ing that the Russian mathematicians Gel'fand and Shilov present an exhaustive and highly useful discussion of regularization in their volume "Generalized Functions," which is often overlooked.

There are a few minor typographical errors: on p. 426, second paragraph, "section XIV.1.1" should be "section XIX.1.1," and "to solved" in the third paragraph should be "to solve." Similarly, "are closed" in the last paragraph of p. 430 should be "are close."

In summary, the second and revised edition of Chadan and Sabatier's book should prove to be indispensable for students and researchers alike in inverse scattering theory, not only in the strict quantum domain, but also in diverse areas of classical physics since many of the inverse methods are common to both and since these have been treated by the authors with superb skill. The book should also be of immense value for its historical perspective and extensive references, a must for doing good research. Unfortunately, the presentations are at times unnecessarily terse and many problems have been treated in a sort of handbook fashion. These make it difficult for an average reader or for a more physically oriented reader to find the book self-sufficient without having to do extra homework. But then again, that may also be an advantage for the reader. Overall, the book is recommended in highest possible terms.

DILIP N. GHOSH ROY
TRA, Inc.
University of Utah

Потоковые Алгоритмы (Flow Algorithms). *By G. M. Adel'son-Vel'ski, E. A. Dinits, and A. V. Karzanov.* Science, Moscow, 1975. 119 pp. 46 kopeks. No ISBN.

Introduction. This is a review of the book *Flow Algorithms* by Adel'son-Vel'ski, Dinits, and Karzanov, well-known researchers in the area of algorithm design and analysis. This remarkable book, published in 1975, is written in Russian and has never been translated into English. What is remarkable about the book is that it describes many major results obtained in the Soviet Union (and originally published in papers by 1976) that were independently discovered later (and in some cases much later) in the West. The book also contains some minor results that we believe are still unknown in the West. The book is well written and a pleasure to read, at least for someone fluent in Russian. Although the book is fifteen years old and we believe that all the major results contained in it are known in the West by now, the book is still of great historical importance. Hence a complete review is in order.

The apparent fact that, until recently, no Western researcher had looked closely at the book and the underlying papers serves as an important reminder for Western researchers to be aware of the work outside of their normal circles. This fact is especially embarrassing because Dinits and Karzanov were well known in the West for their work on network flows by at least 1976, when the theoretical superiority of their network flow algorithms [10], [16], compared to contemporary ones available in the West, was recognized by Western researchers. (The general feeling at the time was that the significance of this work should have been recognized sooner.) After that, these algorithms became the object of intense studies in the West as well as in the Soviet Union. The book in question, however, remained unknown in the West until recently, when it was "discovered" by the second reviewer.[1]

Except for this introduction, we try to keep the review objective; we give a section-by-section summary of the book. All attributions in the review are as the authors of the book make them.[2] As it is not our purpose to attempt a definitive history of network flow algorithms in the framework of this review, we do not explicitly point out which results appeared first in the Russian literature, nor who independently discovered them in the West. We believe that the information and references provided here will enable informed readers to make their own historical comparisons and attributions.

Although the style of the book is contemporary, some of the material is quite dated because of the progress made during the fifteen years since the book was written. The body of our review simply reflects the book's contents.

[1] It should be noted, however, that the book was not widely circulated in the West; after an extensive search, we found only four U.S. libraries that have it in their catalogs.

[2] We believe, however, that the references in the book are very complete and accurately reflect the literature, both Russian and Western, at the time the book was written.

No statements or conjectures from the book have been altered to reflect current knowledge.[3]

In preparing this review, we also read additional papers in Russian that contained valuable results which were new to us, and which we believe will be new to other Western researchers. We invite the Western algorithms community in general, and the network flow community in particular, to actively read the Russian literature and widely disseminate what you find to be new and interesting.

Notation. Next we state some basic definitions that are needed throughout the review. Our definitions are quite standard, and we assume that the reader is familiar with the concepts discussed here.

A *network* G is a graph (directed unless explicitly stated to be undirected) where each arc (u, v) has an assigned capacity $c(u, v)$. We use U to denote the maximum arc capacity, and we use n and m to denote the number of nodes and arcs, respectively, in the network. We use lg to denote the base-2 logarithm.

Given two nodes s and t (called the *source* and *sink* nodes, respectively) of a directed net work G, an s, t *flow* f is an assignment of nonnegative real values to the arcs such that the following two properties hold:

(1) *Capacity constraint*. For any arc (v, w), $0 \leqq f(v, w) \leqq c(v, w)$.

(2) *Flow conservation*. For any node $v \neq s, t$, $f_{\text{in}}(v) = f_{\text{out}}(v)$ where $f_{\text{in}}(v)$ is the total flow on arcs directed into v, and $f_{\text{out}}(v)$ is the total flow on arcs directed out of v.

The *value* of the flow f is the total flow entering t, i.e., $\sum_v f(v, t)$. A *maximum* s, t *flow* in G is an s, t flow whose value is maximum over all s, t flows. For a given s and t, we use $M(G)$ to denote the value of the maximum flow in G. We use $e_f(v)$ to denote the *excess* of flow f at a node v; the excess is equal to the difference between the incoming and outgoing flows.

If G is undirected, the maximum flow problem on G is obtained by replacing each edge (u, v) with two arcs (u, v) and (v, u), each with capacity $c(u, v)$.

An s, t *cut*$(X, \bar{X})$ in G is a partition of the nodes into two sets X and $\bar{X}$ such that $s \in X$ and $t \in \bar{X}$. The capacity of the cut, denoted $c(X, \bar{X})$, is $\sum_{u \in X, v \in \bar{X}} c(u, v)$.

The *edge-connectivity* of an undirected graph (where all edges have capacity 1) is the smallest capacity of any cut in G. The *node connectivity* is similarly defined for nodes.

In a *multicommodity flow* more than one type of commodity simultaneously flows on the network, and each commodity has a given source and sink. The total flow (of all commodities in both directions) on an arc must obey the capacity constraints, and each commodity must obey the conservation constraints. The multicommodity flow problem comes in two variants, the *feasibility problem* and the *max-*Σ *problem*. In the feasibility problem each source has a given supply of each commodity and each sink has a given demand for each commodity (a node may be a source for one commodity and a sink for another); the problem is to determine whether there is a feasible flow that satisfies all the stated demands for all the commodities. In the max-Σ problem, the objective is to find a flow which maximizes the total amount of all commodities which reach their sinks.

In the *minimum-cost flow problem* each arc (u, v) is given a number $k(u, v)$ (called the cost of arc (u, v)), and the objective is to find a maximum s, t flow of minimum total cost, i.e., a maximum s, t flow minimizing $\sum k(u, v) f(u, v)$.

Final remarks. Before giving the section-by-section review of the book, we would like to remind the reader that the body of the review simply reflects the book's contents. In particular, the attributions are as the authors of the book made them and no statement from the book has been altered to reflect today's state of knowledge.

We also would like to make a comment on citations. Understandably, many citations are to the Russian books and journals; regrettably, most of these do not have English translations. We organize the citations in two groups: English and Cyrillic. Each group is sorted according to the corresponding alphabet. For each Cyrillic reference, we also provide either a reference to the corresponding translation (if we are aware of one), or an English transcription and an English translation of the title. Although our title translations are accurate, we cannot

[3] For a survey of the recent developments in this area, see, e.g., A. V. Goldberg, É. Tardos, and R. E. Tarjan, *Network flow algorithms*, in Path, Flows, and VLSI-layout, Springer-Verlag, Berlin, 1990, pp. 101–164.

guarantee that they will be identical to those of "professional" translators (if a paper will be translated in the future, or if a translation already exists but is unknown to us).

1. The maximum flow problem.

1.1. Networks and network flows. This section of the book defines the basic concepts, such as flows, cuts, etc. It also states a flow decomposition theorem, and gives an algorithm to decompose a flow into a collection of paths and cycles.

1.2. Ford–Fulkerson algorithm for the maximum flow problem. The section starts by introducing several fundamental facts about the maximum flow problem. In particular, it shows that the maximum flow value is equal to the minimum cut capacity (the max-flow, min-cut theorem) and that a flow is maximum if and only if it admits no augmenting path (the augmenting path theorem). Then the authors introduce the *Ford–Fulkerson augmenting path algorithm* (FFA) [2], and give its implementation using the labeling method. They use the algorithm to prove the integrality theorem, which states that a maximum flow problem with integral arc capacities has an integral optimal solution.

1.3. The shortest augmenting path algorithm. This section introduces the *shortest augmenting path algorithm* (SPA) of Edmonds and Karp [1] and Dinits [10]. It is shown that the augmenting path length in SPA is monotone and nondecreasing, and at most m augmenting paths of length k, $1 \leqq k \leqq n - 1$, are found by the algorithm. Thus the number of iterations of the algorithm is at most $(n - 1)m = O(n^3)$. This bound is tight [9]. The section concludes with a discussion of the implementation of the SPA using breadth-first search labeling for finding shortest augmenting paths. The complexity of the resulting method is $O(nm^2)$.

1.4. The layered network algorithm. In the SPA algorithm, the work spent looking for augmenting paths dominates the work spent on augmenting the flow and determines the running time of the algorithm. This motivates the *layered network algorithm* (LNA) of Dinits [10], which can be viewed as a variation of the SPA algorithm. This variation finds shortest augmenting paths in a more efficient way using *layered networks*. The layered network is a union of all shortest paths from s to f. The LNA runs in $O(n^2m)$ time, which is better than the $O(nm^2)$ time for the SPA algorithm. This bound is tight, as shown in [12] (and implicitly in [9]).

The section ends with a discussion of the optimized version of the layered network algorithm. Each phase of this algorithm starts with the correct layered network, but the layered network is not updated after each augmentation. Instead, the network is updated only if a "dead end" is discovered during a search for an augmenting path.

1.5. The blocking flow algorithm. The *blocking flow method* (BFM) due to Karzanov [16], [17] is based on the notion of a *blocking flow*; the task of finding a maximum flow in a network is reduced to the task of finding at most $n - 1$ blocking flows in layered networks. The *preflow method* (PM) uses the concept of a *preflow* to find a blocking flow in such a network in $O(n^2)$ time [16], [17]. This method allows flow excesses at nodes in the middle of the execution of the algorithm, giving it more flexibility compared to the augmenting path method. The preflow method solves the maximum flow problem in $O(n^3)$ time.

1.6. Algorithms with bounds depending on logarithms of capacities. Two algorithms are described whose running time is a polynomial in n, m, and lg U. The first algorithm, due to Edmonds and Karp [1], is a variation of FFA that at each iteration selects the highest capacity augmenting path. At each iteration of this algorithm, the residual flow value decreases by at least a factor of $(1 - 1/m)$. For real-valued capacities, the flow computed by the algorithm converges to a maximum flow. For integral capacities, the algorithm terminates in $m \lg U$ iterations. An $O(n^2)$ algorithm for computing the highest capacity augmenting path is also described. Combined with an $O(n^2)$ algorithm for computing the highest capacity augmenting path, this yields an $O(n^2m \lg U)$ bound on the entire algorithm.

The second algorithm is due to Dinits [11] and uses capacity scaling, an idea that was developed independently by Edmonds and Karp [1] and Dinits [11]. The algorithm works by maintaining the parameter K, which starts at $2^{\lfloor \lg U \rfloor}$ and decreases by a factor of two at each scaling iteration. During such an iteration, all capacities are rounded down to the nearest multiple of K, and the maximum flow in the resulting network is computed starting from the flow obtained at the previous iteration, except during the first iteration which starts with the zero flow. If the capacities are integral and LNA is used in the maximum flow computations, then the algorithm runs in $O(nm)$ time per scaling iteration, for a total of $O(nm \lg U)$ time.

2. Combinatorial flow problems. This chapter addresses combinatorial problems that can be reduced to network flow problems. Below, the layered network algorithm and the blocking flow method are referred to as *auxiliary network algorithms* (ANA). Hopcroft and Karp [4] discovered LNA independently of [10] for the special case of the problem of finding systems of distinct representatives. The material of this section is based mainly on the work of Karzakov [14], [15].

2.1. Flow formulation of combinatorial problems.

Problem of distinct representatives. The classical *problem of distinct representatives* [3] is as follows: given a collection of n subsets V_i of a set V of size $r = n$, find a collection of representatives $v_i \in V_i$ so that representatives of different subsets are distinct. In the *problem of restricted representatives*, n and r may be different, and the goal is to find a_i representatives for each V_i, so that each $v_j \in V$ represents b_j subsets.

The book describes a reduction of the generalized problem to a special kind of network flow problem. On this network, FFA is equivalent to the classical alternating path method. Results of § 2.3 imply that ANA solves the problem of distinct representatives in $O(n^{5/2})$ time and the generalized problem in $O(n^{5/3}r)$ time.

Another problem reducible to a special case of the maximum flow problem is that of *common restricted representatives* [2]. In this problem, the input contains two collections of subsets of V, $\{V_i^1\}$ and $\{V_i^2\}$, for $1 \leq i \leq n \leq r$, and the goal is to select representatives from each subset in such a way that for each collection, the representatives are distinct, and the same set of representatives is used for both collections. Results of § 2.3 imply $O(n^{3/2}r)$ time bound for the problem.

Minimum cut problems. The notion of an edge-cut and a node-cut of a graph are defined, as well as the edge and node connectivity problems. It is shown that these problems can be reduced to the maximum flow problems on special kinds of networks. Results of § 2.3 imply that ANA finds a minimum edge-cut between two subsets of nodes in $O(\min(m^{3/2}, n^{2/3}m))$ time and a minimum node-cut between two subsets of nodes in $O(\sqrt{n}m)$ time. The algorithm of § 2.5 solves the edge-connectivity problem in $O(nm)$ time; the node connectivity problem can be solved in $(k + 1)(n - 1)$ maximum flow computations [20] (where k is the node connectivity of the network).

2.2. Combinatorial networks. The concepts introduced here are used later to analyze ANA on special networks that model combinatorial problems. Such *combinatorial* networks have unit capacities on internal arcs or nodes (where node capacity is defined as a minimum of the total incoming and the total outgoing arc capacity). An *arc-combinatorial* network is a network with unit capacities on internal arcs. The *characteristic* χ of a network is the sum of arc and node capacities over all internal arcs and nodes. A *generalized combinatorial network* is a network with characteristic of $O(m)$.

2.3. Analysis for combinatorial networks. This section analyses the performance of ANA on combinatorial networks as a function of n, m, and χ. The material of the section is based on [4], [11], [14], [15].

1. A phase of ANA takes $O(\chi + m)$ time on networks with integer capacities and $O(m)$ time on generalized combinatorial networks.
2. The number of ANA phases is less than $2\sqrt{\chi} + 2$.
3. The ANA runs in $O(\chi + m)\sqrt{\chi}$ time on networks with integer capacities and in $O(m\sqrt{\chi})$ time on generalized combinatorial networks.
4. The number of ANA phases on an arc-combinatorial network is less than $2\pi^{2/3} + 2$.
5. Suppose G is an arc-combinatorial layered network of length l. Then the number of phases of ANA is less than $\sqrt{8lM(G)} + 1$.
6. Suppose all arcs in a layered network of length l have unit capacities. Then the number of iterations of ANA is $O(\min(\sqrt{nl}, n^{2/3}, n^2/l^2))$.

2.4. A difficult example. This section describes a (parameterized) instance of the problem of finding a system of distinct representatives. The ANA takes $O(n^{5/2})$ time on this instance. For the proof, the reader is referred to [14].

2.5. Efficient algorithm for finding edge connectivity of a graph. This section is devoted to the problem of finding edge connectivity of a graph with no multiple edges. It is shown that $n - 1$ maximum flow computations are sufficient to solve the problem. An algorithm of Podderjugin [20] does this in $O(nm)$ time. The algorithm solves the maximum flow problem using a slight variation of FFA, which differs from the standard FFA in the way augmenting paths of length one and two are handled. The cost of processing such "short" augmenting paths is $O(n)$ per one maximum flow computation and $O(n^2)$ overall. The number of other augmentations done by the algorithm (along "long" paths) is n overall, and the cost of these augmentations is $O(nm)$.

3. Multiterminal and multicommodity problems.

3.1. Multiterminal problems and solution methods. The authors define the notion of multiterminal single-commodity flow. This notion leads to several flow problems, in particular the max-Σ problem (where the objective is to maximize the sum of flows out of the sources), the feasibility problem (for capacitated sources and sinks), and the problem of finding maximum flows in networks with lower bounds on arc flow in addition to upper bounds. These problems reduce to the maximum flow problem [2].

The following theorem is from [7], [12]: For any pair of maximum flows f_1 and f_2, there is a maximum flow f which takes the same amount of flow as f_1 from every source and puts the same amount of flow as f_2 into every sink. (This statement is not true for nonmaximum flows of equal value.)

3.2. Problem with priorities of terminals. A common layered network. We consider the following version of the multiterminal single-commodity flow problem that was studied independently in [7], [12]. Given an ordering $s_1, \cdots, s_k$ of the network sources, define a *source-maximal flow* to be a flow f for which the tuple $(e_f(s_1), \cdots e_f(s_k))$ is lexicographically maximal. The *sink-maximal* flow is defined in a similar way. A flow that is both source- and sink-maximal is called *bimaximal*. Source- and sink-maximal flows are maximum flows, and a bimaximal flow always exists.

A bimaximal flow can be constructed from a source-maximal and a sink-maximal flows. The rest of the section is devoted to algorithms for finding a sink-maximal flow. (The problem of finding a source-maximal flow is symmetric.) Several algorithms to find such a flow are described.

The most efficient algorithm [11] is based on LNA and runs in $O(n^2m)$ time. The algorithm constructs a "common" layered network that contains all nodes of the input graph and all residual arcs that are on shortest paths from s. Using the layered network, the algorithm repeatedly finds shortest paths from s to t_1 until no s–t_1 augmenting path exists; the same procedure is repeated for t_2, t_3, etc. It is shown that distances to nodes from the source in the residual graph are nondecreasing, and the work involved in augmentations is $O(n^2m)$. The section concludes by describing a way to construct and maintain the layered network at $O(nm)$ cost.

3.3. Directed multicommodity problems. This section defines the directed *k-commodity flow problem* and the related *demand graph* induced by arcs connecting commodity sources to the corresponding sinks. It also discusses special cases of the problem with known efficient solutions: the case of a unique source (or unique sink), and the case in which the demand graph is a complete bipartite graph. In these cases the problem reduces to a maximum flow problem. The section concludes with a discussion of a decomposition algorithm for the directed multicommodity problem.

3.4. Undirected two commodity problem. The authors introduce the undirected multicommodity flow problem, and comment that the solution methods described in § 3.3 for the directed problem carry through in the undirected case. The main result of the section is an algorithm that solves the undirected two-commodity problem. Algorithms for this problem were first studied by Hu [5], [6]. Cherkassky [21] gave a polynomial-time variant of Hu's algorithm. Then the problem was reduced to a constant number of maximum flow computations, by Dinits based on [21] and by Sakarovitch [8] based on linear programming techniques. The section describes an algorithm that reduces the problem to four maximum flow computations. If the input is integral, then each value of the solution produced by the algorithm is either an integer or an integer plus a half.

3.5. Solved cases of the undirected feasibility problem. Consider the case of the feasibility problem for undirected graphs, such that every commodity has one of two distinguished nodes as a terminal. In this case the problem can be solved using the two-commodity flow algorithm of § 3.4 and the flow decomposition algorithm of § 3.3. A generalization of this is the fact that an l-commodity feasibility problem, where the edges of its demand graph can be covered by k star graphs, can be reduced to k-commodity max-Σ problem, where the objective is to maximize the total flow of commodities; the reduction adds k edges and l nodes to the original network.

The following results are due to Papernov [19]. Note that given a cut, the capacity of the cut must be at least as big as the sum of demands for pairs of terminals separated by the cut in order for the problem to be feasible. We call this the *cut condition*. For which graphs is the converse true? That is, for which graphs is it true that when the cut condition is satisfied for every cut, then there is a feasible multicommodity flow? It is true precisely when the graph is either (1) a union of two star graphs, (2) a cycle of length five, or (3) a complete graph on four nodes.

3.6. Solved cases of the undirected max-Σ problem. This section discusses special cases of the undirected max-Σ multicommodity problem with known efficient solutions.

The first result of the section is a reduction from a class of the problems for which demand graphs are a union of two complete bipartite graphs to the two-commodity multiterminal problem. This reduction, which uses the decomposition algorithm (see § 3.3), is due to Cherkassky [23].

The following is a conjecture of Cherkassky, which is an analog of the max-flow, min-cut theorem for the case of multiterminal undirected two-commodity flow problem. Let S and T be the sets of sources and sinks of commodity 1, respectively, and let P and Q be the sets of sources and sinks of commodity 2. Let $c(N)$ denote the capacity of a minimum cut separating the terminals in N from the remaining terminals. Then the value of the maximum two-commodity flow is equal to

$$\tfrac{1}{2} \min\left(c(X)+c(Y)+c(Z)+c(W)\right),$$

where the minimum is taken over all partitions of terminals such that $X \subset S \cup P$, $Y \subset T \cup P$, $Z \subset S \cup Q$, and $W \subset T \cup Q$.

Now consider an instance of the problem for which the demand graph is a complete graph; in other words, there is a commodity for every pair of terminals. Let c_i be the capacity of the minimum cut separating the ith terminal from the remaining ones. Then the maximum flow value is equal to $\frac{1}{2} \sum_i c_i$. This theorem was stated in [18]. Cherkassky [22] found an error in the proof of [18], gave a correct proof, and presented a polynomial-time algorithm for this class of problems. The section concludes with an outline of the algorithm.

3.7. Completeness of a multicommodity flow problem for a class of linear programming problems. The multicommodity flow problem is shown to be complete for a class of linear programming problems; the problems in this class appear to be as hard as the general linear programming problem. They have two special properties: the variables in the problems are bounded and the matrix has integral coefficients. This result is due to Dinits.

4. Minimum-cost flow problem. This chapter is devoted to the minimum-cost flow problem and the transportation problem reducible to it. Although no strongly polynomial-time algorithm is known for these problems, polynomial-time algorithms (whose running time depends on logarithms of input numbers) are known. These are scaling algorithms due to Edmonds and Karp [1] and Dinits [11].

The main goal of this chapter is to describe instances of the minimum-cost flow problem resulting in exponential number of iterations of the currently known methods. These results are from [9], [13].

4.1. Minimum-cost augmenting path method for finding a maximum flow of minimum cost. The *minimum-cost augmenting path method* (MAPM) starts with a zero flow and iteratively augments it along a cheapest augmenting path. The method works under the assumption that the input network does not have negative cycles. It is shown that the computation of MAPM can be partitioned into phases; each phase finds a maximum flow in the network of minimum-cost augmenting paths using FFA. Since FFA need not terminate [2], MAPM need not terminate either. A variation of MAMP which uses a polynomial-time maximum flow algorithm to find maximum flows in networks of minimum-cost paths always terminates. This variation is called the *generalized method* (GM). For integral capacities, MAPM terminates in time bounded by the sum of all arc capacities.

4.2. Exponential example for the minimum-cost augmentation method. This section describes a parameterized instance such that the number of phases of GM on an instance with $2k$ nodes is 2^{k-1}.

4.3. The imbalance canceling method for the transportation problem. This section introduces the transportation problem and describes a class of *imbalance canceling* algorithms for it. The *imbalance* at v is the difference between the supply and the current deficit at v if v is a source, and between the demand and the current excess if v is a sink. An imbalance canceling algorithm constructs a sequence of flows such that imbalances monotonically decrease and each flow is optimal for the natural choice of supplies and demands.

A common algorithm in this class is the *Hungarian method* (HM). The method maintains a feasible price function and either augments flow on a residual path of zero reduced cost arcs from a supply node to a demand node, or updates the price function if no such path exists. As in the algorithm of § 4.1, a finite maximum flow algorithm can be used to guarantee termination. There are two variations of HM. The *regular* HM starts with a zero flow and price functions. The HM *with a preprocessing stage* first goes through the sources s, finding for each one the arc (s, w) of minimum cost, sets the price of s to the cost of (s, w), and sends as much flow as possible along (s, w).

4.4. The network simplex method for the transportation problem. This section describes the primal network simplex algorithm for the transportation problem. The particular version described chooses the minimum reduced cost arc to enter the basis and applies the perturbation method to avoid cycling.

4.5. A relationship between the network simplex method and the regular Hungarian method. This section shows that the network simplex method is equivalent to HM on a slightly modified network.

4.6. Exponential examples for the network simplex method and the Hungarian method. We are given an example of a $k \times k$ transportation problem on which HM takes 2^{k-1} iterations. A modification of this example gives an exponential lower bound on HM with preprocessing. Results of § 4.5 imply the 2^{k-1} lower bound on the number of iterations of the network simplex method.

Acknowledgments. We would like to thank Dick Karp, David Shmoys, Éva Tardos, and Bob Tarjan for comments on a draft of this review.

Research partially supported by National Science Foundation Presidential Young Investigator Grant CCR-8858097 with matching funds from AT&T, IBM Faculty Development Award, and a grant from 3M Corporation.

Research partially supported by grant CCR-8803704 from the National Science Foundation.

REFERENCES

[1] J. EDMONDS AND R. M. KARP, *Theoretical improvements in algorithmic efficiency for network flow problems*, J. Assoc. Comput. Mach., 19 (1972), pp. 248–264.

[2] L. R. FORD, JR., AND D. R. FULKERSON, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[3] P. HALL, *On representatives in subsets*, J. London Math. Soc., 10 (1935), pp. 26–30.

[4] J. E. HOPCROFT AND R. M. KARP, *An $n^{5/2}$ algorithm for maximum matching in bipartite graphs*, SIAM J. Comput., 2 (1973), pp. 225–231.

[5] T. C. HU, *Multi-commodity network flows*, J. ORSA, 11 (1963), pp. 344–360.

[6] ———, *Integer Programming and Network Flows*, Addison-Wesley, Reading, MA, 1969.

[7] E. MINIEKA, *Maximal, dynamic and lexicographic flows*, Oper. Res., 21, 1973.

[8] M. SAKAROVITCH, *Two-commodity network flow and linear programming*, Math. Programming, 5 (1973), pp. 1–20.

[9] N. ZADEH, *A bad network problem for the simplex method and other minimum cost flow algorithms*, Math. Programming, 5 (1973), pp. 255–266.

[10] E. A. Диниц. *Алгоритм решения задачи о максимальном потоке в сети со*

степенной оценкой, Доклады АН СССР, 194 (1970), pp. 754–757, English translation: E. A. Dinits, *Algorithm for solution of a problem of maximum flow in networks with power estimation*, Soviet Math. Dokl., 11 (1970), pp. 1277–1280.

[11] ———, *Метод Поразрядного Сокращения Невязок и Транспортные Задачи*. Сб. Исследования по Дискретной Математике, Наука, Москва, 1973. Transcription: E. A. Dinits, *Metod Porazryadnogo Sokrashcheniya Nevyazok i Transportnye Zadachi*, Issled. Diskret. Mat., Science, Moscow, 1973. Title translation: *The Method of Scaling and Transportation Problems*.

[12] ———, *Экономные Алгоритмы Решения Задач Транспортного Типа*. Канд. дисс., М.Г.У., 1973. Transcription: E. A. Dinits, *Ekonomnye Algoritmy Resheniya Zadach Transportnogo Tipa*, Ph.D. dissertation, Moscow State University, 1973. Title translation: *Efficient Algorithms for Solving Transportation Problems*.

[13] Е. А. Диниц и А. В. Карзанов. *Об Экспоненциальной Сложности Алгоритмов Решения Общей и Транспортной Задач Линейного Програмирования*. Сб. Труды XIX Конференции Молодых Ученых ИАТ. Москва [ротапринт], 1974. Transcription: E. A. Dinits AND A. V. Karzanov, *Ob Exponentsial'noi Slozhnosti Algoritmov Resheniya Obshchei i Transportnoi Zadach Lineinogo Programirovaniya*, Trudy XIX Konferentsii Molodykh Uchenykh IAT, Moscow, 1974. Title translation: *On Exponential Complexity of Algorithms for the General and Transportation Linear Programming Problems*.

[14] А. В. Карзанов. *Оценка Алгоритма Нахождения Максимального Потока, Примененного к Задаче о Представителях*. Сб. Вопросы Кибернетики. Труды Семинара по Комбинаторной Математике. (Москва 1971), Советское Радио, Москва, 1973. Transcription: A. V. Karzanov, *Otsenka Algoritma Nokhozhdeniya Maksimal'nogo Potoka, Primenennogo k Zadache o Predstavitelyakh*, Voprosy Kibernet., Trudy Seminara po Kombinatornoi Matematike. (Moskva 1971), Soviet Radio, Moscow 1973. Title translation: *A Bound on a Maximum Flow Algorithm Applied to the Problem of Distinct Representatives*.

[15] ———, *О Нахождении Максимального Потока в Сетях Специального Вида и Некоторых Приложениях*. Сб. Математические Вопросы Управления Производством, том 5. Издательство М.Г.У., Москва, 1973. Transcription: A. V. Karzanov, *O Nakhozhdenii Maksimal'nogo Potoka v Setyakh Spetsial'nogo Vida i Nekotorykh Prilozheniyakh*, Mat. Voprosy Upravleniya Proizvodstvom, Vol. V, Moscow State University Press, Moscow. Title translation: *On Finding Maximum Flows in Network with Special Structure and Some Applications*.

[16] ———, *Нахождение Максимального Потока в Сети Методом Предпотоков*. Доклады АН СССР, 215 (1974), pp. 49–53. English translation: A. V. Karzanov, *Determining maximal flow in a network by the method of preflows*, Soviet Math. Dokl., 15 (1974), pp. 434–437.

[17] ———, *Экономный Алгоритм Нахождения Максимального Потока в Сетях*, Экономика и Математические Методы, 11, 1975. Transcription: A. V. Karzanov, *Ekonomnyi algoritm nakhozhdeniya maksimal'nogo potoka v setyakh*, Ekonom. i Mat. Metody. Title translation: *Efficient algorithm for finding a maximum flow in a network*.

[18] В. Л. Куперштох. *Об Одном Обобщении Теоремы Форда и Фалкерсона на Многополюсные сети*, Кибернетика (Киев), 3, 1971. English translation: V. L. Kupershtokh, *The generalization of Ford-Fulkerson theorem to multipole networks*, Cybernetics, 3 (1973), pp. 494–502.

[19] Б. А. Папернов. *Реализуемость Многопродуктовых Потоков*. Сб. Исследовацния по Дискретной Оптимизации. Наука, Москва, 1976. Transcription: B. A. Papernov, *Realizuemost' mnogoproduktovykh potokov*, Issled. Diskretnoi Optimizatsii, Science, Moscow. Title translation: *Feasibility of Multicommodity Flows*.

[20] Б. Д. Поддерюгин. *Алгоритм Определения Реберной Связности Графа*. Сб. Вопросы Кибернетики. Труды Семинара по Комбинаторной Математике (Москеа 1971), Советское Радио, Москва, 1973. Transcription: B. D. Podderjugin, *Algoritm Opredeleniya Rebernoi Svyaznosti Grafa*, Voprosy Kibernet., Trudy Seminara po Kombinatornoi Matematike, Soviet Radio, Moscow. Title translation: An Algorithm for Determining Edge Connectivity of a Graph.

[21] Б. В. Черкасский, *Конечный Алгоритм Решения Задачи о Двухпродуктовом Потоке*, Экономика и Математические Методы, 9 (1973), pp. 1147–1149. Transcription: B. V. Cherkassky, *Konechny algoritm resheniya zadachi o dvukhproduktovom potoke*, Ekonom. i Mat. Metody. Title translation: *A finite algorithm for solving the two commodity flow problem*.

[22] ———, *Решение Одной Задачи о Многопродуктовых Потоках в Сети*, Экономика и Математические Методы, 11, 1975. Transcription: *Reshenie Odnoi Zadachi o Mnogoproduktovykh Potokakh v Seti*,

Ekonom. i Mat. Metody. Title translation: *A Solution of a Problem of Multicommodity Flows in a Network.*

[23] ———, *Многополюсные Двухпродуктовые Задачи*, Сб. Исследоеания по Дискретной Оптимизации, Наука, Москва, 1976. Transcription: *Mnogopolyusnye Dvukhproduktovye Zadachi*, Issled. Diskretnoi Optimizatzii, Science, Moscow, 1976. Title translation: *Multiterminal Two Commodity Problems.*

ANDREW V. GOLDBERG
Stanford University

DAN GUSFIELD
University of California, Davis

Singularity Theory, Rod Theory and Symmetry-Breaking Loads. *By John F. Pierce.* Springer-Verlag, Berlin, 1989. iv + 178 pp. $16.20, paper. ISBN 0-387-51304-3 (US only). Lecture Notes in Mathematics, Vol. 1377.

In 1859 Kirchhoff [1] extended Euler's earlier work [2] by developing a theory for the bending and twisting of an initially straight, prismatic, linearly elastic rod in three-dimensional space (spatial elastica). In that same work he noted that if the rod is subjected to terminal loads and couples only, then the static equilibrium equations are identical to the equations of motion of a rigid body with a fixed point; this is Kirchhoff's dynamical analogy [3]. For example, a particular solution corresponding to the regular precession of a symmetrical top implies that a rod with a cross section having equal moments of inertia admits helical solutions solely under the action of applied terminal thrust. In spite of its classical beginning, the (finite-strain) theory of rods remained more or less dormant until 1958 when Ericksen and Truesdell [4] resurrected and clarified the forgotten and somewhat obscure work of the Cosserats [5]. This, in turn, led to the development of "Cosserat" rod theories (as well as shell and continuum theories) in the spirit of modern continuum mechanics in the 1960s, e.g., [6], [7].

While the formulation of such problems is now well understood (if not widely known), the analysis is not. Indeed, rod theory remains a rich source of interesting nonlinear problems, as exemplified by the monograph under review. The author takes up the problem of a transversely isotropic, hyperelastic Kirchhoff rod subject to terminal compression. (By *hyperelastic* we mean that the material behavior of the rod is characterized by a general class of stored-energy functions of the two curvatures and twist of the rod (strains); the gradient of the stored-energy function is equal to the contact couple). At first glance this appears to be a modern reformulation of the classical problem of Kirchhoff, which has been analyzed in detail, cf. [8], [9]. However, the primary goal of the work under review is to study the effects of small axisymmetry-breaking body forces (perturbations) on the local solution set ("close" to the straight state) in two cases: (i) the "pure orbit-breaking problem"—the compressive end load is small compared to the first buckling load of the unperturbed problem; (ii) the "full problem"—for compressive loads near the lowest buckling load. These problems are more difficult than the more standard one of local spontaneous symmetry-breaking bifurcation (for example, cf. [10]), by virtue of the fact that the symmetry is purposefully broken in the model (not just by certain solutions).

As suggested by the title, the author combines tools and ideas from mechanics and modern mathematics to formulate and tackle these problems. For example, one of the first difficulties arising in this general class of two-point boundary value problems is the nonlinearity of the configuration space; the rotation field (of the cross sections), $[0, 1] \ni s \mapsto R(s) \in SO(3)$, is required. Hence, we naturally encounter differential equations on manifolds. The author confronts this problem head on by presenting a self-contained, coordinate-free formulation for both Kirchhoff and special Cosserat rods (the latter being capable of large shearing and stretching motions, as well as the usual bending and twisting motions) in the modern language of analysis on manifolds. Workers in continuum mechanics will be especially interested in this section. For example, by viewing moments as skew second-order tensors and by working directly with the rotation as a basic unknown field, Pierce obtains a coordinate-free characterization of cross-sectional material symmetry in a formulation free of cross products and directors (two orthonormal vectors spanning a plane parallel to the deformed cross section).

Most of the analysis is devoted to the pure orbit-breaking problem. This is akin to the Signorini–Stoppelli problem of three-dimen-