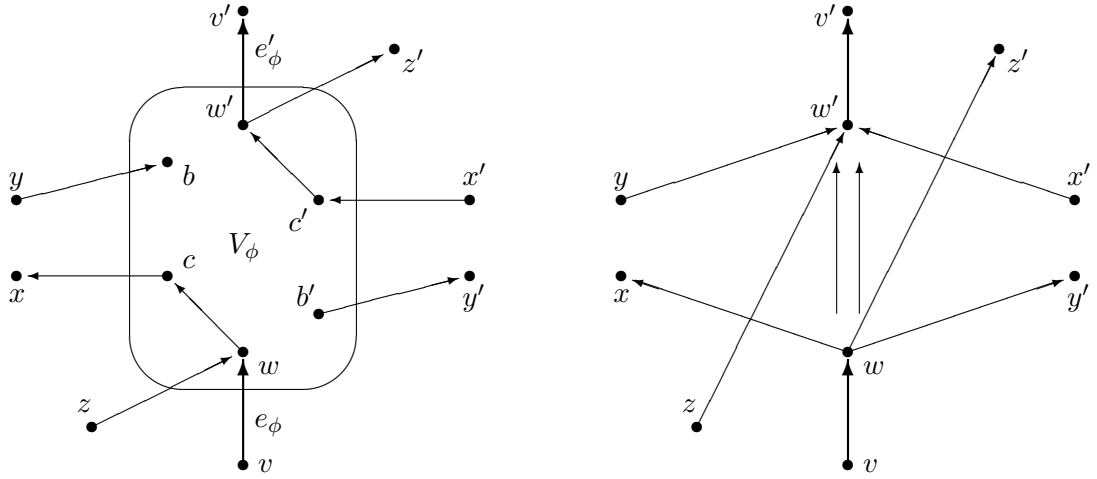
Corollary 6.3 *SBFM terminates in at most* $n - 1$ *phases.*

As mentioned above, SBFM can be considered as a skew-symmetric analog of Dinits' blocking flow algorithm. Recall that each phase of that algorithm constructs a blocking flow in the subnetwork H formed by the nodes and arcs of shortest augmenting paths. Such a network is acyclic (moreover, layered), and a blocking flow in H is easily constructed in $O(nm)$ time.

The problem of finding a shortest blocking IS-flow ((P1) above) is more complicated. Let H be the subgraph of G^+ formed by the nodes and arcs contained in shortest u_f-regular s to s' paths. Such an H need not be acyclic (a counterexample is not difficult). In Section 8 we will show that problem (P1) can be reduced to a seemingly easier task, namely, to finding a totally blocking IS-flow in a certain acyclic network $(\overline{H}, \overline{h})$. Such a network arises when the shortest r-path algorithm from [16] is applied to the split-graph $S(G^+, u_f)$ with unit arc lengths. First, however, we need to tell more about the r-reachability and shortest r-path algorithms from [16].

7 Properties of Regular and Shortest Regular Path Algorithms

In this section we exhibit certain properties of the algorithms in [16], referring the reader to that paper for details. We also establish an additional fact (Lemma 7.4), which will be used later.



Before trimming

After trimming

Figure 3: Fragment trimming example

7.1 The Regular Reachability Algorithm (RA)

Let $\Gamma = (V, E)$ be a skew-symmetric graph with source s and sink $s' = \sigma(s)$ (as before, σ is the symmetry map). A *fragment* (or an *s-fragment*) in Γ is a pair $\phi = (V_\phi, e_\phi = (v, w))$, where V_ϕ is a self-symmetric set of nodes of Γ with $s \notin V_\phi$ and e_ϕ is an arc entering V_ϕ, i.e., $v \notin V_\phi \ni w$ (cf. the definition of odd fragments in Section 4). We refer to e_ϕ and e'_ϕ as the *base* and *anti-base* arcs of ϕ, respectively. Let us say that the fragment is *well-reachable* if

(i) for each node $x \in V_\phi$, there is an r-path from w to x in the subgraph induced by V_ϕ (and therefore, an r-path from x to $w' = \sigma(w)$), and

(ii) there is an r-path from s to v disjoint from V_τ.

The *trimming operation* applied to ϕ (which is analogous to shrinking a blossom in matching algorithms) transforms Γ by removing the nodes of $V_\phi - \{w, w'\}$ and modifying the arcs as follows.

(T1) Each arc $a = (x, y) \in E$ such that either $x, y \in V - V_\phi$ or $a = e_\phi$ or $a = e'_\phi$ remains an arc from x to y.

(T2) Each arc $(x, y) \in E - \{e'_\phi\}$ that leaves V_ϕ is replaced by an arc from w to y, and each arc $(x, y) \in E - \{e_\phi\}$ that enters V_ϕ is replaced by an arc from x to w'.

(T3) Each arc with both ends in V_ϕ is replaced by an arc from w to w'.

(A variant of trimming deletes all arcs in (T3).) The image of an arc a in the new graph is denoted again by a (so its end nodes can be changed, but not its name). Figure 3 illustrates fragment trimming. The new Γ is again skew-symmetric.

The algorithm RA relies on the following property.

Statement 7.1 [16] *If ϕ is a well-reachable fragment, then trimming ϕ preserves the existence (or non-existence) of a regular path from s to s'.*

RA searches for a regular s to s' path in Γ, starting with the trivial path $P = s$. Each iteration either increases the current r-path P, or reveals a well-reachable fragment and trims it, producing the new current graph Γ and accordingly updating P. It terminates when either an s to s' r-path $\overline{P}$ or an s-barrier $\overline{\mathcal{B}}$ in the final graph $\overline{\Gamma}$ is found (cf. Theorem 2.1). The postprocessing stage extends $\overline{P}$ into a regular s to s' path P of the initial Γ (cf. Statement 7.1) in the former case (the *path restoration procedure*) and extends $\overline{\mathcal{B}}$ into a barrier $\mathcal{B}$ of the initial Γ in the latter case (the *barrier restoration procedure*).

The fragments of current graphs revealed by RA determine fragments of the initial Γ in a natural way; all fragments are well-reachable. Moreover, the set Φ of these fragments of the initial Γ is *well-nested*. This means that

(F1) for distinct $\phi, \psi \in \Phi$, either $V_\phi \subset V_\psi$ or $V_\psi \subset V_\phi$ or $V_\phi \cap V_\psi = \emptyset$, and

(F2) for $\phi, \psi \in \Phi$, if $V_\psi \subset V_\phi$ and $e_\psi \in \delta(V_\phi)$ then $e_\psi = e_\phi$, and if $V_\phi \cap V_\psi = \emptyset$ and $e_\psi \in \delta(V_\phi)$ then $e_\phi \notin \delta(V_\psi)$.

(Recall that for $X \subseteq V$, $\delta(X)$ is the set of arcs with one end in X and the other in $V - X$.) Let us say that a path R in Γ is *compatible with* Φ if for each $\phi \in \Phi$, $p := |R \cap \delta(V_\phi)| \leq 2$, and if $p = 2$ then R contains exactly one of e_ϕ, e'_ϕ. The following additional properties (relying on (T1)–(T3)) are important:

(19) any regular s to s' path $\overline{P}$ in the final graph $\overline{\Gamma}$ is extendable to a regular s to s' path P compatible with Φ in the initial Γ, and the path restoration procedure applied to $\overline{P}$ constructs such a P in $O(|P| + d)$ time, where d is the total size of maximal fragments in Φ traversed by P;

(20) for each $\phi \in \Phi$ and each arc $a \neq e'_\phi$ leaving V_ϕ, there exists a compatible with Φ r-path $Q_\phi(a)$ with the first arc e_ϕ, the last arc a and all intermediate nodes in V_ϕ; such a path can be constructed by (a phase of) the path restoration procedure in $O(|V_\phi|)$ time.

A fast implementation of RA (supported by the disjoint set union data structure of [13]) runs in linear time, as indicated in Theorem 2.2.

7.2 The Shortest Regular Path Algorithm (SRA)

We now consider the shortest regular path problem (SRP) in a skew-symmetric graph $\Gamma = (V, E)$ with *nonnegative symmetric* lengths $\ell(e)$ of the arcs $e \in E$: find a minimum length regular path from s to s'. One may assume that s' is r-reachable from s. The dual problem involves above-mentioned fragments. Define the characteristic function χ_ϕ of a fragment $\phi = (V_\phi, e_\phi)$ by

$$\chi_\phi(a) := \begin{cases} 1 & \text{for } a = e_\phi, e'_\phi, \\ -1 & \text{for } a \in \delta(V_\phi) - \{e_\phi, e'_\phi\}, \\ 0 & \text{for the remaining arcs of } \Gamma. \end{cases} \tag{21}$$

(Compare with (7).) For a function $\pi : V \to \mathbb{R}$ (of node *potentials*) and a nonnegative function ξ on a set Φ of fragments, define the *reduced length* of an arc $e = (x, y)$ to be

$$\ell_\pi^\xi(e) := \ell(e) + \pi(x) - \pi(y) + \sum_{\phi \in \Phi} \xi(\phi)\chi_\phi(e).$$

An optimality criterion for SRP can be formulated as follows.

Theorem 7.2 [16] *A regular path P from s to s' is a shortest r-path if and only if there exist a potential $\pi : V \to \mathbb{R}$, a set Φ of fragments, and a* positive *function ξ on Φ such that*

$$\ell_\pi^\xi(e) \geq 0 \quad \textit{for each} \quad e \in E; \tag{22}$$

$$\ell_\pi^\xi(e) = 0 \quad \textit{for each} \quad e \in P; \tag{23}$$

$$\chi^P \cdot \chi_\phi = 0 \quad \textit{for each} \quad \phi \in \Phi. \tag{24}$$

The *shortest r-path algorithm (SRA)* from [16] implicitly maintains π, Φ, ξ in the input graph Γ and iteratively modifies the graph by trimming certain fragments. Let Γ^0 be the subgraph of the current Γ with the same set of nodes and with the arcs having zero reduced length, called the current *0-subgraph* (recall that the arcs of the current graph are identified with the corresponding arcs of the initial one). Each iteration applies the above r-reachability algorithm RA to search for a regular s to s' path in Γ^0. If such a path is found, the algorithm terminates and outputs this path to a postprocessing stage. If such a path does not exist, then, using the barrier $\mathcal{B} = (A; X_1, \ldots, X_k)$ in Γ^0 constructed by RA, the iteration trims the fragments determined by the sets X_i and updates π, Φ, ξ, modifying Γ^0. The reduced lengths of the arcs within the newly and previously extracted fragments, as well as of their base and anti-base arcs, are not changed.

Let $\overline{\Gamma}$ and $\overline{\Gamma}^0$ denote the final graph Γ and the 0-subgraph in it, respectively, and $\overline{P}$ the regular s to s' path in $\overline{\Gamma}^0$ found by the algorithm. Let Γ^0 stand for the 0-subgraph of the initial graph Γ (concerning the reduced arc lengths determined by the resulting π, Φ, ξ). We call Γ^0 and $\overline{\Gamma}^0$ the *full* and *trimmed 0-graphs*, respectively. The postprocessing stage applies the path restoration procedure of RA to extend $\overline{P}$ into a regular s to s' path P in Γ^0, in time indicated in (19). It also explicitly constructs Γ^0 (in linear time).

Note that any s to s' r-path or r-cycle Q in Γ compatible with Φ satisfies $\chi^Q \cdot \chi_\phi = 0$ for each $\phi \in \Phi$. By (19), *any* s to s' r-path $\overline{P}$ in $\overline{\Gamma}^0$ is extendable to an s to s' r-path P in Γ^0 compatible with Φ. Therefore, P is shortest, by Theorem 7.2.

Theorem 7.3 [16] *For nonnegative symmetric arc lengths ℓ, SRA runs in $O(m \log n)$ time, and in $O(m\sqrt{\log L})$ time when ℓ is integer-valued and L is the maximum arc length. Furthermore, the algorithm constructs (implicitly) π, Φ, ξ as in Theorem 7.2, where π is* anti-symmetric *(i.e., $\pi(x) = -\pi(x')$ for all $x \in V$), and constructs (explicitly) the trimmed 0-graph $\overline{\Gamma}^0$ and the full 0-graph Γ^0 such that:*

(A1) *Φ is well-nested (obeys* (F1)–(F2)*) and consists of well-reachable fragments in Γ^0; in particular, $\ell_\pi^\xi(e_\phi) = 0$ for each $\phi \in \Phi$;*

(A2) Φ *satisfies (19) and (20) with* $\Gamma^0, \overline{\Gamma}^0$ *instead of* $\Gamma, \overline{\Gamma}$*; in particular, any regular* s *to* s' *path of* $\overline{\Gamma}^0$ *is (efficiently) extendable to a shortest regular* s *to* s' *path in* (Γ, ℓ)*.*

(Note that the anti-symmetry of π and the symmetry of ℓ and χ_ϕ for all $\phi \in \Phi$ imply that the reduced length function ℓ^ξ_π is symmetric. Therefore, the graphs Γ^0 and $\overline{\Gamma}^0$ are indeed skew-symmetric.) Let $\Phi^{\max}$ denote the set of maximal fragments in Φ. The sets V_ϕ for $\phi \in \Phi^{\max}$ are pairwise disjoint (by (F1)), and the graph $\overline{\Gamma}^0$ can be directly obtained from Γ^0 by simultaneously trimming the fragments in $\Phi^{\max}$.

In the next section we will take advantage of the relationship between r-paths in $\overline{\Gamma}^0$ and shortest r-paths in (Γ, ℓ) indicated in (A2). Another important property of $\overline{\Gamma}^0$ is as follows.

Lemma 7.4 *If the length* $\ell(C)$ *of every cycle* C *in* Γ *is positive, then* $\overline{\Gamma}^0$ *is acyclic. In particular,* $\overline{\Gamma}^0$ *is acyclic if all arc lengths are positive.*

Proof. Suppose $\overline{\Gamma}^0$ contains a (not necessarily regular) simple cycle $\overline{C}$. In view of (20), $\overline{C}$ is extendable to a cycle C of Γ^0 compatible with Φ. Then $\chi^C \cdot \chi_\phi = 0$ for all $\phi \in \Phi$. This implies that the original length $\ell(C)$ and the reduced length $\ell^\xi_\pi(C)$ are the same (since the changes in ℓ^ξ_π due to π cancel out as we go around the cycle). Since all arcs of C have zero reduced length, $\ell(C) = \ell^\xi_\pi(C) = 0$. This contradicts the hypotheses of the lemma. ■

8 Reduction to an Acyclic Network and Special Cases

We continue the description of the shortest blocking IS-flow method (SBFM) for solving the maximum IS-flow problem in a network $N = (G = (V, E), u)$ begun in Section 6. Let f be a current IS-flow in N. We show that the task of finding a shortest blocking IS-flow g in (G^+, u_f) (step (P1) of a phase of SBFM) reduces to finding a totally blocking IS-flow in an acyclic network.

Build the split-graph $\Gamma = S(G^+, u_f)$ and apply the above shortest regular path algorithm to Γ with the *all-unit* length function ℓ on the arcs. It constructs ϕ, Φ, ξ as in Theorems 7.2 and 7.3, taking $O(m)$ time (since $L = 1$). SRA also constructs the trimmed 0-graph $\overline{\Gamma}^0$, the main object we will deal with. By Lemma 7.4, $\overline{\Gamma}^0$ is acyclic. Also the following property takes place.

Lemma 8.1 *Let* $a \in E^+$ *be an arc with* $u_f(a) > 1$*, and let* a_1, a_2 *be the corresponding split-arcs in* Γ*. Then* $\ell^\xi_\pi(a_1) = \ell^\xi_\pi(a_2)$*. Moreover, none of* a_1, a_2 *can be the base or anti-base arc of any fragment in* Φ*.*

Proof. Since a_1, a_2 are parallel arcs, for each $\phi \in \Phi$, a_1 enters (resp. leaves) V_ϕ if and only if a_2 enters (resp. leaves) V_ϕ. This implies that $\ell^\xi_\pi(a_1) \neq \ell^\xi_\pi(a_2)$ can happen only if one of a_1, a_2 is the base or anti-base arc of some fragment in Φ. Suppose $a_1 \in \{e_\phi, e'_\phi\}$ for some $\phi \in \Phi$ (the case $a_2 \in \{e_\phi, e'_\phi\}$ is similar). Then $\ell^\xi_\pi(a_1) = 0$ (by (A1) in Theorem 7.3). Using property (F2) from Section 7 (valid as Φ is well-nested), one can see that a_2 is not the base or anti-base arc of any fragment in Φ. Therefore, $\chi_\psi(a_2) \leq \chi_\psi(a_1)$ for all $\psi \in \Phi$, yielding $\ell^\xi_\pi(a_2) \leq \ell^\xi_\pi(a_1)$. Moreover,

the latter inequality is strict because $\chi_\phi(a_2) = -1 < 1 = \chi_\phi(a_1)$ and $\xi(\phi) > 0$. Now $\ell^\xi_\pi(a_1) = 0$ implies $\ell^\xi_\pi(a_2) < 0$, contradicting (22). ■

Let $E^0 \subseteq E^+$ be the set of (images of) zero reduced length arcs of Γ. Lemma 8.1 implies that the base arc e_ϕ of each fragment $\phi \in \Phi$ in Γ is generated by an arc $e \in E^0$ with $u_f(e) = 1$. We can identify these e and e_ϕ and consider ϕ as a fragment of G^+ as well. One can see that $\overline{\Gamma}^0$ is precisely the split-graph for $(\overline{H}, \overline{h})$, where $\overline{H} = (\overline{V}, \overline{E}^0)$ is obtained from $H = (V, E^0)$ by trimming the maximal fragments in Φ, and $\overline{h}$ is the restriction of u_f to $\overline{E}^0$.

Based on the property of each fragment to have unit capacity of the base arc, we reduce step (P1) to the desired problem, namely:

(B) *Find a totally blocking IS-flow in* $(\overline{H}, \overline{h})$.

To explain the reduction, suppose we have found a solution $\overline{g}$ to (B). For each maximal fragment ϕ in Φ with $e_\phi \in \mathrm{supp}(\overline{g})$, we have $\overline{g}(e_\phi) = 1$; therefore, exactly one unit of flow goes out of the head of e_ϕ, through an arc $a \in \overline{E}^0$ say. We choose the path $Q = Q_\phi(a)$ as in (20) to connect e_ϕ and a in (the subgraph on V_ϕ of) H and then push a unit of flow through Q and a unit of flow through the symmetric path Q'. Doing so for all maximal fragments ϕ, we extend $\overline{g}$ to an IS-flow g in (H, h), where h is the restriction of u_f to E^0. Moreover, g is a shortest blocking IS-flow in (G^+, u_f).

Indeed, the fact that the chosen paths Q have zero reduced length and are compatible with Φ implies that a symmetric decomposition of g consists of shortest u_f-regular paths (cf. (A2) in Theorem 7.3); so g is shortest. Also G^+ cannot contain a $(u_f - g)$-regular s to s' path R of length $g(E^+)/|g|$. For such an R would be a path in H compatible with Φ (in view of Theorem 7.2); then the arcs of R occurring in $\overline{H}$ should form an $(\overline{h} - \overline{g})$-regular s to s' path in it, contrary to the fact that $\overline{g}$ is totally blocking.

Since each path $Q_\phi(a)$ is constructed in $O(|V_\phi|)$ time (by (20)), and the sets V_ϕ of maximal fragments ϕ are pairwise disjoint, the reduction to (B) takes linear time.

Lemma 8.2 *A totally blocking IS-flow in* $(\overline{H}, \overline{h})$ *can be extended to a shortest blocking IS-flow in* (G^+, u_f)*, in* $O(m)$ *time.* ■

Corollary 8.3 *SBFM solves the maximum IS-flow problem in* $O(qT(n, m) + qm)$ *time, where* q *is the number of phases* ($q \leq n$) *and* $T(n, m)$ *is the time needed to find a totally blocking IS-flow in an acyclic network with at most* n *nodes and* m *arcs.*

Clearly $T(n, m)$ is $O(m^2)$, as a totally blocking flow can be constructed by $O(m)$ applications of the regular reachability algorithm; this is slower compared with the phase time $O(nm)$ in Dinits' algorithm. However, we shall show in the next section that problem (B) can be solved in $O(nm)$ time as well. Moreover, the bound will be better for important special cases.

Next we estimate the number of phases. For the standard max-flow problem, the number of phases of Dinits' algorithm becomes significantly less than n in the cases of unit arc capacities and unit "node capacities". To combine these into one case, given a network $N = (G = (V, E), u)$

with integer capacities u, for a node $x \in V$, define the *transit capacity* $u(x)$ to be the minimum of values $\sum_{y:(x,y)\in E} u(x,y)$ and $\sum_{y:(y,x)\in E} u(y,x)$. Define

$$\Delta := \Delta(N) := \sum(u(x) : x \in V - \{s, s'\}).$$

As shown in [8, 21]), the number q of phases of the blocking flow method does not exceed $2\sqrt{\Delta}$. In particular, if $u \equiv 1$ then $q = O(\sqrt{m})$, and if the transit capacities $u(x)$ of all nodes $x \neq s, s'$ (*inner* nodes) are ones, e.g., in the case arising from the bipartite matching problem, then $q = O(\sqrt{n})$.

A similar argument works for skew-symmetric networks (see also [11] for a special case).

Lemma 8.4 *The number of phases of SBFM is at most* $\min\{n, 2\sqrt{\Delta}\}$.

Proof. After performing $d := \sqrt{\Delta}$ phases, the r-distance from s to s' in the network $N' = (G^+, u_f)$ for the current IS-flow f becomes greater than d, by Lemma 6.1. Let f^* be a maximum IS-flow in N, and let g be defined as in the proof of Theorem 3.3. Then g is a feasible IS-flow in N' and $|g| = |f^*| - |f|$. We assert that $|g| \leq d$, which immediately implies that the number of remaining phases is at most $d/2$, thus proving the lemma. To see this, take a symmetric decomposition $\mathcal{D}$ of g consisting of elementary flows (P, P', δ) with $\delta = 1$. Let $\mathcal{D}'$ be the family of s to s' paths P, P' in $\mathcal{D}$; then $|\mathcal{D}'| \geq |g|$. It is easy to see that $u_f(x) = u(x)$ for each inner node x. Each path in $\mathcal{D}'$ contains at least d inner nodes, and therefore, it uses at least d units of the total transit capacity of inner nodes of N'. So we have $d|\mathcal{D}'| \leq \Delta(N)$. This implies $|g| \leq d$. ■

9 Finding a Totally Blocking IS-Flow in an Acyclic Network

Our aim is to show the following.

Theorem 9.1 *A totally blocking IS-flow in an acyclic skew-symmetric graph with $O(n)$ nodes, $O(m)$ arcs, and unit arc capacities can be found in $O(n + m)$ time.*

This together with Corollary 8.3 and Lemma 8.4 yields the following result for the shortest blocking IS-flow method.

Corollary 9.2 *In case of a network N with unit arc capacities, SBFM can be implemented so that it finds a maximum IS-flow in $O(m\sqrt{\Delta(N)})$ time (assuming $n = O(m)$). In particular, if the indegree or outdegree of each node is at most one, then the running time becomes $O(\sqrt{n}m)$.*

In the second half of this section we will extend Theorem 9.1 to general capacities, in which case the phase time will turn into $O(nm)$, similarly to Dinits' algorithm.

For convenience we keep the original notation for the network in question. Let $G = (V, E)$ be a skew-symmetric *acyclic* graph with source s and the capacity $u(e) = 1$ of each arc $e \in E$. One may assume that each node belongs to a path from s to $\sigma(s)$.

First of all we make a reduction to the *maximal balanced path-set problem (MBP)* stated in the Introduction. Since G is acyclic, one can assign, in linear time, a potential function $\pi : V \to \mathbb{Z}$ which is *antisymmetric* ($\pi(x) = -\pi(\sigma(x))$ for each $x \in V$) and *increasing* on the arcs ($\pi(y) > \pi(x)$

for each $(x, y) \in E$). (Indeed, a function $q : V \to \mathbb{Z}$ increasing on the arcs is constructed, in linear time, by use of the standard topological sorting. Now set $\pi(v) := q(v) - q(\sigma(v))$, $v \in V$.) Subdivide each arc (x, y) with $\pi(x) < 0$ and $\pi(y) > 0$ into two arcs (x, z) and (z, y) and assign zero potential to z. The new graph G, with $O(m)$ nodes and $O(m)$ arcs, is again skew-symmetric, and the problem remains essentially the same.

Let Γ be the subgraph of the new G induced by the nodes with nonnegative potentials. Then $\Gamma \cup \sigma(\Gamma) = G$ and $\Gamma \cap \sigma(\Gamma) = (Z, \emptyset)$, where Z is the self-symmetric set of zero potential nodes of G. Also Γ contains $\sigma(s)$.

Clearly every s to $\sigma(s)$ path P of G meets Z at exactly one node z, which subdivides P into an s to z path R' in $\sigma(\Gamma)$ and a z to $\sigma(s)$ path Q in Γ. Then P is regular if and only if $\sigma(R')$ and Q are arc-disjoint. Conversely, let Q, R be two arc-disjoint Z to $\sigma(s)$ paths in Γ beginning at symmetric nodes in Z. Then the concatenation $\sigma(Q) \cdot R$ (as well as $\sigma(R) \cdot Q$) is a regular s to $\sigma(s)$ path of G if and only if Q and R are arc-disjoint.

This shows that our particular totally blocking IS-flow problem is reduced, in linear time, to MBP with $\Gamma, \sigma(s), Z$ (in fact, the problems are equivalent). Theorem 9.1 is implied by the following.

Theorem 9.3 *MBP is solvable in linear time.*

We devise an algorithm for MBP and prove Theorem 9.3. Let the input of MBP consist of an acyclic graph $\Gamma = (X, U)$, a sink t and a source set Z with a map (involution) $\sigma : Z \to Z$ giving a partition of Z into pairs. We say that two arc-disjoint Z to t paths Q, R beginning at "symmetric" sources $z, \sigma(z)$ form a *good pair*, and say that a collection of pairwise arc-disjoint Z to s' paths in Γ is a *balanced path-set* if its members can be partitioned into good pairs. So the task is to find a maximal (or "blocking") balanced path-set.

Each iteration of the algorithm will reduce the arc set of Γ and, possibly, the set Z, and we sometimes will use index i for objects in the input of i-th iteration. So $\Gamma_1 = (X_1, U_1)$ is the initial graph. Without loss of generality, one may assume that initially each source has zero indegree and

(C1) each node of Γ lies on a path from Z to t,

and will maintain these properties during the algorithm.

The iteration input will include a path D from a certain node of Γ to t, called the *pre-path*. Initially, D is trivial: $D = t$. The nodes of Γ not in $Z \cup \{t\}$ are called *inner*. The current Γ may contain special inner nodes, called *complex* ones. They arise when the algorithm shrinks certain subgraphs of Γ; the initial graph has no complex nodes. The adjacency structure of Γ is given by double-linked lists I_x and O_x of the incoming and outgoing arcs, respectively, for each node x. The arc set of a path P is denoted by $E(P)$.

An i-th iteration begins with extending D to a Z to t path P in a natural way; this takes $O(|P| - |D|)$ time. Let z be the first node of P. Then we try to obtain a good pair by constructing a path from $z' = \sigma(z)$ to t, possibly rearranging P. By standard arguments, a good pair for z, z' exists if and only if there exists a path A from z' to t, with possible backward arcs, in which the

forward arcs belong to $U-E(P)$ and the backward arcs belong to $E(P)$, called an *augmenting path* w.r.t. P. For certain reasons, we admit A to be self-intersecting in nodes (but not in arcs). Once A is found, the symmetric difference $E(P)\triangle E(A)$ gives a good pair Q, R (taking into account that Γ is acyclic).

To search for an augmenting path, we replace each arc $e = (x, y) \in E(P)$ by the reverse arc $\overline{e} = (y, x)$; let $\overline{\Gamma} = (X, \overline{U})$ be the resulting graph, and $\overline{P}$ the t to Z path reverse to P. Thus, we have to construct a (directed) path from z' to t in $\overline{\Gamma}$ or establish that it does not exist.

To achieve the desired time bound, we apply a variant of depth first search which we call here *transit depth first search (TDFS)* (such a search procedure was applied in [19]). The difference from the standard depth first search (DFS) is as follows. When scanning a new outgoing arc (x, y) in the list O_x of a current node x, if y has already been visited, then DFS stays at x. In contrast, TDFS moves from x to y, making y the new current node. Both procedures maintain the stack of arcs traversed only in forward direction and ordered by the time of their traversal. If all outgoing arcs of the current node x are already scanned, then the last arc (w, x) of the stack is traversed in backward direction and w becomes the new current node. We refer to the path determined by the stack, from the initial node to the current one, as the *active path*. Note that in case of TDFS the active path may be self-intersecting (while it is simple in DFS).

We impose the condition that the outgoing arc lists of $\overline{\Gamma}$ are arranged so that

(C2) for each node $x \neq z$ of $\overline{P}$, the arc $\overline{e}$ of $\overline{P}$ leaving x is the *last* element of $\overline{O}_x$.

This guarantees that TDFS would scan $\overline{e}$ after all other outgoing arcs of x (i.e., the arcs of $\overline{P}$ are ignored as long as possible).

At an iteration, we apply TDFS to $\overline{\Gamma}$ starting from z' as above. The search terminates when either it reaches t or it returns to z' having all arcs of $O_{z'}$ traversed. In the first case (*breakthrough*) the final active path $\overline{A}$ in $\overline{\Gamma}$ determines the desired augmenting path A in Γ, and we create a good pair Q, R as described above. In the second case, the non-existence of a good pair for the given z, z' is declared. Consider both cases.

Breakthrough case. Delete from Γ the arcs of Q, R and then delete all the nodes and arcs that are no longer contained in Z to t paths (thus maintaining (C1)). This is carried out by an obvious *cleaning procedure* in $O(q)$ time, where q is the number of arcs deleted. If Q or R contains a complex node, the iteration finishes by transforming Q, R into a good pair of paths of the initial graph; this is carried out by a *path expansion procedure* which will be described later. The obtained Γ, Z form the input of the next iteration, and the new pre-path D is assigned to be the trivial path t. If Γ vanishes, the algorithm terminates. The following observation is crucial for estimating the time bound.

Lemma 9.4 *Let q be the number of arcs deleted at an iteration with a breakthrough. Then, excluding the path expansion procedure if applied, the iteration runs in $O(q)$ time.*

Proof. Let $\overline{W}$ be the set of arcs of $\overline{\Gamma}$ traversed by TDFS on the iteration, and W the corresponding set in Γ, i.e., $W = \{e \in U : e \in \overline{W} \text{ or } \overline{e} \in \overline{W}\}$. The iteration runs in $O(q + |W|)$ time, taking into account that each arc of P not in $Q \cup R$ is contained in W. Therefore, it suffices to show that no

arc from W remains in the new graph Γ. Suppose this is not so. Then there is a Z to t path L of the old Γ that meets W but not $E(Q) \cup E(R)$ (as the arcs of $Q \cup R$ are deleted). Let $e = (x, y)$ be the *last* arc of L in W. Let $b = (y, w)$ be the next arc of L (it exists since $y = t$ would imply that e is in A but not in P, whence e belongs to $Q \cup R$). Then $b \notin W \cup E(Q) \cup E(R)$, by the choice of e. Two cases are possible.

(i) e is in P. Then $\overline{e} = (y, x) \in \overline{W}$. According to condition (C2), at the time TDFS traversed $\overline{e}$ from y to x all arcs of $\overline{\Gamma}$ leaving y had already been traversed. So b is not in $\overline{\Gamma}$, implying $b \in E(P) - W$. Then b is in $Q \cup R$; a contradiction.

(ii) e is not in P. Then $e \in \overline{W}$ and e does not lie on the final active path $\overline{A}$ (otherwise e is in $Q \cup R$). Therefore, TDFS traversed e in both directions. To the time of traversal of e in backward direction, from y to x, all arcs of $\overline{\Gamma}$ leaving y have been traversed (at this point the difference between TDFS and DFS is important). So b is not in $\overline{\Gamma}$, whence $b \in E(P)$. Now $\overline{b} \notin \overline{W}$ implies that b is in $Q \cup R$; a contradiction. ■

Non-breakthrough case. Let Y be the set of nodes visited by TDFS. Then no arc of $\overline{\Gamma}$ leaves Y. Therefore, in view of (C1),

(25) the set of arcs of Γ leaving Y consists of a unique arc $a = (v, w)$, this arc lies on P, and the nodes of the part of P from z to v are contained in Y.

Since no arc of Γ enters Z, we also have

$$Y \cap Z = \{z, z'\}. \tag{26}$$

We reduce Γ by shrinking its subgraph $\Gamma_Y = (Y, U_Y)$ induced by Y into one node; the formed *complex* node v_Y is identified with v. We call v the *root* of Γ_Y and store Γ_Y. The list of arcs entering v_Y in the new graph is produced by simply merging the lists I_x for $x \in Y$ from which the arcs occurring in Γ_Y are explicitly removed, using the lists O_y for $y \in Y$. (We do not need to correct the outgoing arc lists O_x for $x \notin Y$ explicitly, as we explain later.) Thus, to update Γ takes time linear in $|U_Y|$. By (25), a is the only arc leaving v_Y in Γ.

From (C1) and (25) it follows that

(27) the new graph Γ is again acyclic, and for each $x \in Y$, there is a path $P_Y(x)$ from x to the root v in Γ_Y.

In view of (26), the set Z is updated as $Z := Z - \{z, z'\}$. If there is at least one arc entering v_Y, then the new graph Γ and set Z satisfy (C1) and form the input of the next iteration. The new pre-path D is assigned to be the part of P from the formed complex node v_Y to t. If no arc enters v_Y, we finish the current iteration by removing the nodes and the arcs not contained in Z to t paths. This further reduce Γ and may reduce Z and shorten D.

One can see that the set U_Y is exactly W. This and the construction of pre-paths imply the following.

Lemma 9.5 *Let q be the number of arcs deleted by an i-th iteration without a breakthrough. Then the iteration runs in $O(q + \max\{0, |D_{i+1}| - |D_i|\})$ time.* ■

As mentioned above, we do not need to explicitly correct the outgoing arc lists O_x for $x \notin Y$ (this would be expensive). Let $\mathcal{V}$ be the current set of all complex nodes created from the beginning of the algorithm. We take advantage of the following facts. First, the elements of $\mathcal{V}$ that are nodes of the current graph (the *maximal* complex nodes) lie on the current pre-path D. Second, at an iteration with a breakthrough, all complex nodes are removed. Third, at an iteration without a breakthrough, the subgraph Γ_Y forming the new complex node v_Y contains a subpath of P from its beginning node (by (25)), and the cleaning procedure (if applied at the iteration) deletes a part of the updated P from its beginning node as well. Therefore, one can store $\mathcal{V}$ as a tree in a natural way and use the *disjoint set union* data structure from [13] to maintain $\mathcal{V}$. This enables us to efficiently access the head v_Y of any arc $e = (x, v_Y)$ when e is traversed by TDFS (with $O(1)$ amortized time per one arc).

To complete the algorithm description, it remains to explain the *path expansion procedure* to be applied when an iteration with a breakthrough finds paths Q, R contained complex nodes. It proceeds in a natural way by recursively expanding complex nodes occurring in the current Q, R into the corresponding paths $P_Y(x)$ as in (27) and building $P_Y(x)$ into Q or R (this takes $O(|P_Y(x)|)$ time). The arc sets of subgraphs Γ_Y extracted during the algorithm are pairwise disjoint, so the total time for all applications of the procedure is $O(m)$.

Thus, we can conclude from Lemmas 9.4 and 9.5 that the algorithm runs in $O(m)$ time, yielding Theorem 9.3.

In the rest of this section we extend the above approach and algorithm (*Algorithm 1*) to a general case of acyclic (G, u). The auxiliary graph $\Gamma = (X, U)$ and the set Z are constructed as above, and the capacity $u(e)$ of each arc $e \in U$ is defined in a natural way. We call an integer Z to t flow g in (Γ, u) *balanced* if the flow values out of "symmetric" sources are equal, i.e.,

$$\mathrm{div}_g(z) = \mathrm{div}_g(\sigma(z)) \quad \text{for each } z \in Z,$$

and *blocking balanced* if there exists no balanced flow g' satisfying $g \neq g' \geq g$ (taking into account that Γ is acyclic). Then the problem of finding a totally blocking IS-flow in (G, u) is reduced to *problem BBF*: find a balanced blocking flow for Γ, u, Z, t.

Algorithm 2 will find a balanced blocking flow g in the form $g = \alpha_1\chi^{Q_1} + \alpha_1\chi^{R_1} + \ldots + \alpha_r\chi^{Q_r} + \alpha_r\chi^{R_r}$, where each α_i is a positive integer, Q_i is a path from some $z \in Z$ to t, and R_i is a path from $\sigma(z)$ to t. It iteratively constructs pairs Q_i, R_i for current Γ, u, Z, assigns the weight α_i to them as large as possible, and accordingly reduces the current capacities as $u := u - \alpha_i\chi^{Q_i} - \alpha_i\chi^{R_i}$. All arc capacities in Γ are positive: once the capacity of an arc becomes zero, this arc is immediately deleted from Γ.

Each pair Q_i, R_i is constructed as in Algorithm 1 when it is applied to the corresponding *split-graph* $S = S(\Gamma, u)$. More precisely (cf. Section 3), S is formed by replacing each arc $e = (x, y)$ of Γ by two parallel arcs (*split-mates*) e_1, e_2 from x to y with the capacities $\lceil u(e)/2 \rceil$ and $\lfloor u(e)/2 \rfloor$, respectively. When $u(e) = 1$, e_2 vanishes in S, and e_1 is called *critical*. The algorithm maintains S explicitly. The desired pair Q_i, R_i in (Γ, u) is determined by a good pair in S in a natural way.

The main part of an iteration of Algorithm 2 is a slight modification of an iteration of Algorithm 1. The difference is the following. While Algorithm 1 deletes *all* arcs of the paths Q, R

found at an iteration, Algorithm 2 deletes only a *nonempty subset* B of arcs in $Q \cup R$ (concerning the graph S) including all critical arcs in these paths. One may think that Algorithm 2 essentially treats with a graph S (ignoring (Γ, u)) in which some disjoint pairs of parallel arcs (analogs of split-mates) are distinguished and the other arcs are regarded as critical, and at each iteration, the corresponding subset $B \subseteq E(Q) \cup E(R)$ to be deleted is given by an oracle. Emphasize that the unique arc leaving a complex node is always critical. Therefore, each complex node in $Q \cup R$ will be automatically removed. Computing α_i's and other operations of the algorithm beyond the work with the graph S do not affect the asymptotic time bound.

We now estimate the complexity of an iteration of Algorithm 2. In case without a breakthrough, properties (25),(26),(27) and Lemma 9.5 (with S instead of Γ) remain valid. Note that the arc a in (25) is critical (since it is a unique arc leaving Y); therefore, the arc leaving the created complex node is critical. Our analysis of the breakthrough case involves the subset $\overline{W}_2 \subset \overline{W}$ of arcs traversed by TDFS in both directions (where $\overline{W}$ is the set of all traversed arcs in the corresponding auxiliary graph $\overline{S}$). Let W_2 be the corresponding set in S.

Lemma 9.6 *Suppose an iteration of Algorithm 2 results in a breakthrough. Let $e = (x, y)$ be an arc of S such that $e \in W_2$ or $e \in E(P) \cap W$. Then any x to t path L in S starting with the arc e contains a critical arc in $Q \cup R$ (and therefore, e vanishes in the new graph S).*

Proof. Suppose this is not so and consider a counterexample (e, L) with $|L|$ minimum. Let $\overline{A}_0$ be the active path in $\overline{S}$ just before the traversal of e or $\overline{e}$ from y to x, and A_0 the corresponding (undirected) path in S. At that time, for the set $\overline{O}_y$ of arcs of $\overline{S}$ leaving y,

(28) all arcs in $\overline{O}_y$ except $\overline{e}$ (in case $e \in E(P)$) are already traversed

(in view of condition (C2)). Let $b = (y, w)$ be the second arc of L (existing as $y = t$ is impossible). By (28), if b is not in P, then $b \in W$. Also $b \notin W_2$ and $b \notin E(P) \cap W$ (otherwise the part of L from y to t would give a smaller counterexample). This implies that b belongs to $Q \cup R$, and therefore, b is not critical. Let b' be the split-mate of b. Considering the path starting with b' and then following L from w to t (which is smaller than L), we similarly conclude that $b' \notin W_2$ and $b' \notin E(P) \cap W$. To come to a contradiction, we proceed as follows.

The fact that S is acyclic implies that the symmetric difference (on the arcs) of P and A_0 is decomposed into a path from Z to t and a path from Z to y; therefore, $E(P) \triangle E(A_0)$ contains *at most one* arc a leaving y. This and (28) imply that all arcs in $O_y \cap \overline{O}_y$ except, possibly, a have been traversed twice; so they are in W_2. Hence, one of b, b' must be in P; let for definiteness $b \in E(P)$ (then $b' \notin E(P)$).

Now $b \notin W$ implies $b \in E(P) - E(A_0)$, and $b' \in W - W_2$ implies $b' \in E(A_0) - E(P)$. Thus, both arcs b, b' leaving y are in $E(P) \triangle E(A_0)$; a contradiction. ■

The running time of an iteration with a breakthrough is $O(|P| + |W| + q)$, where q is the number of arcs deleted from S. Lemma 9.6 allows us to refine this bound as $O(|Q| + |R| + q)$. Combining this with Lemma 9.5, we can conclude that, up to a constant factor, the total time of Algorithm 2 is bounded from above by m plus the sum Σ of lengths of paths $Q_1, R_1, \ldots, Q_r, R_r$

in the representation of the flow g constructed by the algorithm. Since $|Q_i|, |R_i| \leq n$ and $r \leq 2m$ (as each iteration decreases the arc set of S), Σ is $O(nm)$. Also Σ does not exceed the sum of the transit capacities $u(x)$ of inner nodes x of Γ (assuming, without loss of generality, that no arc goes from s to s'). Thus, Theorem 9.1 is generalized as follows.

Theorem 9.7 *For an acyclic capacitated skew-symmetric network N with $O(n)$ nodes and $O(m)$ arcs, a totally blocking IS-flow can be found in $O(\min\{m + \Delta(N), nm\})$ time.*

Together with Corollary 8.3 and Lemma 8.4, this yields the desired generalization.

Corollary 9.8 *SBFM can be implemented so that it finds a maximum IS-flow in an arbitrary skew-symmetric network N in $O(\min\{n^2m, \sqrt{\Delta(N)}(m + \Delta(N))\})$ time.*

10 Applications to Matchings

Apply the reduction of the maximum u-capacitated b-matching problem (CBMP) in a graph $G' = (V', E')$ to the maximum IS-flow problem in a network $N = (G = (V, E), \sigma, u, s)$; see Section 2. The best time bound for a general case of CBMP is attained by applying the algorithm of Section 5. Theorem 5.1 implies the following.

Corollary 10.1 *CBMP can be solved in $O(M(n, m) + nm)$ time, where $n := |V'|$ and $m := |E'|$.*

When the input functions u, b in CBMP are small enough, the transit capacities of nodes in N become small as well. Then the application of the shortest blocking IS-flow method may result in a competitive or even faster algorithm for CBMP. Let the capacities of all edges of G' be ones. We have $\Delta(N) = O(m)$ in general, and $\Delta(N) = O(n)$ if b is all-unit. Then Corollary 9.2 yields the same time bounds as in [12, 25] for the corresponding cases.

Corollary 10.2 *SBFM (with the fast implementation of a phase as in Section 9) solves the maximum degree-constrained subgraph (or b-factor) problem in $O(m^{3/2})$ time and solves the maximum matching problem in a general graph in $O(\sqrt{n}m)$ time.*

Feder and Motwani [9] elaborated a clique compression technique and used it to improve the $O(\sqrt{n}m)$ bound for the maximum bipartite matching problem to $O(\sqrt{n}m \log(n^2/m)/\log n)$. We explain how to apply a similar approach to a special case of MSFP, lowering the bound for dense nonbipartite graphs. We need a brief review of the method in [9].

Let $H = (X, Y, E)$ be a bipartite digraph, where $E \subseteq X \times Y$, $|X| = |Y| = n$ and $|E| = m$. A *(bipartite) clique* is a complete bipartite subgraph $(A, B, A \times B)$ of H, denoted by $C(A, B)$. Define the *size* $s(C)$ of $C = C(A, B)$ to be $|A| + |B|$. A *clique partition* of H is a collection $\mathcal{C}$ of cliques whose arc sets form a partition of E; the *size* $s(\mathcal{C})$ of $\mathcal{C}$ is the sum of sizes of its members.

Let a constant $0 < \delta < 1/2$ be fixed. Then a clique $C(A, B)$ of H is called a *δ-clique* if $|A| = \lceil n^{1-\delta} \rceil$ and $|B| = \lfloor \delta \log n / \log(2n^2/m) \rfloor$. It is shown in [9] that a δ-clique exists.

The *clique partition algorithm* in [9] finds a δ-clique C_1 in the initial graph $H_1 = (X, Y, E =: E_1)$ and deletes the arcs of C_1, obtaining the next graph $H_2 = (X, Y, E_2)$. Then it finds a δ-clique

C_2 (concerning the number of arcs of H_2) and delete the arcs of C_2 from H_2, and so on while the number of arcs of the current graph is at least $2n^{2-\delta}$ and the Y-part of a δ-clique is nonempty. The remaining arcs are partitioned into cliques consisting of a single arc each. So the cliques C_i extracted during the algorithm form a clique partition. The running time of the algorithm is estimated as the sum of bounds $\tau(C_i)$ on the time to extract the cliques C_i plus a time bound τ' to maintain a certain data structure (so-called neighborhood trees). One shows that

(29) the algorithm runs in $O(\sqrt{n}m\beta)$ time and finds a clique partition $\mathcal{C}$ of H such that $s(\mathcal{C}) = O(m\beta)$, where $\beta := \log(n^2/m)/\log n$.

Suppose we wish to find a maximum matching in a bipartite graph or, equivalently, to find a maximum integer flow from s to t in a digraph G with unit arc capacities, node set $X \cup Y \cup \{s, t\}$ and arc set $E \cup (s \times X) \cup (Y \times t)$, where $E \subseteq X \times Y$. One may assume $|X| = |Y| = n$. Using the above algorithm, form a clique partition $\mathcal{C}$ as in (29) for (X, Y, E). Transform each clique $C(A, B)$ in $\mathcal{C}$ into a star by replacing its arcs by a node z, arcs (x, z) for all $x \in A$ and arcs (z, y) for all $y \in B$. There is a natural one-to-one correspondence between the s to t paths in G and those in the resulting graph G^*, and the problem for G^* is equivalent to that for G. Compared with G, the graph G^* has $|\mathcal{C}|$ additional nodes but the number m^* of its arcs becomes $2n + s(\mathcal{C})$, or $O(m\beta)$. Given a flow in G^*, any (simple) augmenting path of length q meets exactly $(q-1)/2$ nodes in $X \cup Y$, and these nodes have unit transit capacities. This implies that Dinits' algorithm has $O(\sqrt{n})$ phases (arguing as in [8, 21]). Since each phase takes $O(m^*)$ time, the whole algorithm runs in $O(\sqrt{n}m\beta)$ time, as desired.

Now suppose $H = (X, Y, E, \sigma)$ is a skew-symmetric bipartite graph without parallel arcs, where the sets X and Y are symmetric each other. We modify the above method as follows. Note that any two symmetric cliques in H are disjoint (otherwise some $x \in X$ is adjacent to $\sigma(x)$, implying the existence of two arcs from x to $\sigma(x)$). We call a clique partition $\mathcal{C}$ *symmetric* if $C \in \mathcal{C}$ implies $\sigma(C) \in \mathcal{C}$. An iteration of the *symmetric clique partition algorithm* works as in the previous algorithm, searching for a δ-clique C' in the current H, but then deletes the arcs of *both* C' and $\sigma(C')$. Let the algorithm construct a partition $\mathcal{C}'$ consisting of cliques $C'_1, \sigma(C'_1), \ldots, C'_r, \sigma(C'_r)$ obtained in this order.

To estimate the size of $\mathcal{C}'$ and the running time, imagine we would apply the previous algorithm to our H (ignoring the fact that H is skew-symmetric). Let the resulting partition $\mathcal{C}$ be formed by cliques $C_1, \ldots, C_q$ (in this order). Note that for a bipartite graph with n nodes and m arcs, both the number $e(C)$ of arcs of a δ-clique C and its size $s(C)$ are computed uniquely, and these are monotone functions in m, as well as the above-mentioned time bound $\tau(C)$ (indicated in [9]). Moreover, one can check that $m' \leq m$ and $e(C') \neq 0$ imply $m' - 2e(C') \leq m - e(C)$, where C' is a δ-clique in a graph with n nodes and m' arcs. Using these, we can conclude that $r \leq q$ and that for $i = 1, \ldots r$, $s(C'_i) \leq s(C_i)$ and $\tau(C'_i) \leq \tau(C_i)$. Then $s(\mathcal{C}') \leq 2s(\mathcal{C})$, implying $s(\mathcal{C}) = O(m\beta)$, by (29). Also the time of the modified algorithm is $O(\sqrt{n}m\beta)$ (by (29) and by the fact that the above bound τ' remains the same) plus the time needed to treat the symmetric cliques $\sigma(C'_i)$, which is $O(m)$.

Finally, the graph H^* obtained from H by transforming the cliques $C'_1, \sigma(C'_1), \ldots, C'_r, \sigma(C'_r)$ into stars has a naturally induced skew-symmetry. By the above argument, H^* has $O(m\beta)$ arcs, and computing H^* takes $O(\sqrt{n}m\beta)$ time. Apply such a transformation to the input graph of MSFP arising from an instance of the maximum matching problem. Arguing as in the bipartite matching case above and as in the proof of Lemma 8.4, we conclude with the following.

Theorem 10.3 *A maximum matching in a general graph with n nodes and m edges can be found in $O(\sqrt{n}m \log(n^2/m)/\log n)$ time.*

References

[1] R.P. Anstee. An algorithmic proof of Tutte's f-factor theorem. *J. of Algorithms* **6** (1985) 112–131.

[2] R.P. Anstee. A polynomial algorithm for b-matchings: An alternative approach. *Information Proc. Letters* **24** (1987) 153–157.

[3] N. Blum. A new approach to maximum matching in general graphs. In *Automata, Languages and Rpogramming* [Lecture Notes in Comput. Sci. 443] (Springer, Berlin, 1990), pp. 586–597.

[4] E. A. Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Math. Dokl.* **11** (1970) 1277–1280.

[5] J. Edmonds. Paths, trees and flowers. *Canada J. Math.* **17** (1965) 449–467.

[6] J. Edmonds and E. L. Johnson. Matching, a well-solved class of integer linear programs. In R. Guy, H. Haneni, and J. Schönhein, eds, *Combinatorial Structures and Their Applications*, Gordon and Breach, NY, 1970, pp. 89–92.

[7] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. Assoc. Comput. Mach.* **19** (1972) 248–264.

[8] S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM J. Comput.* **4** (1975) 507–518.

[9] T. Feder and R. Motwani. Clique partitions, graph compression and speeding-up algorithms. In *Proc. 23rd Annual ACM Symp. on Theory of Computing*, 1991, pp. 123–133.

[10] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton Univ. Press, Princeton, NJ, 1962.

[11] C. Fremuth-Paeger and D. Jungnickel. Balanced network flows. Parts I–III. *Networks* **33** (1999) 1–56.

[12] H. N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. *Proc. of STOC* **15** (1983) 448–456.

[13] H. N. Gabow and R. E. Tarjan. A linear-time algorithm for a special case of disjoint set union. *J. Comp. and Syst. Sci.* **30** (1985) 209–221.

[14] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph-matching problems. *J. ACM* **38** (1991) 815–853.

[15] A. V. Goldberg and A. V. Karzanov. Maximum skew-symmetric flows. In P. Spirakis, ed., *Algorithms – ESA '95* (Proc. 3rd European Symp. on Algorithms), *Lecture Notes in Computer Sci.* **979**, 1995, pp. 155–170.

[16] A. V. Goldberg and A. V. Karzanov. Path problems in skew-symmetric graphs. *Combinatorica* **16** (1996) 129–174.

[17] A. V. Goldberg and A. V. Karzanov. Maximum skew-symmetric flows and their applications to b-matchings. *Preprint* 99-043, SFB 343, Bielefeld Universität, Bielefeld, 1999, 25 pp.

[18] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM J. Comput.* **2** (1973) 225–231.

[19] A. V. Karzanov. Èkonomnyĭ algoritm nakhozhdeniya bikomponent grafa [Russian; An efficient algorithm for finding the bicomponents of a graph]. In *Trudy Tret'eĭ Zimneĭ Shkoly po Matematicheskomu Programmirovaniyu i Smezhnym Voprosam* [Proc. of 3rd Winter School on Mathematical Programming and Related Topics], issue **2**. Moscow Engineering and Construction Inst. Press, Moscow, 1970, pp. 343-347.

[20] A.V. Karzanov. Tochnaya otsenka algoritma nakhozhdeniya maksimal'nogo potoka, primenennogo k zadache "o predstavitelyakh" [Russian; An exact estimate of an algorithm for finding a maximum flow, applied to the problem "of representatives"]. In *Voprosy Kibernetiki* [Problems of Cybernetics], volume **3**. Sovetskoe Radio, Moscow, 1973, pp. 66–70.

[21] A.V. Karzanov. O nakhozhdenii maksimal'nogo potoka v setyakh spetsial'nogo vida i nekotorykh prilozheniyakh [Russian; On finding maximum flows in networks with special structure and some applications]. In *Matematicheskie Voprosy Upravleniya Proizvodstvom* [Mathematical Problems for Production Control], volume **5**. Moscow State University Press, Moscow, 1973, pp. 81-94.

[22] W. Kocay and D. Stone. Balanced network flows. *Bulletin of the ICA* **7** (1993) 17–32.

[23] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Reinhart, and Winston, New York, NY., 1976.

[24] L. Lovász and M. D. Plummer. *Matching Theory*. Akadémiai Kiadó, Budapest, 1986.

[25] S. Micali and V.V. Vazirani. An $O(\sqrt{V}E)$ algorithm for finding maximum matching in general graphs. *Proc. of the 21st Annual IEEE Symposium in Foundation of Computer Science*, 1980, pp. 71–109.

[26] A. Schrijver. *Combinatorial Optimization. Polyhedra and Efficiency*, Volume A. (*Algorithms and Combinatorics* **24**), Springer, Berlin and etc., 2003.

[27] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.* **1** (1972) 146–160.

[28] W.T. Tutte. Antisymmetrical digraphs. *Canadian J. Math.* **19** (1967) 1101–1117.

[29] V.V. Vazirani. A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ general graph maximum matching algorithm. In: R. Kannan and W.R. Cunningham, eds., *Integer Programming and Combinatorial Optimization* (Proc. 1st IPCO Conference), University of Waterloo Press, Waterloo, Ontario, 1990, pp. 509–535.