DA
1322 (16)
1994

Algorithms for Network Flow Problems with Matroidal and Submodular Constraints

by

Xiaodong ZHANG

January 1995

A DISSERTATION

submitted to the University of Tsukuba
in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY
in
Management Science and Engineering

DOCTORAL PROGRAM IN SOCIO-ECONOMIC PLANNING
THE UNIVERSITY OF TSUKUBA

Abstract

In this thesis, we present new algorithms for network flow problems with boundary constraints described by matroids or submodular functions.

In Chapter 1 we introduce some preliminaries from theories of network flow problems, matroids, submodular systems and greedy algorithms for matroidal and submodular optimizations.

In Chapter 2 we consider the problem of finding a maximum common subbase of two submodular systems on E with $|E| = n$, which is called the submodular intersection problem. The submodular intersection problem is a generalization of the so-called matroid intersection problem and has a lot of practical applications. The existing algorithms for the submodular intersection problem can be found in [72] of P. Schönsleben and [77] of É. Tardos, C. A. Tovey and M. A. Trick. Both algorithms consist of augmentations along successive augmenting paths in an auxiliary network. The number of augmentations are $\mathrm{O}(n^3)$. The key to get this bound is that the successive augmenting paths are chosen in a lexicographic order; this idea is due to P. Schönsleben [72]. On the other hand, in [43] A. V. Goldberg and R. E. Tarjan proposed an excellent algorithm for ordinary maximum flow problems, which is called the preflow-push algorithm. The algorithm consists of only local operations of relabelings for distance labels and pushes for flows on arcs. Goldberg and Tarjan's idea is extended, in Chapter 2, to obtain an efficient algorithm for submodular intersection problem. Let $(\mathcal{D}_i, f_i)$ $(i = 1, 2)$ be two submodular systems on E. The submodular intersection problem is to find a maximum common subbase $x \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2)$, where for each $i = 1, 2$ $\mathrm{P}(f_i)$ is the submodular polyhedron associated with $(\mathcal{D}_i, f_i)$. First, we present a new algorithm by finding shortest augmenting paths in the auxiliary graph, which begins with a pair (y, z) of subbases of the given submodular systems and is convenient for adopting the preflow-push approach of A. V. Goldberg and R. E. Tarjan [43]. We require that $y \in \mathrm{B}(f_1)$,

$z \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2)$ and $y \geq z$. Using the technique of selecting the lexicographically shortest path, the initial common subbase z converges to a maximum common subbase by at most $\mathrm{O}(n^3)$ augmentations. Secondly, by using the basic ideas of the preflow-push method and starting from the same initial subbase pair (y, z), we devise a faster algorithm for the intersection problem consisting of only local operations of relabelings and pushes, which requires in total $\mathrm{O}(n^3)$ push and $\mathrm{O}(n^2)$ relabeling operations by the largest-label implementation with a specific order on the out-going arc list for each vertex in the auxiliary graph. Since the proposed algorithm uses only local operations, the present algorithm is very different from the previous ones. In Chapter 2, we also propose a first-active implementation of our preflow-push approach. The resultant algorithm finds a maximum common subbase of two given submodular systems by $\mathrm{O}(n^4)$ saturating push, $\mathrm{O}(n^3)$ nonsaturating push and $\mathrm{O}(n^2)$ relabeling operations. It is the first algorithm that solves the submodular intersection problem (or equivalently the maximum submodular flow problem) in strongly polynomial time without using a lexicographical ordering. For the submodular intersection problem, the algorithm devised by P. Schönslben [72], finds a maximum common subbase by $\mathrm{O}(n^3)$ augmentations along the lexicographically shortest augmenting paths. Each augmenting path can be found in $\mathrm{O}(n^2)$ time by Dijkstra's shortest path algorithm with oracles for identifying arcs in auxiliary networks. A complexity improvement over P. Schönsleben's algorithm is made by É. Tardos, C. A. Tovey and M. A. Trick [77] by reducing the total time in finding augmenting paths, using the idea of layered network due to E. A. Dinits [12] for ordinary maximum flows. Their algorithm runs in $\mathrm{O}(n^4 h)$ time, where h is the time to identify an arc in an auxiliary network and an arc capacity. We show that our preflow-push algorithm has a lower time complexity with oracles for identifying arcs and their capacities.

In Chapter 3 an efficient cost scaling algorithm is presented for the independent assignment problem of M. Iri and N. Tomizawa [53], which is equivalent to the weighted matroid intersection problem of J. Edmonds [13]. The independent assignment problem or the weighted matroid intersection problem is used in determining the order of complexity of an electrical network [51], the unique solvability of an electrical network [51], the structural solvability and controllability problems of large systems [64], controllability and observability of a linear dynamical system with combinatorial constraints

[51], and the minimum-weight spanning arborescence problem [59].

Consider a bipartite graph $G = (V^+, V^-; A)$ with the left (right) end-vertex set V^+ (V^-) and the arc set A directed from V^+ to V^-. Two matroids $\mathcal{M}^+ = (V^+, \mathcal{I}^+)$ and $\mathcal{M}^- = (V^-, \mathcal{I}^-)$ are defined, respectively, on V^+ and V^- with families $\mathcal{I}^+ \subseteq 2^{V^+}$ and $\mathcal{I}^- \subseteq 2^{V^-}$ of independent sets. A cost function $c : A \to \mathbf{Z}$ is given, where $\mathbf{Z}$ is the set of all integers. A subset M of A is called an independent matching if any two arcs in M have no common end-vertices (i.e., M is a matching) and the left (right) end-vertex set of M is an independent set of $\mathcal{M}^+$ $(\mathcal{M}^-)$. For a positive integer k, a k-independent matching M is an independent matching of cardinality k. An optimal k-independent assignment in $\mathcal{N}$ is a k-independent matching M having the minimum cost $c(M) = \sum_{e \in M} c(e)$ among all the k-independent matchings.

We propose an algorithm for solving the optimal k-independent assignment problem, which can be viewed as a generalization of J. B. Orlin and R. K. Ahuja's scaling algorithm [66] for the ordinary assignment problem. The concept of ε-optimality is used in our algorithm which consists of scaling phases, each finds an ε-optimal k-independent assignment. Each scaling phase starts with an ε-optimal 0-independent assignment, ends with an ε-optimal k-independent assignment. It can be decomposed into two parts: an auction-like algorithm (see [4], [66]) and a successive shortest path algorithm. In the auction-like algorithm we perform relabelings on a dual variable (potential), flow pushes on an arc and elementary transformations for an independent set. The successive shortest path algorithm uses shortest path augmentations and is similar to the algorithm proposed by M. Iri and N. Tomizawa [53]. On a bipartite graph with n vertices and integer arc costs bounded by C, an optimal k-independent assignment can be found in $O(\sqrt{k}n^2 \log(kC))$ time by our algorithm under an independence oracle for matroids.

There are two algorithms with a computational complexity similar to ours, one proposed by H. N. Gabow and Y. Xu [40] and the other recently by M. Shigeno and S. Iwata [74]. Both are given for the weighted matroid intersection problem. Comparing our algorithm with that of H. N. Gabow and Y. Xu [40], we can see that our algorithm is simple and has advantage in the data structure and memory space. The algorithm of M. Shigeno and S. Iwata has a computation structure which is similar to that of our algorithm.

In Chapter 4 we consider the problem of finding minimum-cost submodular flows. A submodular flow is a feasible flow in a capacitated network with the constraint that the flow boundary belongs to the base polyhedron described by a given submodular system. The problem is to find a submodular flow of minimum cost. The minimum-cost submodular flow problem was first considered by J. Edmonds and R. Giles [16]. This problem is a generalization of the submodular intersection problem, the independent assignment problem, the ordinary minimum-cost flow problem and others. We give an algorithm for the problem, which is a generalization of the algorithm for ordinary minimum-cost flows devised by A.V. Goldberg and R. E. Tarjan [44] and is also a direct generalization of our algorithm for the submodular intersection problem. The algorithm uses the technique of cost scaling and the concept of ε-optimality. The procedure for each scaling phase converts a 2ε-optimal submodular flow to an ε-optimal submodular pseudo-flow and then converts it to an ε-optimal submodular flow. This process is repeated for successively smaller values of ε. In the procedure, we perform local operations such as relabelings for a potential variable on a vertex, flow pushes on an arc and elementary transformations of a base. For a directed graph with n vertices, m arcs, integer arc costs bounded by Γ and arc capacities bounded by U, a minimum-cost submodular flow can be found in $\mathrm{O}(\log(n\Gamma))$ cost scalings and each scaling phase performs at most $\mathrm{O}(mnU)$ relabeling and $\mathrm{O}(mn^3U)$ push operations. The complexity of the algorithm is pseudo-polynomial in general and the complexity becomes polynomial in case of minimum-cost 0-1 submodular flows. Since our algorithm is based on the so-called "local operations" consisting of relabeling and push operations, the structure of the algorithm is very simple and easy to implement. To get a polynomial time complexity for our algorithm is related to capacity scaling technique and other ideas. To devise a polynomial time algorithm based on a capacity scaling for submodular flows is still an unsolved problem.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Professor Satoru Fujishige of the Institute of Socio-Economic Planning, University of Tsukuba, for introducing me to the theories of network flows and submodular systems and for his never-failing guidance. It is my very fortunate to have had Professor Satoru Fujishige as my supervisor.

This dissertation summarizes part of the research carried out over a period of five years at Doctoral Program in Socio-Economic Planning at University of Tsukuba. The thesis is based on three papers ([37, 38, 84]). The first two papers are co-authored by Professor Satoru Fujishige.

I wish to thank Professor Yoshitsugu Yamamoto and Associated Professor Kazuo Kishimoto of the University of Tsukuba for their valuable advice and kindly help.

I am very grateful to Associate Professor Akira Nakayama of Fukushima University, Mr. Takeshi Naitoh of Shiga University, Mr. Sekitani of Science University of Tokyo, Mrs. Ping Zhan of the University of Tsukuba, Mr. Wakase Kise of Mitsubishi Electric Corporation, my colleagues Mr. Toshio Nemoto and Mr. Kazutoshi Ando for their friendly help and encouragement, and to Mr. Satoru Iwata of Kyoto University and Miss Maiko Shigeno of Tokyo Institute of Technology for their useful comments on Chapter 3.

This research is financially supported by the Ministry of Education, Science and Culture of Japan to which I am very grateful.

Finally, I would like to express my gratitude to my wife and child for their love and patient.

Contents

Chapter 1.

Introduction

1.1. Introduction

Historical Views

The classical network flow theory was founded by L. R. Ford and D. R. Fulkerson ([19],[20]) and others. The most typical network flow problems are the maximum flow problem and the minimum-cost flow problem.

For a maximum flow problem, a flow which satisfies capacity constraints on arcs and flow conservation constraints on vertices except from source and sink vertices is called a feasible flow. Changing a feasible flow only on one arc will violate the flow conservation constrains. The operations on a flow which maintain the feasibility of the flow are such as flow augmentations on an augmenting path from the source vertex to the sink vertex and flow changes along a directed cycles. Most of the algorithms for the maximum flow problem are based on the augmenting path approach which was initialized by L. R. Ford and P. R. Fulkerson [20]. Among those famous algorithms of such type, there are J. Edmonds and R. M. Karp's algorithm [17] and E. A. Dinits's algorithm [12].

Another type of algorithms for maximum flow problems is originated by A. V. Karzanov [55] who first introduced the concept of preflows and the local operations called push and balance instead of the global operations by the augmenting path approach. A preflow is a flow satisfying capacity constraints on arcs and non-negativity constraints on vertices except from the source vertex, that is, the total flow into a vertex v is at least as great as the total flow out of v. Changing a preflow only on one arc may keep the constraints for the preflow. Such a change is called a push. A. V. Goldberg and R. E. Tarjan [43] developed an excellent algorithm of such type. They used the distance label instead of the level graph used by E. A. Dinits and A. V. Karzanov. Distance labels are the dual variables of a linear optimization problem, i.e., here the maximum flow problem. The algorithm terminates when a preflow becomes a feasible flow and the feasibility of the distance label ensures the optimality of the flow. A. V. Goldberg and R. E. Tarjan's preflow-push algorithm is viewed as the most important development in the algorithmic theory for the maximum flow and minimum-cost flow problems in 1980's. Also many refinements on their algorithm have been suggested, such as R. K. Ahuja and J. B. Orlin's algorithm [1] by a excess scaling and Cheriyan and Hagerup's algorithm [8] which introduced a randomized method for R. K. Ahuja and J. B. Orlin's excess scaling algorithm. In Chapter 2, we generalize A. V. Goldberg and R. E. Tarjan's preflow-push algorithm to the submodular intersection problem. The complexity of the resultant algorithm is among the best.

It is reported that A. V. Goldberg and R. E. Tarjan's algorithm is the most practical algorithm for maximum flow problems (see [54]) and the complexity of their algorithm is among those of best. The algorithms by the preflow-push approach are easily applied to construct parallel algorithms for maximum flow problems (A. V. Goldberg [42]).

The minimum-cost flow problem is much complicated than the maximum flow problem. There are many different algorithms for the minimum-cost flow problem. M. Klein [56] proposed an algorithm by the negative cycle canceling, i.e., a feasible flow is modified into an optimal flow by successively canceling negative cycles. R. Hassin proposed an algorithm [49] by maintaining the dual feasibility of a flow and then modifying it into a primal feasible one through the positive cut canceling method. Another important type of algorithms for minimum-cost flow problems is called the successive shortest path method. The algorithm starts from an optimal 0-flow and augments the flow on the shortest path in the residual network while maintaining the non-negativity of reduced costs in arcs of the residual network. There are other algorithms such as the primal-dual method, the primal simplex method, the dual simplex method and the out-of-kilter method (see [2]).

Cost scaling and capacity scaling are fundamental techniques for recently developed low complexity algorithms for the minimum-cost flow problem. Based on the scaling method É. Tardos [76] proposed the first strongly polynomial algorithm for minimum-cost flow problems. In [31] S. Fujishige improved the complexity of Tardos' algorithm by introducing the dual framework of Tardos' algorithm. A. V. Goldberg and R. E. Tarjan [44] proposed a cost scaling method based on the concept of ε-optimality. Their algorithm is a generalization of the preflow-push algorithm for the maximum flow problem. The algorithm finds ε-optimal flows for successive smaller value of ε. That is, given an ε-optimal flow, a subprocedure called refine procedure transfers it into an $\varepsilon/2$-optimal flow. Similarly to the concept of preflow, they use a pseudo-flow which satisfies capacity constraints only in the algorithm in stead of a feasible circulation. In the refine procedure, two basic local operations of push and relabeling are performed repeatedly, where a relabeling is a change on the potential (a dual variable) of a vertex. In fact, the refine procedure is very similar to the preflow-push algorithm for the maximum flow problem. Implementations of the Goldberg-Tarjan minimum-cost flow algorithm have been investigated in [54]. A generalization of this algorithm to the submodular flow problem is presented in Chapter 4.

An important special case of the minimum-cost flow problem is the optimal assignment problem. In this problem, the graph is assumed to be a bipartite graph and the flow value of each arc takes on 0 or 1. D. P. Bertsekas and J. Eckstein proposed a practical and efficient algorithm called the auction method [5] which assigns jobs to persons using auction, a kind of local operations. Incorporating cost scaling in this method, they obtained an efficient polynomial time algorithm. Their computational results find the auction algorithm to be substantially faster than the best other methods for the assignment problem for several classes of networks. J. B. Orlin and R. K. Ahuja [66] constructed an efficient cost scaling algorithm by combining the auction method and the successive-shortest-path method. The hybrid version substantially improves the running times obtained by using either technique alone. For a bipartite graph with n vertices and m arcs, the auction algorithm starts with a null assignment and assigns all but at most $O(\sqrt{n})$ vertices; these unassigned vertices are subsequently assigned by the successive-shortest-path algorithm. In Chapter 3, we show that this combination method is also applicable to the independent assignment problem. The resultant algorithm has a time bound which is comparable to the best available time bound.

As mentioned above, in this thesis, several algorithms for classical network flow problems were generalized to network flow problems with matroidal and submodular constraints on flow boundaries. The concept of matroid was first introduced in 1935 by H. Whitney [83] and independently by B. L. van der Waerden [78]. Matroid defines a framework of an abstract linear independence structure of a given finite set. A submodular function is a real valued function defined on subsets of a given finite set. The rank function of a matroid is a special case of submodular functions.

The research of H. Whitney and W. T. Tutte for matroids can be viewed as an early stage of the theory of submodular functions. J. Edmonds made a lot of contributions on matroids and polymatroids in 1960's. In cooperative games of a characteristic function form, L. S. Shapley's work [73] is related to the theory of submodular functions. S. Fujishige's extensive research on base polyhedra and submodular systems provided further theoretical developments in this area. Submodular functions frequently appear in the analysis of combinatorial systems such as graphs, networks, and algebraic systems. It often plays an important rôle in revealing the fundamental structure of a combinatorial problem.

Many network flow problems can not be modeled as classical network flow problems. J. Edmonds and R. Giles [16] described a kind of network flows called submodular flows. Besides the capacity constraints of flows, it is required that the boundary of the flow belongs to the base polyhedron defined by a submodular function. In [16] J. Edmonds and R. Giles proved a very general min-max relation concerning submodular functions on directed graphs. Based on the ellipsoid method, M. Grötschel, L. Lovász and A. Schrijver [45] solved the minimum-cost submodular flow problem in polynomial time. The minimum-cost submodular flow problem includes many other network flow problems and combinatorial optimization problems such as the orientation problem [23], dijoin problem [22, 25], minimum-cost flow problem, intersection problem of two submodular systems [13, 28], and directed cut covering problem [62]. In [26], the applications of submodular flows in finding optimal supporting set in bipartite graphs and directed graphs and improving networks so as to have k edge-disjoint paths were discussed. W. H. Cunningham and A. Frank [11] proposed a polynomial time primal-dual algorithm to solve the minimum-cost submodular flow problem with an oracle for minimizing submodular functions. The algorithm uses only combinatorial steps (like building auxiliary networks and finding augmenting paths) instead of the ellipsoid method which is only of theoretical significance. The first strongly polynomial algorithm was based on the ellipsoid method (cf. M. Grötschel, L. Lovasz and A. Schrijver [46]). In [35] S. Fujishige, H. Röck and U. Zimmermann generalized the cost scaling method for the minimum-cost flow problem [71], [31] to the submodular flow problem and gave a strongly polynomial time algorithm, provided that an oracle for exchange capacities is available. The cycle canceling method of M. Klein [56] for the minimum-cost flow problem was adapted by U. Zimmermann [85] to the minimum-cost submodular flow problem. W. Cui and S. Fujishige [10] devised a finite variant of the cycle canceling method for the submodular flow problem with minimum-mean cycle selection and U. Zimmermann [86] developed a pseudo-polynomial variant.

In [28] S. Fujishige defined the independent flow problem. In the problem, there are two specific vertex sets called the source vertex set and sink vertex set and a submodular function defined on the source vertex set and the sink vertex set, respectively. It is required that for an independent flow its boundary on the source vertex set (the sink vertex set) should be an element of the base polyhedron of the submodular function on the source vertex set (sink vertex set). It was pointed out in [32, S. Fujishige] that the submodular flow problem and the independent flow problem can be reduced to each other. In [28, S. Fujishige] a primal algorithm by canceling negative cycles and a primal-dual algorithm by shortest path augmentations were proposed to find an optimal independent flow.

For the independent flow problem, when submodular constraints are simplified into matroidal constraints and the original graph is a bipartite graph with the source vertex set (the sink vertex set) as its left vertex (right vertex) set, the resultant problem is called the independent assignment problem which is first formulated and solved by M. Iri and N. Tomizawa [53]. M. Iri and N. Tomizawa's algorithm in [53] for finding an optimal independent assignment is a successive shortest path approach and S. Fujishige's algorithm in [27] for the same problem is a matroidal counterpart of the primal-type algorithm for the ordinary assignment problem due to M. Klein [56]. For the independent assignment problem with graphic matroids, an efficient cost scaling algorithm was given in [41] by H. N. Gabow and Y. Xu. J. Edmonds [13] and E. L. Lawler [58] have also considered an essentially equivalent problem called the weighted matroid intersection problem, i.e., the problem of finding a common independent set in a fixed cardinality of two matroids, having the smallest (largest) total weight. The algorithms for weighted matroid intersection problem can be found in [13], [15] of J. Edmonds, [58] of E. L. Lawler, [21] of A. Frank, [67] of J. B. Orlin and J. Vande Vate, [6] of C. Brezovec, G. Cornuéjols and F. Glover, and [40] of H. N. Gabow and Y. Xu.

As a special case of independent flow problems, we have the intersection problem of two submodular systems which is an optimization problem on the common subbases of two base polyhedra. An algorithm devised by P. Schönsleben [72] finds a maximum common subbase by $O(n^3)$ augmentations on the lexicographically shortest augmenting paths in auxiliary networks and each augmenting path can be found in $O(n^2)$ time by Dijkstra's shortest path algorithm with oracles for identifying arcs in auxiliary networks. A complexity improvement over P. Schönsleben's algorithm is made by É. Tardos, C. A. Tovey and M. A. Trick's algorithm [77] by reducing the total time in finding augmenting paths using the idea of layered network due to E. A. Dinits [12]. Their algorithm runs in $O(n^4h)$ where h is the time to identify an arc in an auxiliary network and the arc's capacity.

The practical applications of the independent assignment problem can be found in electric network problems, systems analysis and others such as the problems of minimum fundamental equations in an electric network, topological conditions for the existence of the unique solution in an electric network, order of complexity of a linear electric network [51] and controllability/observability of a linear dynamical systems with combinatorial constraints [51]. In [64] of K. Murota the applications of the independent assignment problem and the submodular intersection problem to system analysis are extensively studied. In [70] applications in engineering and in statics are surveyed by A. Recski.

Outline of the Thesis

In this thesis, we present new algorithms for network flow problems with boundary constraints described by matroids or submodular functions.

In Chapter 1 we introduce some preliminaries from theories of network flow problems, matroids, submodular systems and greedy algorithms for matroidal and submodular optimizations. We also describe some examples of submodular (supmodular) functions and their relations to practical problems.

In Chapter 2 we consider the problem of finding a maximum common subbase of two submodular systems on E with $|E| = n$ called the intersection problem. The submodular intersection problem is a generalization of the so-called matroid intersection problem and has a lot of practical applications. Let $(\mathcal{D}_i, f_i)$ $(i = 1, 2)$ be two submodular systems on E. The submodular intersection problem is to find a maximum common subbase $x \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2)$, where for each $i = 1, 2$ $\mathrm{P}(f_i)$ is the submodular polyhedron associated with $(\mathcal{D}_i, f_i)$. This problem is usually considered as a flow problem with boundary constraints described by submodular functions. First, we present a new algorithm by finding shortest augmenting paths in the auxiliary graph, which begins with a pair (y, z) of subbases of the given submodular systems and is convenient for adopting the preflow-push approach of A. V. Goldberg and R. E. Tarjan [43]. We require that $y \in \mathrm{B}(f_1)$, $z \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2)$ and $y \geq z$. Using the technique of selecting lexicographically shortest path, the initial common subbase z converges to a maximum common subbase by at most $O(n^3)$ path augmentations. Secondly, by using the basic ideas of the preflow-push method and starting from the same initial subbase pair (y, z), we devise a faster algorithm for the intersection problem, which requires $O(n^3)$ push and $O(n^2)$ relabeling operations in total by the largest-label implementation with a specific order on the out-going arc list of each vertex in the auxiliary graph.

In Chapter 3 an efficient cost scaling algorithm is presented for the independent assignment problem of M. Iri and N. Tomizawa [53], which is equivalent to the weighted matroid intersection problem of J. Edmonds [13].

Let $G = (V^+, V^-; A)$ be a *bipartite graph* with the left (right) end-vertex set V^+ (V^-) and the arc set A. For any $M \subseteq A$, $\partial^+ M$ $(\partial^- M)$ denotes the set of the left (right) end-vertices of arcs in M. A subset M of A is called a *matching* in the bipartite graph $G = (V^+, V^-; A)$ if $|\partial^+ M| = |M| = |\partial^- M|$.

Let $\mathcal{M}^+ = (V^+, \mathcal{I}^+)$ and $\mathcal{M}^- = (V^-, \mathcal{I}^-)$, respectively, be matroids on V^+ and V^- with families $\mathcal{I}^+ \subseteq 2^{V^+}$ and $\mathcal{I}^- \subseteq 2^{V^-}$ of independent sets. A cost function $c : A \to \mathbf{Z}$ is given, where $\mathbf{Z}$ is the set of all integers. An *independent matching* $M \subseteq A$ is a matching in G such that $\partial^+ M \in \mathcal{I}^+$ and $\partial^- M \in \mathcal{I}^-$. For a positive integer k, a *k-independent matching* M is an independent matching of cardinality k. An *optimal k-independent assignment* is a k-independent matching M having the minimum cost $c(M) = \sum_{e \in M} c(e)$ among all the k-independent matchings.

Our algorithm, which solves the optimal k-independent assignment problem, in general can be viewed as a generalization of J. B. Orlin and R. K. Ahuja's scaling algorithm [66] for the ordinary assignment problem. The cost scaling technique is adopted in our algorithm. The procedure for each scaling phase can be decomposed into two parts: an auction-like algorithm (see [4], [66]) and a successive shortest path algorithm. On a bipartite graph with n vertices and integer arc costs bounded by C, an optimal k-independent assignment can be found in $O(\sqrt{k}n^2 \log(kC))$ time by our algorithm under an independence oracle for matroids.

Several applications of the independent assignment problem are described in a section of Chapter 3.

In Chapter 4 we consider the problem of finding minimum-cost submodular flows. A submodular flow is a feasible flow on a flow network with the constraint that the flow boundary belongs to the base polyhedron described by a submodular system. The problem is to find a submodular flow of minimum cost. We give many combinatorial problems in a section, which can be formulated into a submodular flow problem.

An algorithm for the problem is constructed, which is a generalization of the algorithm for minimum-cost flows devised by A.V. Goldberg and R. E. Tarjan [44]. Our algorithm uses the technique of cost scaling and the concept of ε-optimality. The procedure for each scaling phase converts a 2ε-optimal submodular flow to an ε-optimal submodular pseudo-flow and then converts it to an ε-optimal submodular flow for successively smaller values of ε. In the procedure, two basic operations called Relabel and Push are performed. The complexity of the algorithm is proved to be pseudo-polynomial in general and the complexity becomes polynomial in case of minimum-cost 0-1 submodular flows.

In Chapter 5 we discuss some future topics in theories and algorithms, summarize the computation complexity of some existing algorithms and make some comparisons of our algorithms with other algorithms in theoretical and computatinal aspects.

1.2. Graphs and Network Flow Problems

Let V and A be two finite sets, where V is called the *vertex set* and A the *arc set.* ∂^+ and ∂^- are two functions from A to V. $\partial^+ a$ is called the *initial end-vertex* of a and $\partial^- a$ is called the *terminal end-vertex* of a. $G = (V, A; \partial^+, \partial^-)$ is called a *directed graph* with vertex set V and arc set A.

An arc a such that $\partial^+ a = \partial^- a$ is called a *self-loop* and arcs a_1, a_2 such that $\{\partial^+ a_1, \partial^- a_1\} = \{\partial^+ a_2, \partial^- a_2\}$ are called *parallel arcs*. An arc a is often expressed by the ordered pair $(\partial^+ a, \partial^- a)$ when there is no confusion from the context.

Sometimes, we do not distinguish $\partial^+ a$ from $\partial^- a$ for each $a \in A$. In this case we call the graph $G = (V, A)$ an *undirected graph* and call each $a \in A$ an *edge* instead of an arc.

A *path* in G is an alternating sequence $(v_0, a_1, v_1, a_2, \cdots, v_{k-1}, a_k, v_k)$ $(k \geq 0)$ of vertices v_i $(i = 0, 1 \cdots, k)$ and arcs a_i $(i = 1, 2, \cdots, k)$ such that $\{\partial^+ a_i, \partial^- a_i\} = \{v_{i-1}, v_i\}$. If $\partial^+ a_i = v_{i-1}$ and $\partial^- a_i = v_i$ $(i = 1, 2, \cdots, k)$, then we call the alternating sequence a *directed path* in G. A path or a directed path $(v_0, a_1, v_1, a_2, \cdots, v_{k-1}, a_k, v_k)$ in G with $v_0 = v_k$ is called a *cycle* or a *directed cycle* $(k \geq 1)$ in G, respectively. A graph G is said to be *connected* if for every two vertices u, $v \in V$ there exists a path from u to v. A connected graph G is called a *tree* if there exists no cycle in G.

We call a graph $H = (W, B; \partial^+{}_H, \partial^-{}_H)$ a *subgraph* of a graph $G = (V, A; \partial^+, \partial^-)$ if $W \subseteq V$, $B \subseteq A$ and $\partial^+{}_H$ and $\partial^-{}_H$ are, respectively, the restriction of ∂^+ and ∂^- to B. A maximal connected subgraph of G is called a *connected component* of G. A *forest* is a graph G such that every connected component of G is a tree. The *rank* of a graph $G = (V, A)$ is the number of its vertices minus the number of its connected components.

Let R be the set of all real numbers.

Shortest path problem. Given a directed graph $G = (V, A)$ and a length function $l : A \to \mathrm{R}$. For a directed path $P = (v_0, a_1, v_1, a_2, \cdots, v_{k-1}, a_k, v_k)$ of G, the length of P is defined to be the sum of $\sum_{i=1}^{k} l(a_i)$. For two specified vertices s, t in V, the shortest path problem is to find a directed path of the shortest length among all directed paths from s to t.

Define $\delta^+ v = \{a \mid a \in A, \partial^+ a = v\}$ and $\delta^- v = \{a \mid a \in A, \partial^- a = v\}$ for each $v \in V$.

Maximum flow problem. Given a network $\mathcal{N} = (G = (V, A); \bar{c}, \underline{c}, s, t)$ where s (t) is a specified source vertex (sink vertex) of V and $\bar{c}$, $\underline{c}$ $(\bar{c} \geq \underline{c})$ are, respectively, the upper and lower capacity functions from A to R. Let $\varphi : A \to \mathrm{R}$ be a flow function of $\mathcal{N}$. A flow function φ satisfying $\underline{c}(a) \leq \varphi(a) \leq \bar{c}(a)$ for each $a \in A$ is called a *feasible flow* on $\mathcal{N}$. The maximum flow problem is formulated as follows.

$$\begin{aligned}
\text{Maximize} \quad & \sum_{a\in\delta^+ s}\varphi(a) - \sum_{a\in\delta^- s}\varphi(a)\\
\text{subject to} \quad & \underline{c}(a) \le \varphi(a) \le \bar{c}(a) \quad (a \in A),\\
& \sum_{a\in\delta^+ v}\varphi(a) - \sum_{a\in\delta^- v}\varphi(a) = 0 \quad (v \in V - \{s,t\}).
\end{aligned} \tag{1.2.1}$$

That is, the maximum flow problem seeks a feasible flow which sends the maximum amount of flow from s to t. The function $\partial\varphi : V \to \mathrm{R}$ defined by

$$\partial\varphi(v) = \sum_{a\in\delta^+ v}\varphi(a) - \sum_{a\in\delta^- v}\varphi(a) \quad (v \in V) \tag{1.2.2}$$

is called the *boundary* of φ.

Minimum-cost flow problem. Consider a network $\mathcal{N} = (G = (V, A); \bar{c}, \underline{c}, \gamma, b)$, where $\bar{c}$ and $\underline{c}$ are the capacity functions defined as above, $\gamma : A \to \mathrm{R}$ is a cost function for $\mathcal{N}$ and $b : V \to \mathrm{R}$ is a demand function for $\mathcal{N}$. A feasible flow φ satisfying $\partial\varphi(v) = b(v)$ $(v \in V)$ is called a *feasible circulation* in network $\mathcal{N}$. The minimum-cost flow problem is formulated as follows.

$$\begin{aligned}
\text{Minimize} \quad & \sum_{a\in A}\gamma(a)\varphi(a)\\
\text{subject to} \quad & \underline{c}(a) \le \varphi(a) \le \bar{c}(a) \quad (a \in A),\\
& \partial\varphi(v) = b(v) \quad (v \in V).
\end{aligned} \tag{1.2.3}$$

Assignment problem. Let V^+ and V^- be two disjoint sets. $G = (V = V^+ \cup V^-, A; \partial^+, \partial^-)$ is said to be a *bipartite (directed) graph* if $\partial^+ a \in V^+$ and $\partial^- a \in V^-$ for each $a \in A$. Let a network $\mathcal{N}$ be $(G = (V^+ \cup V^-, A); \gamma)$ with a cost function $\gamma : A \to \mathrm{R}$. For a given positive integer k the *k-assignment problem* is described as follows.

$$\begin{aligned}
\text{Minimize} \quad & \sum_{a\in A}\gamma(a)\varphi(a)\\
\text{subject to} \quad & \varphi(a) = 0 \text{ or } 1 \ (a \in A),\\
& \partial\varphi(v) = 0 \text{ or } 1 \ (v \in V^+),\\
& \partial\varphi(v) = 0 \text{ or } -1 \ (v \in V^-),\\
& \sum_{a\in A}\varphi(a) = k.
\end{aligned} \tag{1.2.4}$$

Given a network $\mathcal{N} = (G = (V, A); \bar{c}, \underline{c})$ as above, for each $U \subseteq V$ define

$$\Delta^+ U = \{a \in A \mid \partial^+ a \in U,\ \partial^- a \in V - U\}, \tag{1.2.5}$$

$$\Delta^- U = \{a \in A \mid \partial^- a \in U,\ \partial^+ a \in V - U\}. \tag{1.2.6}$$

The function $\kappa_{\bar{c},\underline{c}} : 2^V \to \mathbf{R}$ defined by

$$\kappa_{\bar{c},\underline{c}}(U) = \sum_{a \in \Delta^+ U} \bar{c}(a) - \sum_{a \in \Delta^- U} \underline{c}(a) \tag{1.2.7}$$

is called the *cut function* of $\mathcal{N}$, where the sum over the empty set is defined to be equal to zero.

Theorem 1.2.1 (Hoffman): *There exists a feasible circulation in network $\mathcal{N} = (G = (V, A); \bar{c}, \underline{c}, b)$ with $\bar{c} \geq \underline{c}$ and $\sum_{v \in V} b(v) = 0$, if and only if for each $U \subseteq V$ we have*

$$\kappa_{\bar{c},\underline{c}}(U) \geq b(U) = \sum_{v \in U} b(v). \tag{1.2.8}$$

□

From Theorem 1.2.1 we can show

Theorem 1.2.2 (Gale): *Consider a bipartite graph $G = (V^+, V^-; A)$ and suppose we are given nonnegative (supply and demand) functions $s : V^+ \to \mathbf{R}$ and $d : V^- \to \mathbf{R}$. Then, there exists a nonnegative flow $\varphi : A \to \mathbf{R}$ such that*

$$\partial\varphi(v) \leq s(v),\ (v \in V^+) \text{ and } -\partial\varphi(v) \geq d(v),\ (v \in V^-) \tag{1.2.9}$$

if and only if for each $U^- \subseteq V^-$ we have

$$\sum_{v \in \{\partial^+ a \mid a \in \Delta^- U^-\}} s(v) \geq \sum_{v \in U^-} d(v). \tag{1.2.10}$$

□

1.3. Preliminaries from Theory of Matroids

Let E be a finite set. For any $X \subseteq E$ and $e \in E$ we write $X + e$ and $X - e$ instead of $X \cup \{e\}$ and $X - \{e\}$, respectively. The cardinality of $X \subseteq E$ is denoted by $|X|$. Suppose that a family $\mathcal{I}$ of subsets of E satisfies the following (I0)~(I2):

(I0) $\emptyset \in \mathcal{I}$.

(I1) $I_1 \subseteq I_2 \in \mathcal{I} \Longrightarrow I_1 \in \mathcal{I}$.

(I2) $I_1, I_2 \in \mathcal{I},\ |I_1| < |I_2| \Longrightarrow \exists e \in I_2 - I_1 : I_1 + e \in \mathcal{I}$.

The pair $\mathcal{M} = (E, \mathcal{I})$ is called a *matroid.* Each $I \in \mathcal{I}$ is called an *independent set* of matroid $(E, \mathcal{I})$ and $\mathcal{I}$ the family of independent sets of matroid $(E, \mathcal{I})$.

An independent set which is maximal with respect to set inclusion is called a *base.* Every base of matroid $(E, \mathcal{I})$ has the same cardinality. A subset of E which is not an independent set is called a *dependent set.* A minimal dependent set is called a *circuit.*

The *closure function* $\mathrm{cl} : 2^E \to 2^E$ of matroid $(E, \mathcal{I})$ is defined by

$$\mathrm{cl}(X) = X \cup \{v \in E \mid X + v \text{ is dependent}\} \tag{1.3.11}$$

for each $X \subseteq E$. The closure function cl satisfies the following:

(cl0) $\forall X \subseteq E : \ X \subseteq \mathrm{cl}(X)$.

(cl1) $\forall X,\ Y \subseteq E : \ X \subseteq \mathrm{cl}(Y) \Longrightarrow \mathrm{cl}(X) \subseteq \mathrm{cl}(Y)$.

(cl2) $\forall X \subseteq E,\ \forall e \in E : \ e' \in \mathrm{cl}(X + e) - \mathrm{cl}(X) \Longrightarrow e \in \mathrm{cl}(X + e') - \mathrm{cl}(X)$.

For any independent set $I \in \mathcal{I}$ and any element $e \in \mathrm{cl}(I) - I$, there exists a unique circuit contained in $I + e$. Such a circuit is called the *fundamental circuit* with respect to I and e, and is denoted by $\mathrm{C}(I|e)$. For any base B of $\mathcal{M}$ and any element $e \in B$, the set $E - \mathrm{cl}(B - e)$ is called the *fundamental cocircuit* with respect to B and e, and is denoted by $\mathrm{K}(B|e)$.

For each $X \subseteq E$ we define the *rank* $\rho(X)$ of X by

$$\rho(X) = \max\{|I| \mid I \subseteq X,\ I \in \mathcal{I}\}. \tag{1.3.12}$$

The rank function $\rho : 2^E \to \mathbf{Z}$ satisfies the following $(\rho 0) \sim (\rho 2)$:

(ρ0) $\forall X \subseteq E: \ 0 \leq \rho(X) \leq |X|$.

(ρ1) $X \subseteq Y \subseteq E \Longrightarrow \rho(X) \leq \rho(Y)$.

(ρ2) $\forall X, \ Y \subseteq E: \ \rho(X) + \rho(Y) \geq \rho(X \cup Y) + \rho(X \cap Y)$.

We describe three examples of matroids. These are the graphic matroid, the linear matroid and the partition matroid.

For a graph $G = (V, A)$ with a vertex set V and arc set A let $\mathcal{I}(G)$ be the set of those arc subsets each of which does not contain any cycle of G. Then $\mathcal{M}(G) = (A, \mathcal{I}(G))$ is a matroid with the family $\mathcal{I}(G)$ of independent sets. A matroid which can be obtained in this way is called a *graphic matroid.*

For a set $F = \{a_1, a_2, \cdots, a_m\}$ of vectors in R^n let $\mathcal{I}(F)$ be the set of those subsets of A each of which consists of linear independent vectors. $\mathcal{M}(F) = (F, \mathcal{I}(F))$ is a matroid with the family $\mathcal{I}(F)$ of independent sets. A matroid which can be obtained in this way is called a *linear matroid.* A graphic matroid is a linear matroid.

Let $E = E_1 \cup E_2 \cup \cdots \cup E_k$ be a union of k disjoint finite sets and let $r_1, r_2, \cdots, r_k$ be given nonnegative integers. Let $\mathcal{I}$ be the set of those subsets I of E that satisfies the property that for all $i = 1, 2, \cdots k$, I contains no more than r_i elements of E_i. $(E, \mathcal{I})$ is a matroid and is called a *partition matroid.*

1.4. Preliminaries from Theory of Submodular Systems

In this section we give definitions of terms related to submodular systems. Also some basic preliminary results are given. This section is largely due to [34].

Let E be a nonempty finite set and $\mathcal{D}$ be a collection of subsets of E which forms a distributive lattice with set union and intersection as the lattice operations, join and meet, i.e., for each $X, \ Y \in \mathcal{D}$ we have $X \cup Y, \ X \cap Y \in \mathcal{D}$. Let $f: \mathcal{D} \rightarrow \mathrm{R}$ be a *submodular function* on the distributive lattice $\mathcal{D}$, i.e.,

$$\forall X, \ Y \in \mathcal{D}: \ f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y). \qquad (1.4.1)$$

If $\emptyset, \ E \in \mathcal{D}$ and $f(\emptyset) = 0$, we call the pair $(\mathcal{D}, f)$ a *submodular system* on E. Function f is called the *rank function* of $(\mathcal{D}, f)$. From Property (ρ2) of a matroid rank function defined in (1.3.12), the rank function of a matroid $(E, \mathcal{I})$ satisfies (1.4.1) and thus is a submodular function on 2^E. Another example of submodular function is the cut function of a capacitated network defined in (1.2.7).

Define a polyhedron in R^E by

$$\mathrm{P}(f) = \{x \mid x \in \mathrm{R}^E,\ \forall X \in \mathcal{D} :\ x(X) \leq f(X)\}, \tag{1.4.2}$$

where R^E is the set of all functions from E to R and for any $X \subseteq E$ we have $x(X) = \sum_{e \in X} x(e)$ together with the convention $x(\emptyset) = 0$. We call $\mathrm{P}(f)$ the *submodular polyhedron* associated with submodular system $(\mathcal{D}, f)$. Also a vector in the submodular polyhedron $\mathrm{P}(f)$ is called a *subbase* of $(\mathcal{D}, f)$.

Define

$$\mathrm{B}(f) = \{x \mid x \in \mathrm{P}(f),\ x(E) = f(E)\}. \tag{1.4.3}$$

We call $\mathrm{B}(f)$ the *base polyhedron* associated with submodular system $(\mathcal{D}, f)$. A vector in $\mathrm{B}(f)$ is called a *base* of $(\mathcal{D}, f)$.

We give some other examples of submodular functions and their applications.

Given a directed graph $G = (V, A)$, define, for each $\emptyset \subset X \subset V$, the *in-degree function* $\rho(X)$ as the number of arcs in A directed from $V - X$ to X and $\rho(\emptyset) = \rho(V) = 0$. Then, ρ is a submodular function on subsets of V.

Definition 1.4.1: A directed graph $G = (V, A)$ is said to be *k-strongly-connected* if for any pair $u, v \in V$, there exists k arc disjoint directed paths from u to v.

Theorem 1.4.2 (Nash-Williams [65]): *A directed graph $G = (V, A)$ is k-strongly-connected if and only if $\rho(X) \geq k$ for $\emptyset \subset X \subset V$.* □

Another example is the *Entropy function.* Let $E = \{x_1, x_2, \cdots, x_n\}$ be the set of n random variables of values in $\{1, 2, \cdots, N\}$. For each nonempty set $X \subseteq E$ (for convenience, say $X = \{x_1, x_2, \cdots, x_k\}$), define

$$h(X) = -\sum_{i_1=1}^{N} \cdots \sum_{i_k=1}^{N} p(x_1 = i_1, \cdots, x_k = i_k) \log_2 p(x_1 = i_1, \cdots, x_k = i_k) \tag{1.4.4}$$

and $h(\emptyset) = 0$ where $p(x_1 = i_1, \cdots, x_k = i_k)$ is the probability for event $x_1 = i_1, x_2 = i_2, \cdots, x_k = i_k$ with the convention $0 \log_2 0 = 0$. Then $h(X)$ is a submodular function which plays an important rôle in some problems on information theories [29], [47].

Lemma 1.4.3: *For any subbase $x \in \mathrm{P}(f)$ define*

$$\mathcal{D}(x) = \{X \mid X \in \mathcal{D},\ x(X) = f(X)\}. \tag{1.4.5}$$

Then,

$$\forall\, X,\ Y \in \mathcal{D}(x) :\ X \cup Y,\ X \cap Y \in \mathcal{D}(x). \tag{1.4.6}$$

Proof: For any $X,\ Y \in \mathcal{D}(x)$,

$$0 = f(X) - x(X) + f(Y) - x(Y) \geq f(X \cup Y) - x(X \cup Y) + f(X \cap Y) - x(X \cap Y) \geq 0,$$

where

$$\min\{f(X \cup Y) - x(X \cup Y), f(X \cap Y) - x(X \cap Y)\} \geq 0. \tag{1.4.7}$$

Hence we must have $f(X \cup Y) = x(X \cup Y)$ and $f(X \cap Y) = x(X \cap Y)$, i.e.,

$$X \cup Y,\quad X \cap Y \in \mathcal{D}(x). \tag{1.4.8}$$

□

For any $x \in \mathrm{P}(f)$ define

$$\mathrm{sat}(x) = \bigcup\{X \mid X \in \mathcal{D},\ x(X) = f(X)\}. \tag{1.4.9}$$

We call sat: $\mathrm{P}(f) \to 2^E$ the *saturation function*.

For any subbase $x \in \mathrm{P}(f)$ and $e \in \mathrm{sat}(x)$ define

$$\mathrm{dep}(x, e) = \bigcap\{X \mid e \in X \in \mathcal{D},\ x(X) = f(X)\}. \tag{1.4.10}$$

For $x \in \mathrm{P}(f)$ and $e \in E - \mathrm{sat}(x)$ we define $\mathrm{dep}(x, e) = \emptyset$. We call dep: $\mathrm{P}(f) \times E \to 2^E$ the *dependence function*.

By Lemma 1.4.3 we have for any $x \in \mathrm{P}(f)$ and $e \in E$

$$f(\mathrm{sat}(x)) = x(\mathrm{sat}(x)),\quad f(\mathrm{dep}(x, e)) = x(\mathrm{dep}(x, e)). \tag{1.4.11}$$

For any $x \in \mathrm{P}(f)$ and $e \in E - \mathrm{sat}(x)$ the *saturation capacity* $\hat{c}(x, e)$ is defined by

$$\hat{c}(x, e) = \min\{f(X) - x(X) \mid e \in X \in \mathcal{D}\}. \tag{1.4.12}$$

Lemma 1.4.4: *For a nonnegative α, $x \in \mathrm{P}(f)$, $e \in \mathrm{sat}(x)$ and $e' \in \mathrm{dep}(x,e) - \{e\}$, we have $x + \alpha\chi_e \in \mathrm{P}(f)$ if and only if $0 \le \alpha \le \hat{c}(x,e)$, where $\chi_e \in \mathrm{R}^E$ is defined by $\chi_e(e) = 1$ and $\chi_e(e') = 0$ for $e' \in E - \{e\}$.*

Proof: For $0 \le \alpha \le \hat{c}(x,e)$ and $X \in \mathcal{D}$, if $e \notin X$, then

$$(x + \alpha\chi_e)(X) = x(X) \le f(X), \tag{1.4.13}$$

and if $e \in X$, then by the definition of $\hat{c}(x,e)$

$$(x + \alpha\chi_e)(X) = x(X) + \alpha \le f(X). \tag{1.4.14}$$

This implies $x + \alpha\chi_e \in \mathrm{P}(f)$.

On the other hand, for a nonnegative α such that $x + \alpha\chi_e \in \mathrm{P}(f)$ we have $\alpha \le f(X) - x(X)$ for any X satisfying $e \in X \in \mathcal{D}$ since $(x + \alpha\chi_e)(X) \le f(X)$. That is, $\alpha \le \hat{c}(x,e)$. □

For any $x \in \mathrm{P}(f)$, $e \in \mathrm{sat}(x)$ and $e' \in \mathrm{dep}(x,e) - \{e\}$ the *exchange capacity* $\tilde{c}(x,e,e')$ is defined by

$$\tilde{c}(x,e,e') = \min\{f(X) - x(X) \mid e \in X \in \mathcal{D},\ e' \notin X\}. \tag{1.4.15}$$

Lemma 1.4.5: *For a nonnegative α and $x \in \mathrm{P}(f)$, we have $x + \alpha(\chi_e - \chi_{e'}) \in \mathrm{P}(f)$ if and only if $0 \le \alpha \le \tilde{c}(x,e,e')$.*

Proof: For α such that $0 \le \alpha \le \tilde{c}(x,e,e')$ and $X \in \mathcal{D}$,
(i) if $e,\ e' \in X$ or $e,\ e' \notin X$, then

$$(x + \alpha(\chi_e - \chi_{e'}))(X) = x(X) \le f(X); \tag{1.4.16}$$

(ii) if $e' \in X,\ e \notin X$, then

$$(x + \alpha(\chi_e - \chi_{e'}))(X) = x(X) - \alpha \le f(X) - \alpha \le f(X); \tag{1.4.17}$$

(iii) if $e \in X,\ e' \notin X$, then by the definition of $\tilde{c}(x,e,e')$,

$$(x + \alpha(\chi_e - \chi_{e'}))(X) = x(X) + \alpha \le x(X) + f(X) - x(X) = f(X). \tag{1.4.18}$$

Hence $x + \alpha(\chi_e - \chi_{e'}) \in \mathrm{P}(f)$.

On the other hand, for a nonnegative α such that $x + \alpha(\chi_e - \chi_{e'}) \in \mathrm{P}(f)$ we have $\alpha \leq f(X) - x(X)$ for any $X \in \mathcal{D}$ satisfying $e \in X$ and $e' \notin X$ since $(x + \alpha(\chi_e - \chi_{e'}))(X) \leq f(X)$. That is, $\alpha \leq \tilde{c}(x, e, e')$. □

We call the transformation of x into $x + \alpha(\chi_e - \chi_{e'})$ as in Lemma 1.4.5 an *elementary transformation* of x.

Lemma 1.4.6: *The base polyhedron* $\mathrm{B}(f)$ *is the set of all the maximal subbases of* $(\mathcal{D}, f)$. *In particular,* $\mathrm{B}(f) \neq \emptyset$. *Here, the partial order among vectors in* R^E *is defined by:* $x \leq y \iff \forall e \in E : x(e) \leq y(e)$.

Proof: Denote by $\mathrm{B}'(f)$ the set of all the maximal subbases of $(\mathcal{D}, f)$. Any $x \in \mathrm{B}(f)$ is maximal in $\mathrm{P}(f)$ since $x(E) = f(E)$, so that we have $\mathrm{B}(f) \subseteq \mathrm{B}'(f)$. Conversely, for any $x \in \mathrm{B}'(f)$ we have $\mathrm{sat}(x) = E$ due to the maximality of x. It follows that $x(E) = x(\mathrm{sat}(x)) = f(\mathrm{sat}(x)) = f(E)$, i.e., $x \in \mathrm{B}(f)$. Therefore, $\mathrm{B}'(f) \subseteq \mathrm{B}(f)$, and hence we have $\mathrm{B}'(f) = \mathrm{B}(f)$. □

The following lemmas are obtained by a direct adaptation of the results shown in [28] for polymatroids.

Lemma 1.4.7: *Suppose* $x \in \mathrm{P}(f)$, $u \in \mathrm{sat}(x)$ *and* $v \in \mathrm{dep}(x, u) - \{u\}$. *For any* $\alpha \in \mathrm{R}$ *such that* $0 < \alpha \leq \tilde{c}(x, u, v)$ *define* $y = x + \alpha(\chi_u - \chi_v)$. *Then,* $y \in \mathrm{P}(f)$ *and*

$$\mathrm{sat}(y) = \mathrm{sat}(x). \tag{1.4.19}$$

Proof: From the definition of the exchange capacity we have $y \in \mathrm{P}(f)$. Also, since for any $X \in \mathcal{D}$ such that $X \supseteq \mathrm{sat}(x)$ we have $y(X) = x(X)$ and $\mathrm{sat}(x)$ is the unique maximal tight set X such that $x(X) = f(X)$, we have $\mathrm{sat}(y) = \mathrm{sat}(x)$. □

Lemma 1.4.8: *Under the same assumption as in Lemma 1.4.7,*

$$\hat{c}(y, w) = \hat{c}(x, w) \quad (w \in E - \mathrm{sat}(x)). \tag{1.4.20}$$

Proof: Put $X_0 = \mathrm{sat}(x)$ $(= \mathrm{sat}(y))$. Since $x(X_0) = f(X_0)$ and $y(X_0) = f(X_0)$, we have for any $X \in \mathcal{D}$

$$f(X) - y(X) = f(X) - y(X) + f(X_0) - y(X_0)$$

$$\begin{aligned} &\geq f(X\cup X_0)-y(X\cup X_0)+f(X\cap X_0)-y(X\cap X_0)\\ &\geq f(X\cup X_0)-y(X\cup X_0)\\ &= f(X\cup X_0)-x(X\cup X_0), \end{aligned} \tag{1.4.21}$$

and similarly,

$$f(X)-x(X)\geq f(X\cup X_0)-x(X\cup X_0). \tag{1.4.22}$$

Hence the lemma follows from (1.4.12). □

Lemma 1.4.9: *For any* $x\in \mathrm{P}(f)$ *let* u_1, u_2 *and* v_2 *be three distinct elements of* E *such that*

$$u_i\in \mathrm{sat}(x)\quad (i=1,2), \tag{1.4.23}$$

$$v_2\in \mathrm{dep}(x,u_2),\quad v_2\notin \mathrm{dep}(x,u_1). \tag{1.4.24}$$

For any $\alpha\in \mathrm{R}$ *such that* $0<\alpha\leq \tilde{c}(x,u_2,v_2)$ *define*

$$y=x+\alpha(\chi_{u_2}-\chi_{v_2}). \tag{1.4.25}$$

Then we have $u_1\in \mathrm{sat}(y)$ *and*

$$\mathrm{dep}(y,u_1)=\mathrm{dep}(x,u_1). \tag{1.4.26}$$

Proof: From Lemma 1.4.7 we have $u_1\in \mathrm{sat}(y)$. Also we have $u_2\notin \mathrm{dep}(x,u_1)$, since otherwise we would have $\mathrm{dep}(x,u_2)\subseteq \mathrm{dep}(x,u_1)$ by the minimality of $\mathrm{dep}(x,u_2)$ and hence $v_2\in \mathrm{dep}(x,u_1)$. Therefore, putting $X_0=\mathrm{dep}(x,u_1)$, we have $y(X_0)=x(X_0)=f(X_0)$ and $y(X)=x(X)$ for any $X\in\mathcal{D}$ with $X\subseteq X_0$. (1.4.26) follows from the definition of the dependence function. □

Lemma 1.4.10: *For any* $x\in \mathrm{P}(f)$ *let* u_1, u_2, v_1 *and* v_2 *be four distinct elements of* E *satisfying (1.4.23), (1.4.24) and*

$$v_1\in \mathrm{dep}(x,u_1). \tag{1.4.27}$$

Then for the vector y *defined by (1.4.25) for any* $0<\alpha\leq\tilde{c}(x,u_2,v_2)$ *we have*

$$\tilde{c}(y,u_1,v_1)=\tilde{c}(x,u_1,v_1). \tag{1.4.28}$$

Proof: For any $z \in \mathrm{P}(f)$ and $X_0 \in \mathcal{D}$ such that $z(X_0) = f(X_0)$ we have

$$f(X) - z(X) \geq f(X \cap X_0) - z(X \cap X_0) \quad (X \in \mathcal{D}). \tag{1.4.29}$$

For $X_0 \equiv \mathrm{dep}(x, u_1)$, we have

$$y(X_0) = x(X_0) = f(X_0) \tag{1.4.30}$$

and since $u_2,\ v_2 \notin \mathrm{dep}(x, u_1)$, we have

$$y(X) = f(X) \quad (X \subseteq X_0,\ X \in \mathcal{D}). \tag{1.4.31}$$

Since (1.4.29) holds for $z = x,\ y$, (2.27) follows from (1.4.30) and (1.4.31). □

Lemma 1.4.11: *For any $x \in \mathrm{P}(f)$ let $u_i,\ v_i\ (i = 1, 2, \cdots, q)$ be $2q$ distinct elements of E such that*

$$u_i \in \mathrm{sat}(x),\ \ v_i \in \mathrm{dep}(x, u_i)\ (i = 1, 2, \cdots, q), \tag{1.4.32}$$

$$v_j \notin \mathrm{dep}(x, u_i)\ (1 \leq i < j \leq q). \tag{1.4.33}$$

For any $\alpha_i\ (i = 1, 2, \cdots, q)$ satisfying $0 < \alpha_i \leq \tilde{c}(x, u_i, v_i)\ (i = 1, 2, \cdots, q)$ define a vector $y \in \mathrm{R}^E$ by

$$y = x + \sum_{i=1}^{q} \alpha_i(\chi_{u_i} - \chi_{v_i}). \tag{1.4.34}$$

Then,

$$y \in \mathrm{P}(f), \quad \mathrm{sat}(y) = \mathrm{sat}(x), \tag{1.4.35}$$

$$\hat{c}(y, w) = \hat{c}(x, w) \quad (w \in E - \mathrm{sat}(x)). \tag{1.4.36}$$

Proof: Considering the elementary transformations in the order of the pairs

$$(u_q, v_q), (u_{q-1}, v_{q-1}), \cdots, (u_1, v_1),$$

the present lemma can be shown by repeatedly applying Lemmas 1.4.7∼1.4.10. □

Lemma 1.4.12 ([34, p. 119]): *For an arbitrary $x \in \mathrm{P}(f)$, let $u,\ v \in E$, $0 < \alpha \leq \tilde{c}(x, u, v)$ and $y \in \mathrm{P}(f)$ be such that*

$$u \in \mathrm{sat}(x), \quad v \in \mathrm{dep}(x, u), \quad y = x + \alpha(\chi_u - \chi_v). \tag{1.4.37}$$

Consider $w,\ s \in E$ such that

$$s \notin \mathrm{dep}(x, w), \quad s \in \mathrm{dep}(y, w). \tag{1.4.38}$$

then

$$s \in \mathrm{dep}(x, u), \quad v \in \mathrm{dep}(x, w). \tag{1.4.39}$$

□

A function $g : \mathcal{D} \to \mathbf{R}$ on the distributive lattice $\mathcal{D}$ is called a *supermodular function* if $-g$ is a submodular function, i.e.,

$$\forall X,\ Y \in \mathcal{D}:\ g(X) + g(Y) \leq g(X \cup Y) + g(X \cap Y). \tag{1.4.40}$$

Given a submodular function $f : \mathcal{D} \to \mathbf{R}$, we define

$$\overline{\mathcal{D}} = \{Y \mid E - Y \in \mathcal{D}\} \tag{1.4.41}$$

and $f^{\#}$ by

$$f^{\#}(Y) = f(E) - f(E - Y) \tag{1.4.42}$$

for each $Y \in \overline{\mathcal{D}}$. Since for any $X,\ Y \in \overline{\mathcal{D}}$

$$\begin{aligned}
&f^{\#}(X) + f^{\#}(Y) \\
&= 2f(E) - (f(E - X) + f(E - Y)) \\
&\leq 2f(E) - f((E - X) \cup (E - Y)) - f((E - X) \cap (E - Y)) \\
&= f(E) - f(E - X \cap Y) + f(E) - f(E - X \cup Y) \\
&= f^{\#}(X \cup Y) + f^{\#}(X \cap Y),
\end{aligned} \tag{1.4.43}$$

we see that $f^{\#}$ is a supermodular function on $\overline{\mathcal{D}}$. $(\overline{\mathcal{D}}, f^{\#})$ is called the *dual supermodular system* of $(\mathcal{D}, f)$.

Define a polyhedron in $\mathbf{R}^E$ by

$$\mathrm{P}(f^{\#}) = \{x \mid x \in \mathbf{R}^E,\ \forall X \in \overline{\mathcal{D}}:\ x(X) \geq f^{\#}(X)\}. \tag{1.4.44}$$

We call $\mathrm{P}(f^{\#})$ the *supermodular polyhedron* associated with supermodular system $(\overline{\mathcal{D}}, f^{\#})$.

We describe two practical examples of supermodular functions.

Let $A_1, \cdots, A_n$ be random events, $S = \{A_1, \cdots, A_n\}$, and let, for

$$X = \{A_{i_1}, \cdots, A_{i_k}\} \subseteq S, \tag{1.4.45}$$

$g(X) = p(A_{i_1}, \cdots, A_{i_k})$ denote the probability that all events in X occur. Then g is a supermodular function on the subsets of S. This fact is related to sieving techniques.

The second example is on convex games. Let $E = \{1, 2, \cdots, n\}$ be n players and $v : 2^E \to \mathbf{R}$ be a characteristic function where each $X \subseteq E$ is considered as a coalition.

A characteristic-function game (E, v) is called a *convex game* [73] if the characteristic function v is a supermodular function. The *core* of the game (E, v) is the set of payoff vectors defined by

$$\{x \mid x \in \mathbf{R}^E,\ \forall X \in E:\ x(X) \geq v(X),\ x(E) = v(E)\}. \tag{1.4.46}$$

Next, we introduce two fundamental operations on a submodular system $(\mathcal{D}, f)$ called reduction and contraction by vectors.

For any vector $x \in \mathbf{R}^E$ define $f^x : 2^E \to \mathbf{R}$ by

$$f^x(X) = \min\{f(Z) + x(X - Z) \mid X \supseteq Z \in \mathcal{D}\} \tag{1.4.47}$$

for each $X \subseteq E$. The function f^x is a submodular function on 2^E. Define

$$\mathrm{P}(f)^x = \{y \mid y \in \mathrm{P}(f),\ y \leq x\} \tag{1.4.48}$$

and if $x \in \mathrm{P}(f^\#)$, define

$$\mathrm{B}(f)^x = \{y \mid y \in \mathrm{B}(f),\ y \leq x\}, \tag{1.4.49}$$

then we have $\mathrm{P}(f^x) = \mathrm{P}(f)^x$ and $\mathrm{B}(f^x) = \mathrm{B}(f)^x$. We call the submodular system $(2^E, f^x)$ the *reduction* of $(\mathcal{D}, f)$ by vector x.

For a submodular system $(\mathcal{D}, f)$ and any vector $x \in \mathrm{P}(f)$ define $f_x : 2^E \to \mathbf{R}$ by

$$f_x(X) = \min\{f(Z) - x(Z - X) \mid X \subseteq Z \in \mathcal{D}\} \tag{1.4.50}$$

for each $X \subseteq E$. The function f_x is a submodular function on 2^E. Define

$$[\mathrm{P}(f)]_x = \{y \mid y \in \mathrm{R}^E,\ x \vee y \in \mathrm{P}(f)\}, \tag{1.4.51}$$

$$\mathrm{B}(f)_x = \{y \mid y \in \mathrm{B}(f),\ y \geq x\}, \tag{1.4.52}$$

where $x \vee y$ is the vector in R^E defined by $(x \vee y)(e) = \max\{x(e), y(e)\}$ $(e \in E)$. Then we have $\mathrm{P}(f_x) = [\mathrm{P}(f)]_x$ and $\mathrm{B}(f_x) = \mathrm{B}(f)_x$. We call the submodular system $(2^E, f_x)$ the *contraction* of $(\mathcal{D}, f)$ by vector x.

Lemma 1.4.13: *For each $z \in \mathrm{B}(f_x)$, we have $z - x \geq \mathbf{0}$.*

Proof: For each $v \in E$, $z \in \mathrm{B}(f_x)$ implies that

$$z(v) = z(E) - z(E - v) = f_x(E) - z(E - v) \geq f_x(E) - f_x(E - v). \tag{1.4.53}$$

From (1.4.50) and $f_x(E) = f(E)$, we have

$$f_x(E - v) \leq f(E) - x(E - (E - v)) = f(E) - x(v). \tag{1.4.54}$$

The present lemma follows from (1.4.53) and (1.4.54). □

1.5. Greedy Algorithm

In this section, we introduce an efficient algorithm called the greedy algorithm for the following linear optimization problem. The greedy algorithm is a procedure that at each stage of the algorithm it chooses the locally most desirable element then available without regard to future consequences. J. B. Kruskal [57] presented a greedy algorithm to find a minimum weight spanning subgraph of a directed graph. It was extended by R. Rado [68] to the problem of finding an optimal independent set of a matroid.

Given a matroid $\mathcal{M} = (E, \mathcal{I})$ and a weight function $w : E \rightarrow \mathbf{R}$. Let $\mathcal{B}$ be the set of all bases of $\mathcal{M}$. A linear optimization problem is defined as

$$(\mathrm{P_m})\quad \text{Minimize}\quad \sum_{e \in B} w(e) \tag{1.5.55}$$

$$\text{subject to}\quad B \in \mathcal{B}. \tag{1.5.56}$$

The solution B of the above problem is called a *minimum-weight base* with respect to the weight function w.

Given a submodular system $(\mathcal{D}, f)$ on E and a weight function $w : E \to \mathbf{R}$, a linear optimization problem is defined as

$$(\mathrm{P}^1_\mathrm{s}) \quad \text{Minimize} \quad \sum_{e \in E} w(e)x(e) \tag{1.5.57}$$

$$\text{subject to} \quad x \in \mathrm{B}(f). \tag{1.5.58}$$

For Problem $(\mathrm{P_m})$ the following lemma gives an optimality condition for a base.

Lemma 1.5.1: *Given a weight function $w : E \to \mathrm{R}$ (the set of all real numbers). A base B of $\mathcal{M} = (E, \mathcal{I})$ satisfies $w(B) = \min\{w(B') \mid B' \in \mathcal{B}\}$ if and only if for each pair (u, v) such that $u \in \mathrm{C}(B|v)$ we have $w(u) \leq w(v)$.* □

The following is a greedy algorithm for Problem $(\mathrm{P_m})$.
A greedy algorithm (See J. Edmonds [14])
Input: Let $E = \{e_1, e_2, \cdots, e_n\}$ be such that

$$w(e_1) \leq w(e_2) \leq \cdots \leq w(e_n) \tag{1.5.59}$$

for a proper ordering of E.
Output: A minimum-weight base I_n.
Step 0: $I_0 \leftarrow \emptyset$, $i \leftarrow 1$.
Step 1: If $I_{i-1} \cup \{e_i\} \in \mathcal{I}$, put $I_i \leftarrow I_{i-1} \cup \{e_i\}$. Otherwise, put $I_i \leftarrow I_{i-1}$.
Step 2: If $i = n$, output I_n and stop. Otherwise, set $i \leftarrow i + 1$ and go to Step 1.
(End)

Theorem 1.5.2 (Rado [68]): *The greedy algorithm for matroids terminates with a minimum-weight base.* □

For a submodular system $(\mathcal{D}, f)$ on E, given any two elements u, $v \in E$, $X \in E$ is said to be a *$u\bar{v}$-set* if $u \in X$ and $v \notin X$. We define a directed graph $G(\mathcal{D}) = (E, A)$ by setting $A = \{(u, v) \mid \text{there is no } u\bar{v}\text{-set in } \mathcal{D}\}$. We call two element u, $v \in E$ *equivalent* if both (u, v) and (v, u) are in A of $G(\mathcal{D})$. By this definition, we have an equivalence relation on E, i.e.,
u is equivalent to $v \Longrightarrow v$ is equivalent to u,
u is equivalent to v and v is equivalent to $w \Longrightarrow u$ is equivalent to w.

Denote the equivalence classes by $E_1, \cdots, E_k$ which form a partition of E. Let $\preceq_{\mathcal{D}}$ be an order on the set $\Pi(\mathcal{D}) = \{E_i \mid i = 1, \cdots, k\}$ defined by: $E_i \preceq_{\mathcal{D}} E_j$ if and only if there exists an arc in A from a vertex in E_j to a vertex in E_i or $i = j$. We see that $\preceq_{\mathcal{D}}$ is a partial order on $\Pi(\mathcal{D})$, i.e., it satisfies that
(i) $E_i \preceq_{\mathcal{D}} E_i$ $(i = 1, \cdots, k)$,
(ii) $E_i \preceq_{\mathcal{D}} E_j$, $E_j \preceq_{\mathcal{D}} E_i \Longrightarrow E_i = E_j$,
(iii) $E_i \preceq_{\mathcal{D}} E_j$, $E_j \preceq_{\mathcal{D}} E_l \Longrightarrow E_i \preceq_{\mathcal{D}} E_l$.

We denote by $\mathcal{P} = (\Pi(\mathcal{D}), \preceq_{\mathcal{D}})$ for this partially ordered partition of E. If $k = n = |E|$, i.e., $\Pi(\mathcal{D})$ consists of singletons, then we call $\mathcal{D}$ *simple*. If u, v is equivalent, then for an arbitrary $x \in \mathrm{B}(f)$, $y = x + \alpha(\chi_u - \chi_v) \in \mathrm{B}(f)$ holds for any $\alpha \in \mathbf{R}$. It follows that

$$\sum_{e \in E} w(e)y(e) = \alpha(w(u) - w(v)) + \sum_{e \in E} w(e)x(e). \tag{1.5.60}$$

If $w(u) \neq w(v)$, $\sum_{e \in E} w(e)y(e)$ can take an arbitrary value by choosing a proper value of α. Hence there is no optimal solution for Problem (P_s^1) in this case. When for each equivalent pair u, v, $w(u) = w(v)$ holds, then we can treat E_i $(i = 1, \cdots, k)$ as one element e_i for Problem (P_s^1). Therefore, without loss of generality we suppose that $\mathcal{D}$ is simple and consider $\preceq_{\mathcal{D}}$ as a partial order on E. We have

Theorem 1.5.3 (Fujishige and Tomizawa [36]): *Problem* (P_s^1) *has a finite optimal solution if and only if* $w : E \to \mathrm{R}$ *is a monotone nondecreasing function from* $\mathcal{P} = (E, \preceq_{\mathcal{D}})$ *to* $(\mathrm{R}, \leq)$, *i.e.,* $\forall e, e' \in E : e \preceq_{\mathcal{D}} e' \Longrightarrow w(e) \leq w(e')$. □

Corollary 1.5.4 (see Fujishige [34]): *When the weight function* w *satisfies the condition in Theorem* 1.5.3, *an optimal solution* x *of* (P_s^1) *is given by*

$$x(e_i) = f(A_i) - f(A_{i-1}) \ (i = 1, 2, \cdots, n), \tag{1.5.61}$$

where $\emptyset = A_0 \subset A_1 \subset \cdots \subset A_n = E$ *is such that* $\{e_i\} = A_i - A_{i-1}$,

$$e_1 \preceq_{\mathcal{D}} e_2 \preceq_{\mathcal{D}} \cdots \preceq_{\mathcal{D}} e_n \tag{1.5.62}$$

and

$$w(e_1) \leq w(e_2) \leq \cdots \leq w(e_n). \tag{1.5.63}$$

□

Theorem 1.5.5 (see Fujishige [34]): *A base $x \in \mathrm{B}(f)$ is an optimal solution of Problem* (P_s^1) *if and only if for each e, $e' \in E$ such that $e' \in \mathrm{dep}(x, e)$ we have $w(e) \geq w(e')$.* □

If w is a monotone nondecreasing function from $\mathcal{P} = (E, \preceq_{\mathcal{D}})$ to $(\mathrm{R}, \leq)$ and $E = \{e_1, e_2, \cdots, e_n\}$ satisfies the condition that $e_i \preceq_{\mathcal{D}} e_j$ implies $i \leq j$ $(i, j = 1, 2 \cdots, n)$, then from Corollary 1.5.4 we can devise the following algorithm.

A greedy algorithm

Step 1: Find an ordering $(e_1, e_2, \cdots, e_n)$ of E such that $e_i \preceq_{\mathcal{D}} e_j$ implies $i \leq j$ $(i, j = 1, 2, \cdots, n)$.

Step 2: Compute a vector $x \in \mathrm{R}^E$ by

$$x(e_i) = f(S_i) - f(S_{i-1}) \quad (i = 1, 2, \cdots, n), \tag{1.5.64}$$

where for each $i = 1, 2 \cdots, n$ S_i is the set of the first i elements of $(e_1, e_2, \cdots, e_n)$ and $S_0 = \emptyset$. Then x is a minimum-weight base of $(\mathcal{D}, f)$ with respect to weight w.

(End)

Chapter 2.

New Algorithms for the Intersection Problem of Submodular Systems

2.1. Introduction

We consider the problem of finding a maximum common subbase of two submodular systems, called the intersection problem (see [34]); the precise description of the problem will be given at the end of this section. The intersection problem is a generalization of the so-called matroid intersection problem and has a lot of practical applications (see [50], [51], [64], [70]). An algorithm for the intersection problem was presented in [34] (also see [28], [33],[72],[77]). Starting from a common subbase and finding the lexicographically shortest augmenting paths $O(n^3)$ times in auxiliary networks, one can reach a solution of the problem.

The submodular intersection problem is closely related to the maximum flow problem. In the network theory, there are two main approaches to solve the maximum flow problem—algorithms by finding augmenting paths and algorithms by applying the basic preflow-push and relabeling operations. The latter is relatively new and has been shown to be very efficient for maximum flow problems. In the so called "neoflow" problems [32], which include the submodular intersection problem as a special case, several algorithms for such problems by finding shortest augmenting paths have been presented. The main purpose of our research in this chapter is to introduce the preflow-push approach to the intersection problem of submodular systems.

In this chapter we first present a new algorithm by path augmentations, but it begins with a pair of subbases and is convenient for adopting the preflow-push approach of Goldberg and Tarjan [43]. Secondly, by using the basic ideas of the preflow-push method, we devise a faster algorithm for the intersection problem which requires $\mathrm{O}(n^3)$ push and $\mathrm{O}(n^2)$ relabeling operations in total by the largest-label implementation with a specific order (related to the lexicographical ordering of P. Schönsleben's [72]) on the arc list of each vertex in the auxiliary graph. Instead of using the lexicographical ordering, we propose a first-active implementation. The algorithm finds a maximum common subbase of two given submodular systems by $\mathrm{O}(n^4)$ saturating push, $\mathrm{O}(n^3)$ nonsaturating push and $\mathrm{O}(n^2)$ relabeling operations. The algorithm by the first-active implementation is the first one that the intersection problem (or equivalently the maximum submodular flow problem) is solved in strongly polynomial time without using a lexicographical ordering.

The algorithm devised by P. Schönsleben [72] finds a maximum common subbase by $\mathrm{O}(n^3)$ augmentations on the lexicographically shortest augmenting paths in auxiliary networks and each augmenting path can be found in $\mathrm{O}(n^2)$ time by Dijkstra's shortest path algorithm with oracles for identifying arcs in auxiliary networks. A complexity improvement over P. Schönsleben's algorithm is made by É. Tardos, C. A. Tovey and M. A. Trick's algorithm [77] by reducing the total time in finding augmenting paths using the idea of layered network due to E. A. Dinits [12]. Their algorithm runs in $\mathrm{O}(n^4 h)$ time where h is the time to identify an arc in an auxiliary network and an arc's capacity.

Let $(\mathcal{D}_i, f_i)$ $(i = 1, 2)$ be two submodular systems on E with $|E| = n$, where $|E|$ is the cardinality of E. The *intersection problem* is given as follows (see [34]).

$$\text{(P)} \qquad \begin{array}{l} \text{Maximize } x(E) \\ \text{subject to } x \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2), \end{array}$$

where for each $i = 1, 2$ $\mathrm{P}(f_i)$ is the submodular polyhedron associated with $(\mathcal{D}_i, f_i)$.

Theorem 2.1.1 (The Intersection Theorem [34]): *For the intersection problem* (P),

$$\begin{aligned} &\max\{x(E) \mid x \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2)\} \\ &= \min\{f_1(X) + f_2(E - X) \mid X \in \mathcal{D}_1 \cap \overline{\mathcal{D}}_2\}, \end{aligned} \tag{2.1.1}$$

where $\overline{\mathcal{D}}_2 = \{E - X \mid X \in \mathcal{D}_2\}$.

The intersection theorem will play a very fundamental rôle in developing our algorithms. The proof is left in Section 2.2.

In our algorithms we assume oracles for saturation functions, saturation capacities, dependence functions and exchange capacities of the given submodular systems.

For each $i = 1, 2$ we denote the saturation function of $(\mathcal{D}_i, f_i)$ by sat_i, and similarly, the other functions and capacities associated with $(\mathcal{D}_i, f_i)$.

2.2. A New Algorithm by Path Augmentations

Initiate with a subbase pair $\beta = (y, z)$ such that

$$y \in \mathrm{B}(f_1), \quad z \in \mathrm{P}(f_2), \tag{2.2.2}$$

$$y \geq z. \tag{2.2.3}$$

(2.2.3) implies that z is a common subbase of $(\mathcal{D}_i, f_i)$ $(i = 1, 2)$. Also for any $x \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2)$, from (2.2.2) we have $y(E) = f_1(E) \geq x(E)$. Therefore,

$$y(E) \geq \max\{x(E) \mid x \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2)\}. \tag{2.2.4}$$

It follows that if we have $y = z$, then $z(= y)$ is a maximum common subbase.

We can apply the greedy algorithm described in Section 1.5 to find a base y of $(\mathcal{D}_1, f_1)$ and a base z' of $(\mathcal{D}_2, f_2)$ (see [34]). Putting $z(e) = \min\{y(e), z'(e)\}$ $(e \in E)$, the subbase pair (y, z) satisfies the above conditions (2.2.2) and (2.2.3). Finding a base of $(\mathcal{D}_i, f_i)$ for each $i = 1, 2$ requires n evaluations of function f_i.

Define an auxiliary network $\mathcal{N}_\beta = (G_\beta = (V, A_\beta), T_\beta^+, T_\beta^-, c_\beta)$ associated with $\beta = (y, z)$ as follows.

$$V = E, \quad A_\beta = A_\beta^1 \cup A_\beta^2, \tag{2.2.5}$$

where

$$A_\beta^1 = \{a' \mid \partial^+ a' = u,\ \partial^- a' = v,\ u, v \in V,\ u \in \mathrm{dep}_1(y, v) - \{v\}\}, \tag{2.2.6}$$

$$A_\beta^2 = \{a'' \mid \partial^+ a'' = u,\ \partial^- a'' = v,\ u, v \in V,\ v \in \mathrm{dep}_2(z, u) - \{u\}\}. \tag{2.2.7}$$

$c_\beta : A_\beta \to \mathrm{R}$ is the capacity function defined by

$$c_\beta(a) = \begin{cases} \tilde{c}_1(y, v, u) & (a = (u, v) \in A_\beta^1) \\ \tilde{c}_2(z, u, v) & (a = (u, v) \in A_\beta^2). \end{cases} \tag{2.2.8}$$

T^+_β (the entrance set) and T^-_β (the exit set) are defined by

$$T^+_\beta = \{v \mid v \in V,\ y(v) > z(v)\}, \tag{2.2.9}$$

$$T^-_\beta = \{v \mid v \in V,\ v \in E - \mathrm{sat}_2(z)\}. \tag{2.2.10}$$

A new algorithm for solving the intersection problem is described as follows.

An Algorithm by Path Augmentations

Input: $(\mathcal{D}_i, f_i)$ $(i = 1, 2)$ on E and an initial subbase pair $\beta = (y, z)$ satisfying (2.2.2) and (2.2.3).

Output: A solution z of (P).

Step 1: While $y \neq z$, do the following (a)$\sim$(c):

(a) Construct the auxiliary network $\mathcal{N}_\beta = (G_\beta = (V, A_\beta), T^+_\beta, T^-_\beta, c_\beta)$ associated with $\beta = (y, z)$. If there is no directed path from T^+_β to T^-_β, then stop (z is a solution of (P)).

(b) Let L be a directed path from T^+_β to T^-_β in G_β having the smallest number of arcs and put

$$\alpha = \min\{\min\{c_\beta(a) \mid a \text{ is an arc on } L\},\ y(\partial^+ L) - z(\partial^+ L),\ \hat{c}_2(z, \partial^- L)\} \tag{2.2.11}$$

($\partial^+ L$ is the initial vertex of L and $\partial^- L$ the terminal vertex of L).

(c) For each arc a in L,
if $a \in A^1_\beta$, then put

$$y(\partial^+ a) \leftarrow y(\partial^+ a) - \alpha, \quad y(\partial^- a) \leftarrow y(\partial^- a) + \alpha, \tag{2.2.12}$$

and if $a \in A^2_\beta$, then put

$$z(\partial^+ a) \leftarrow z(\partial^+ a) + \alpha, \quad z(\partial^- a) \leftarrow z(\partial^- a) - \alpha. \tag{2.2.13}$$

Also put $z(\partial^- L) \leftarrow z(\partial^- L) + \alpha$.

Step 2: The current $z (= y)$ is a solution of (P) and the algorithm terminates.
(End)

In the execution of (c), we assume that the arcs in A^1_β lying on path L are in this order $(u^1_1, v^1_1), \cdots, (u^1_p, v^1_p)$ along the direction of path L. Similarly the arcs in A^2_β lying on path L are assumed to have order as $(u_1^2, v_1^2), \cdots, (u_q^2, v_q^2)$. By the way of choosing path L, for each i, j such that $1 \leq i < j \leq p$ we have

$$u_i^1 \notin \mathrm{dep}_1(y, v_j^1) \tag{2.2.14}$$

and for each i, j such that $1 \leq i < j \leq q$ we have

$$v_j^2 \notin \mathrm{dep}_2(z, u_i^2). \tag{2.2.15}$$

After the execution of (c), the generated subbase pair is denote by (y_α, z_α). It can be expressed as

$$y_\alpha = y + \sum_{i=1}^{p} \alpha(\chi_{v_i^1} - \chi_{u_i^1}) \tag{2.2.16}$$

and

$$z_\alpha = z + \sum_{i=1}^{q} \alpha(\chi_{u_i^2} - \chi_{v_i^2}) + \alpha\chi_{v_p^1}. \tag{2.2.17}$$

By Lemma 1.4.11, we have $y_\alpha \in \mathrm{B}(f_1)$ and $z_\alpha \in \mathrm{P}(f_2)$. Obviously, $y_\alpha(E) = y(E)$ and $y_\alpha \geq z_\alpha$. Therefore, we have

Lemma 2.2.1: *If the algorithm terminates at (a) of Step 1, z is a solution of* (P).

Proof: Let U be the set of the vertices in G_β which are reachable by directed paths from T_β^+. Note that $U \neq E$. For each $v \in E - U$ and $u \in U$ we have

$$\mathrm{dep}_1(y, v) \subseteq E - U \subseteq \mathrm{sat}_1(y) = E, \tag{2.2.18}$$

$$\mathrm{dep}_2(z, u) \subseteq U \subseteq \mathrm{sat}_2(z). \tag{2.2.19}$$

Therefore, from (2.2.18) and the relation $v \in \mathrm{dep}_1(y, v)$, we have

$$E - U = \bigcup\nolimits_{v \in E-U} \{v\} \subseteq \bigcup\nolimits_{v \in E-U} \mathrm{dep}_1(y, v) \subseteq E - U. \tag{2.2.20}$$

Hence $E - U = \bigcup_{v \in E-U} \mathrm{dep}_1(y, v)$. According to Lemma 1.4.3 and Equation (1.4.11), we have $f_1(E - U) = y(E - U)$. For the same reason, $f_2(U) = z(U)$. Hence for any $x \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2)$,

$$z(E) = z(U) + z(E - U) = f_2(U) + y(E - U) = f_2(U) + f_1(E - U) \geq x(E). \tag{2.2.21}$$

We conclude that z is a solution of (P). □

Lemma 2.2.2: *$z(=y)$ in Step 2 is a solution of* (P). □

Proof of Theorem 2.1.1:

First we need to show the values of

$$\max\{x(E) \mid x \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2)\} \tag{2.2.22}$$

and

$$\min\{f_1(X) + f_2(E - X) \mid X \in \mathcal{D}_1 \cap \overline{\mathcal{D}}_2\} \tag{2.2.23}$$

in Theorem 2.1.1 are well defined. The second is the minimum on finitely many values, thus is well defined. Note that (2.2.22) is a linear programming problem and that for any $x \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2)$, we have $x(E) \leq f_1(E)$. So the optimal value, if any, is bounded above and it is easy to see that there exist feasible solutions. Applying the well known duality theorem on linear programming, we see that (2.2.22) always possesses an optimal solution. Let z be an optimal solution to it. In $\mathrm{P}(f_1)$ there exists a maximal subbase y with $y \geq z$. From Lemma 1.4.6, $y \in \mathrm{B}(f_1)$. This subbase pair (y, z) satisfies conditions (2.2.2) and (2.2.3). If we start with this pair in our algorithm, then the algorithm will soon stop at (a) of Step 1 when $y \neq z$ or stop at Step 2 when $y = z$. From the proof of Lemma 2.2.1 and condition (2.2.2), there exists $Y \in \mathcal{D}_1 \cap \overline{\mathcal{D}}_2$ such that $z(E) = f_1(Y) + f_2(E - Y)$. This implies

$$\min\{f_1(X) + f_2(E - X) \mid X \in \mathcal{D}_1 \cap \overline{\mathcal{D}}_2\} \leq \max\{x(E) \mid x \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2)\}. \tag{2.2.24}$$

The inverse inequality

$$\min\{f_1(X) + f_2(E - X) \mid X \in \mathcal{D}_1 \cap \overline{\mathcal{D}}_2\} \geq \max\{x(E) \mid x \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2)\}. \tag{2.2.25}$$

follows trivially from the inequality

$$x(E) = x(X) + x(E - X) \leq f_1(X) + f_2(E - X) \tag{2.2.26}$$

for any common subbase x and $X \in \mathcal{D}_1 \cap \overline{\mathcal{D}}_2$. □

If the rank functions f_1 and f_2 are integer-valued, then our algorithm stops in a finite number of steps provided that we start with an integral initial subbase pair.

Each time we carry out (c) of Step 1, the value of $z(E)$ increases by $\alpha > 0$ given by (2.2.11). By our assumption α should be integral and thus $\alpha \geq 1$.

Furthermore, we can refine our algorithm by using the technique [60, 72] of selecting the lexicographically shortest path in (b) of Step 1 so that the cycle (a)$\sim$(c) in Step 1 is repeated at most $\mathrm{O}(n^3)$ times.

2.3. A Faster Algorithm by the Preflow-Push Approach

For the intersection problem (P), we start with a subbase pair $\beta = (y, z)$ satisfying conditions (2.2.2) and (2.2.3) of Section 2.2. Define an auxiliary graph $\hat{G}_\beta = (\hat{V}, \hat{A}_\beta)$ with vertex set $\hat{V}$ and arc set $\hat{A}_\beta$ as follows:

$$\hat{V} = \{s^+, s^-\} \cup E, \tag{2.3.1}$$

$$\hat{A}_\beta = \hat{S}_\beta^+ \cup \hat{S}_\beta^- \cup \hat{A}_\beta^1 \cup \hat{A}_\beta^2, \tag{2.3.2}$$

where

$$\hat{S}_\beta^+ = \{(e, s^+) \mid e \in E\}, \tag{2.3.3}$$

$$\hat{S}_\beta^- = \{(e, s^-) \mid e \in E - \mathrm{sat}_2(z)\}, \tag{2.3.4}$$

$$\hat{A}_\beta^1 = \{a' \mid \partial^+ a' = u,\ \partial^- a' = v,\ u, v \in E,\ u \in \mathrm{dep}_1(y, v) - \{v\}\}, \tag{2.3.5}$$

$$\hat{A}_\beta^2 = \{a'' \mid \partial^+ a'' = u,\ \partial^- a'' = v,\ u, v \in E,\ v \in \mathrm{dep}_2(z, u) - \{u\}\}. \tag{2.3.6}$$

Also, define $e(v) = y(v) - z(v)$ for each $v \in E$, where note that $e(v) \geq 0$. When $e(v) > 0$, we call v an *active* vertex.

Definition 2.3.1 ([43]): A function d from $\hat{V}$ to nonnegative integers is said to be a *valid labeling* for $\hat{G}_\beta$ if $d(s^+) = n + 2$, $d(s^-) = 0$ and $d(\partial^+ a) \leq d(\partial^- a) + 1$ for every arc $a \in \hat{A}_\beta$.

Lemma 2.3.2: *For any valid labeling d, if $d(v) < n + 2$, then $d(v)$ is a lower bound of the actual distance from v to s^- in $\hat{G}_\beta$, where the length of each arc is equal to 1. If $d(v) \geq n + 2$, then $d(v) - (n + 2)$ is a lower bound of the actual distance from v to s^+ in $\hat{G}_\beta$ and s^- is not reachable from v in $\hat{G}_\beta$.*

Proof: If $d(v) < n+2$, let L be a shortest path from v to s^- in $\hat{G}_\beta$ having length l. We express L as a sequence $(v = v_1, v_2, \cdots, v_l, s^-)$ of the vertices in L. By the definition of a valid labeling d we have

$$d(v_1) \le d(v_2) + 1 \le d(v_3) + 2 \le \cdots \le d(s^-) + l = l, \tag{2.3.7}$$

i.e., $d(v) \le l$. The proof for the case $d(v) \ge n+2$ can be made similarly. □

We start with a valid labeling d such that $d(s^+) = n+2$, $d(s^-) = 0$ and $d(v) = 1$ $(v \in E)$. We define two basic operations Push(a) and Relabel(v) for each $a \in \hat{A}_\beta$ and $v \in \hat{V}$ for our algorithm:

Push(a): Applicability: $\partial^+ a$ is active, $a \in \hat{A}_\beta$ and $n+2 \ge d(\partial^+ a) = d(\partial^- a) + 1$;
Action:

Case 1: If $a \in \hat{A}^1_\beta$, then put $y \leftarrow y_\alpha = y + \alpha(\chi_w - \chi_v)$, where $v = \partial^+ a$, $w = \partial^- a$ and $\alpha = \min\{\tilde{c}_1(y, w, v), e(v)\}$.

Case 2: If $a \in \hat{A}^2_\beta$, then put $z \leftarrow z_\alpha = z + \alpha(\chi_v - \chi_w)$, where $v = \partial^+ a$, $w = \partial^- a$ and $\alpha = \min\{\tilde{c}_2(z, v, w), e(v)\}$.

Case 3: If $a \in \hat{S}^-_\beta$, then put $z \leftarrow z_\alpha = z + \alpha\chi_v$, where $v = \partial^+ a$ and $\alpha = \min\{\hat{c}_2(z, v), e(v)\}$.

It should be noted that if Push(a) is applicable for $a \in \hat{A}_\beta$, then $a \notin \hat{S}^+_\beta$.

Lemma 2.3.3: *Actions in all cases maintain the initial conditions* (2.2.2) and (2.2.3) *required for* (y, z).

Proof: In Case 1, because of the definition of α, we have $y \in \mathrm{B}(f_1)$ and $y \ge z$. Case 2 is similar to Case 1. In Case 3, y is not changed and z is still in $\mathrm{P}(f_2)$ because of the definition of α. □

Relabel(v): Applicability: v is active and for any $a \in \hat{A}_\beta$ with $\partial^+ a = v$ we have $d(\partial^+ a) \le d(\partial^- a)$;
Action: Put $d(v) \leftarrow \min\{d(w) + 1 \mid (v, w) \in \hat{A}_\beta\}$.

Lemma 2.3.4: *If* $e(v) > 0$ *and* $d(v) \le n+2$, *then either a push for some* $a \in \hat{A}_\beta$ *with* $\partial^+ a = v$ *or a relabel of* v *is applicable.*

Proof: Suppose $e(v) > 0$ and $d(v) \leq n + 2$. From the definition of the arc set $\hat{A}_\beta$, there exists an arc $a \in \hat{A}_\beta$ such that $\partial^+ a = v$. If a push operation is not applicable to v, then $d(\partial^+ a) < d(\partial^- a) + 1$ for any $a \in \hat{A}_\beta$ with $\partial^+ a = v$. This implies that a relabeling operation is applicable to v. Also note that such a push and a relabel are not simultaneously applicable. □

Lemma 2.3.5: *The basic operations keep d a valid labeling.*

Proof: Obviously, any relabeling operation does not change the values of $d(s^+)$ and $d(s^-)$ and keeps the relation $d(\partial^+ a) \leq d(\partial^- a) + 1$ for any $a \in \hat{A}_\beta$.

Next, we consider the push operations. In Case 1, suppose a push on $a \in \hat{A}^1_\beta$ with $\partial^+ a = u$ and $\partial^- a = v$ introduces new arc a' with $\partial^+ a' = w$ and $\partial^- a' = p$. Then, before the push we have (i) $u = p$ or there exists an arc in $\hat{A}^1_\beta$ from u to p and (ii) $v = w$ or there exists an arc in $\hat{A}^1_\beta$ from w to v (see Lemma 1.4.12). Consequently,

$$d(u) \leq d(p) + 1, \quad d(u) = d(v) + 1, \quad d(w) \leq d(v) + 1. \tag{2.3.8}$$

Hence,

$$d(w) \leq d(v) + 1 = d(u) \leq d(p) + 1. \tag{2.3.9}$$

Therefore, d is still valid in Case 1. The proof for Case 2 is similar to that for Case 1.

For a push on $a \in \hat{S}^-_\beta$ in Case 3, if $\alpha < \hat{c}_2(z, v)$, then $\hat{A}^2_\beta$ remains the same since for any $e \in \mathrm{sat}_2(z_\alpha)$ we have $\mathrm{dep}_2(z_\alpha, e) = \mathrm{dep}_2(z, e)$. If $\alpha = \hat{c}_2(z, v)$, then $v \in \mathrm{sat}_2(z_\alpha)$. For any $e \in \mathrm{sat}_2(z)$ we have $\mathrm{dep}_2(z, e) = \mathrm{dep}_2(z_\alpha, e)$, since $v \notin \mathrm{dep}_2(z, e)$; and for any $e \in \mathrm{sat}_2(z_\alpha) - \mathrm{sat}_2(z)$ there may be some new arc (e, e') (i.e., $e' \in \mathrm{dep}_2(z_\alpha, e)$), for which we get $d(e) < d(e') + 1$ since $d(e) \leq d(s^-) + 1 = 1$. □

Lemma 2.3.6: *For any $v \in E$ the distance label $d(v)$ never decreases by basic operations, and we have $d(v) \leq n + 3$.*

Proof: For any $v \in E$ we have $d(v) \leq d(s^+) + 1 = n + 3$, since $(v, s^+) \in \hat{A}_\beta$. □

Lemma 2.3.7: *Relabeling operations are carried out at most $n(n + 2)$ times.*

Proof: A relabeling operation makes the value of $\sum_{v \in E} d(v)$ increase by at least one and we have $n \leq \sum_{v \in E} d(v) \leq n(n + 3)$. The present lemma easily follows from this fact. □

Lemma 2.3.8: *For a subbase pair $\beta = (y, z)$ satisfying conditions* (2.2.2) and (2.2.3), *if there is no active vertex v in $\hat{G}_\beta$ with $d(v) \leq n+2$, then z is a solution of* (P).

Proof: If there is no active vertex, then we have $e(v) = 0$ for each $v \in E$, i.e., $y = z$, and from condition (2.2.2) $y(= z)$ is a solution of (P). If there is an active vertex, let $U \subseteq E$ be the set of the vertices in $\hat{G}_\beta$ which are reachable by directed paths from the active vertices. By the definition we have $U \neq \emptyset$. If $U = E$ and $E - \mathrm{sat}_2(z) \neq \emptyset$, then there is an active vertex v such that s^- is reachable from v. This contradicts the fact $d(v) = n+3$. Therefore, if $U = E$, we have $\mathrm{sat}_2(z) = E$, which implies that z is a solution of (P). If $U \neq E$, we have $y(E-U) = f_1(E-U)$ and $z(U) = f_2(U)$. Hence, $z(E) = z(U) + z(E-U) = f_2(U) + f_1(E-U)$. It follows from Theorem 2.1.1 that z is a solution of (P). □

Our algorithm repeats basic operations until there is no active vertex v with $d(v) \leq n+2$. In the following we make refinements on the order of the basic operations to be carried out.

Let $\pi : \hat{V} \rightarrow \{1, 2, \cdots, n+2\}$ be a one-to-one mapping, i.e., a numbering of the vertices in $\hat{V}$. For any $v \in E$ we have an arc list $List_\beta(v)$ formed by the out-going arc set $\{a \mid a \in \hat{A}_\beta,\ \partial^+ a = v\}$ arranged in the order of the increasing magnitude of the values of $\pi(\partial^- a)$. Each vertex v has a *current arc* a in the list. Initially, the current arc of v is the first element of $List_\beta(v)$.

In our implementation of the algorithm, we always select the active vertex v such that

$$d(v) = \max\{d(w) \mid w \in E,\ d(w) \leq n+2,\ e(w) > 0\} \tag{2.3.10}$$

and check if a push operation is applicable for the current arc a of $List_\beta(v)$. If the push operation is not applicable, then the next arc, if any, in $List_\beta(v)$ becomes the current arc. Otherwise, we perform a push operation, which results in (1) $e(v) = 0$ or (2) $e(v) > 0$ and $a \notin \hat{A}_\beta$. In the first case we select a new active vertex with the largest label and repeat the above process. In the second case, we let the next arc, if any, in the arc list of v be the current arc of v. During the algorithm, if we reach the end of the arc list of v with $e(v) > 0$ and want to renew the current arc in $List_\beta(v)$, then we let the first arc in the list be the current arc and carry out a relabeling operation for v.

Lemma 2.3.9: *Throughout the algorithm the following property (∗) is maintained:*
(∗) For each $v \in E$ any arc a before the current arc in $List_\beta(v)$ satisfies $d(\partial^+ a) \le d(\partial^- a)$.

Proof: Suppose that currently (∗) holds and that the next basic operation is a relabeling operation for a vertex v. This operation does not generate any new arc. Denote the current distance label by d and the one after the operation by d'. Note that $d(w) \le d'(w)$ $(w \in \hat{V})$. For $List_\beta(v)$ the current arc is made to be the first element of the list. Furthermore, for any other list $List_\beta(u)$ $(u \neq v)$ and any arc a before the current arc in $List_\beta(u)$ we have $d'(\partial^+ a) \le d'(\partial^- a)$ since $d(\partial^+ a) = d'(\partial^+ a)$ $(= d(u))$ and $d(\partial^- a) \le d'(\partial^- a)$. Hence, (∗) holds after the relabeling operation.

Next, suppose that currently (∗) holds and that the next basic operation is a push for the current arc a in $List_\beta(u)$ for a vertex u. Note that label d is not changed by the push. Therefore, it suffices to show that after the push operation any new arc $\hat{a}$ placed before the current arc in $List_\beta(w)$ for some $w \in E$ satisfies $d(\partial^+ \hat{a}) \le d(\partial^- \hat{a})$. Suppose, on the contrary, that some such new arc $\hat{a}$ satisfies

$$d(\partial^+ \hat{a})(= d(w)) = d(\partial^- \hat{a}) + 1. \tag{2.3.11}$$

We show that (2.3.11) leads us to a contradiction. We examine Cases 1∼3 for the push indicated below the definition of Push(a).

Case 1: Let $u = \partial^+ a$, $v = \partial^- a$ and $p = \partial^- \hat{a}$. Recall that $w = \partial^+ \hat{a}$. Before the push on arc a we have (i) $u = p$ or there exists an arc in $\hat{A}^1_\beta$ from u to p and (ii) $v = w$ or there exists an arc in $\hat{A}^1_\beta$ from w to v ([34]). Therefore,

$$d(w) \le d(v) + 1, \quad d(u) = d(v) + 1, \quad d(u) \le d(p) + 1, \quad d(w) = d(p) + 1. \tag{2.3.12}$$

Note that the last equation in (2.3.12) is (2.3.11). Since from (2.3.12)

$$d(w) = d(p) + 1 \ge d(u) = d(v) + 1 \ge d(w), \tag{2.3.13}$$

we have

$$d(u) = d(p) + 1, \quad d(w) = d(v) + 1. \tag{2.3.14}$$

It follows from (2.3.14) that
(i) $u \neq p$ and there exists an arc $a' \in \hat{A}^1_\beta$ from u to p for which a push is applicable,

(ii) $v \neq w$ and there exists an arc $a'' \in \hat{A}^1_\beta$ from w to v for which a push is applicable. From the assumption, (i) implies that arc a precedes arc a' in $List_\beta(u)$ and hence $\pi(p) > \pi(v)$, whereas (ii) implies that arc $\hat{a}$ precedes arc a'' in $List_\beta(w)$ and hence $\pi(p) < \pi(v)$, a contradiction.

Case 2: Case 2 leads us to a contradiction similar to Case 1.

Case 3: In the proof of Lemma 2.3.5 we have already shown that $d(\partial^+\hat{a}) \leq d(\partial^-\hat{a})$, which contradicts (2.3.11). □

Definition 2.3.10: A push on $(v, w) \in \hat{A}_\beta$ is called a *saturating push* if one of the following three conditions holds:

(a) The push is of Case 1 and $e(v) \geq \tilde{c}_1(y, w, v)$,

(b) The push is of Case 2 and $e(v) \geq \tilde{c}_2(z, v, w)$,

(c) The push is of Case 3 and $e(v) \geq \hat{c}_2(z, v)$.

Definition 2.3.11: If a push is not a saturating push, then it is called a *nonsaturating push*.

Lemma 2.3.12: *The number of saturating push operations is at most $2n^2(n + 2)$.*

Proof: By a saturating push on an arc in $List_\beta(v)$ the current arc shifts to the next arc. From Lemma 2.3.9, we know that between two successive relabeling operations on v, there are at most $2n$ saturating pushes on arcs going out from v. So the total number of saturating pushes on arcs going out from v is at most $2n(n+2)$. This proves the lemma. □

Lemma 2.3.13: *The number of nonsaturating pushes is at most $n^2(n + 2)$.*

Proof: Between two successive nonsaturating pushes on arcs going out from v there is at least one relabeling operation on some other vertex, since vertex v becomes inactive just after the first nonsaturating push and turns active again before the second nonsaturating push. This is impossible if no relabeling happens between the two nonsaturating pushes, since v had the largest label among all active vertices at the first nonsaturating push. Hence, the number of nonsaturating pushes on arcs going out from v is at most $n(n+2)$. Therefore, the total number of nonsaturating pushes required by the algorithm are at most $n^2(n+2)$. □

Now, we have

Theorem 2.3.14: *The new algorithm terminates after carrying out* $\mathrm{O}(n^2)$ *relabeling operations and* $\mathrm{O}(n^3)$ *push operations.*

Proof: Directly from Lemmas 2.3.7, 2.3.12 and 2.3.13. □

Finally, we describe an implementation of our preflow algorithm without using the specific order on the arc list and the current arcs.

Define a list L_d of all vertices in V with $d(v) \le n+2$. The vertices of V in L_d is arranged in the order of the decreasing magnitude of the values of d. For the vertices having the same value of d, they can be ordered in any way. In list L_d there is a vertex called the current vertex of L_d. Initially the current vertex is the first vertex of L_d. Since $d(v) > d(w)$ holds for each admissible arc in $\hat{A}_\beta$, we have an important property of list L_d, i.e., if (v,w) in $\hat{A}_\beta$ is an admissible arc, then v appears before w in L_d.
Preflow Algorithm (first-active implementation)
Step 1: Let L_d be the list of V and v be the current vertex. If v is an active vertex, go to Step 2. If v is not the last vertex of L_d, replace v as the current vertex by the vertex right after v on L_d and go to the beginning of Step 1. If v is the last vertex of L_d, then output the current β and stop.
Step 2: Find an admissible arc (v,w), perform push on (v,w) and go to Step 1. If there is no admissible arc going out from v, then relabel v. For the new $d(v)$, if $d(v) > n+2$ then delete v from L_d. If L_d is empty then output the current β and stop, otherwise let the first vertex of L_d be the current vertex and go to Step 1.
(End)

Define a *pass* over L_d as a period of the algorithm that begins with the first vertex of L_d and ends when a relabeling is performed or when the algorithm terminates. Since the distance label d is not changed and the excess of each active vertex is pushed to vertices after it, the vertices before the current vertex are inactive. It follows that the output β gives a required maximum subbase.

Lemma 2.3.15: *There are at most $n(n+2)$ passes before the algorithm terminates.*

Proof: Since there is a relabeling operation in each pass except the last one, the present lemma follows from Theorem 2.3.7. □

Lemma 2.3.16: *The number of nonsaturating pushes in the algorithm (first-active implementation) is at most $n^2(n+2)$.*

Proof: In each pass, a nonsaturating push on v makes v an inactive vertex and the current vertex shifts. So, there are at most n nonsaturating pushes in each pass. □

Lemma 2.3.17: *The number of saturating push operations in the algorithm (first-active implementation) is at most $2n^3(n+2)$.*

Proof: In a fixed pass, after a saturating push on (v,w), (v,w) is not an arc in $\hat{A}_\beta$ until v becomes inactive (see Lemma 1.4.12). In the pass, there is no push on (v,w) again after v becomes inactive. Hence, in a pass there are at most $2n$ saturating pushes on arcs going out from v. It follows that there are at most $2n^2$ saturating pushes in a pass. The present lemma is from Lemma 2.3.15. □

Example: Consider an example ([51]) of an electric network with the underlying graph $G = (V, A^*)$ shown in Figure 1, where each dotted line with an arrow denotes the existence of self- or mutual-coupling in the direction of the arrow. Under some generality assumption on the coupling parameters the network is uniquely solvable if and only if the optimal value of the objective function $\varphi : \hat{A} \to \mathrm{R}$ of the following problem (2.3.15) is equal to the rank $r_G(A^*)$ of G, where r_G is the rank function of graph G ([82]). Consider a bipartite graph $\hat{G} = (\hat{V}^+, \hat{V}^-; \hat{A})$ shown in Figure 2, where each arc $a \in \hat{A}$ of $\hat{G}$ denotes the existence of self- or mutual-coupling from branch $\partial^- a \in A^*$ to branch $\partial^+ a \in A^*$.

Now, the problem is:

$$\begin{aligned} &\text{Maximize } \sum_{a\in\hat{A}} \varphi(a) \\ &\text{subject to } \partial^+\varphi \in \mathrm{P}(r_G), \\ &\qquad\qquad\ \partial^-\varphi \in \mathrm{P}(r_G), \end{aligned} \tag{2.3.15}$$

where

$$\partial^+\varphi(v) = \sum_{a\in\delta^+ v} \varphi(a), \tag{2.3.16}$$

$$\partial^-\varphi(v) = \sum_{a\in\delta^- v} \varphi(a) \tag{2.3.17}$$

and $\delta^+ v$ ($\delta^- v$) is the set of the arcs going out from v (coming into v) in the bipartite graph shown in Figure 2. Problem (2.3.15) can be reduced to an intersection problem as follows. Put $E = \hat{A}$ and define submodular functions f_i ($i = 1, 2$) on 2^E as follows.

$$f_1(X) = r_G(\partial^+ X) \quad (X \subseteq E(=\hat{A})), \tag{2.3.18}$$

$$f_2(X) = r_G(\partial^- X) \quad (X \subseteq E(=\hat{A})), \tag{2.3.19}$$

where $\partial^+ X$ ($\partial^- X$) is the set of the initial (terminal) vertices of the arcs in X in the bipartite graph $\hat{G}$. Then, Problem (2.3.15) is equivalent to the following.

$$\begin{aligned} P: \quad &\text{Maximize } x(E) \\ &\text{subject to } x \in \mathrm{P}(f_1) \cap \mathrm{P}(f_2). \end{aligned} \tag{2.3.20}$$

We show how our algorithm works for this example. We start from $y \in \mathrm{B}(f_1)$ and $z \in \mathrm{P}(f_2)$ given by

$$y = \chi_{(1,1)} + \chi_{(2,1)} + \chi_{(4,1)}, \qquad z = \mathbf{0}. \tag{2.3.21}$$

Note that $\{1, 2, 4\}$ is a spanning tree of G and y is a base of $(2^E, f_1)$. The initial auxiliary graph is shown in Figure 3. For convenience, we denote y_T, z_H for any T, $H \subseteq E$ by

$$y_T = \sum_{e\in T} \chi_e, \qquad z_H = \sum_{e\in H} \chi_e. \tag{2.3.22}$$

In order to avoid confusions between a vertex set and its vertex numbering, instead of the numbering function π we introduce a linear ordering on $E \cup \{s^+, s^-\}$ as follows.

$$\begin{aligned} &s^- < (1,1) < (2,1) < (2,3) < (3,1) < (4,1) < \\ &(5,3) < (6,2) < (6,4) < (6,5) < (6,6) < s^+. \end{aligned} \tag{2.3.23}$$

At the beginning, we put $d(s^+) = 12$, $d(s^-) = 0$ and $d(v) = 1$ for any $v \in E$. Initially we take $\beta = (y_T, z_H)$ with $T = \{(1,1),(2,1),(4,1)\}$ and $H = \emptyset$. After a selection of $v \in E$ such that $e(v) > 0$ and $d(v) = \max\{d(u) \mid e(u) > 0,\ u \in E,\ d(u) \leq 12\}$, we have two types of actions (A) and (B) as follows:

(A) Finding the first arc in $List_\beta(v)$ which is applicable for push. This can be done by two steps. The first step is to find out the set

$$A(v) = \{u \mid u \in E \cup \{s^+, s^-\},\ (v,u) \in \hat{A}_\beta,\ d(u) = d(v) - 1\}.$$

The second is to choose, if $A(v) \neq \emptyset$, an arc (v,u) such that u is the minimum element in $A(v)$ with respect to the linear ordering (2.3.23). Perform a push operation on the selected arc and revise T and H after the push.

(B) If $A(v) = \emptyset$, then relabel v.

Here, we note that the values of the saturation capacities and the exchange capacities are always 0 or 1 since we start with a 0-1 valued subbase pair. By an arc with (or without) an underline, we mean that the arc is in $\hat{A}^2_\beta$ (or not in $\hat{A}^2_\beta$). After 11 push and 7 relabeling operations we reach an optimal solution of Problem P in (2.3.20). The algorithm is performed as follows:

(A) Push on ((1,1),s^-), and we have $T = \{(1,1),(2,1),(4,1)\}$ and $H = \{(1,1)\}$.

(B) Relabel (2,1) we have d((2,1))=2.

(A) Push on $\underline{((2,1),(1,1))}$, and we have $T = \{(1,1),(2,1),(4,1)\}$ and $H = \{(2,1)\}$.

(B) Relabel (1,1) we have d((1,1))=2.

(A) Push on ((1,1),(3,1)), and we have $T = \{(2,1),(3,1),(4,1)\}$ and $H = \{(2,1)\}$.

(B) Relabel (3,1) we have d((3,1))=2.

(A) Push on ((3,1),(5,3)), and we have $T = \{(2,1),(4,1),(5,3)\}$ and $H = \{(2,1)\}$.

(B) Relabel (4,1) we have d((4,1))=3.

(A) Push on ((4,1),(1,1)), and we have $T = \{(1,1),(2,1),(5,3)\}$ and $H = \{(2,1)\}$.

(B) Relabel (1,1) we have d((1,1))=3

(A) Push on ((1,1),(2,1)), and we have $T = \{(1,1),(2,1),(5,3)\}$ and $H = \{(1,1)\}$.

(A) Push on ((2,1),(6,2)), and we have $T = \{(1,1),(5,3),(6,2)\}$ and $H = \{(1,1)\}$.

(A) Push on ((5,3),s^-), and we have $T = \{(1,1),(5,3),(6,2)\}$ and $H = \{(1,1),(5,3)\}$.

(B) Relabel (6,2) we have d((6,2))=2.

(A) Push on ((6,2),(2,3)), and we have $T = \{(1,1),(2,3),(5,3)\}$ and $H = \{(1,1),(5,3)\}$.

(B) Relabel (2,3) we have d((2,3))=2.

(A) Push on ((2,3),(6,4)), and we have $T = \{(1,1),(5,3),(6,4)\}$ and $H = \{(1,1),(5,3)\}$.

(A) Push on ((6,4),s^-), and we have $T = H = \{(1,1),(5,3),(6,4)\}$.

There is no vertex with $e(v) > 0$ and the algorithm terminates.

For Problem (2.3.15), put $\varphi = z_H$ (the output of the algorithm). Then φ is an optimal solution of Problem (2.3.15) with the optimal value being equal to the rank of graph G. Hence we conclude that the given electric network is uniquely solvable.

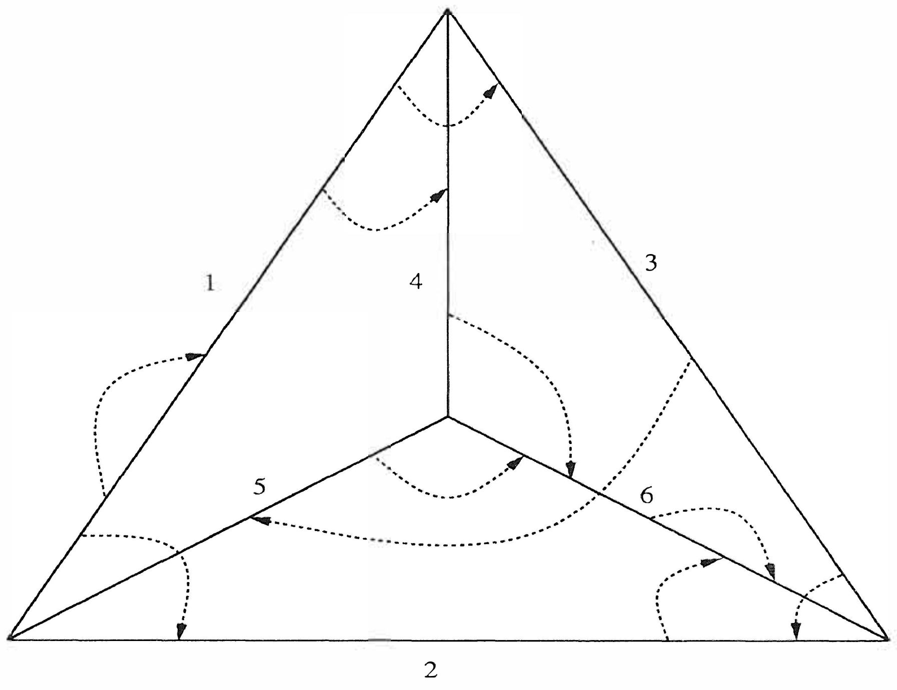

Figure 1: A graph G representing an electric network with mutual couplings

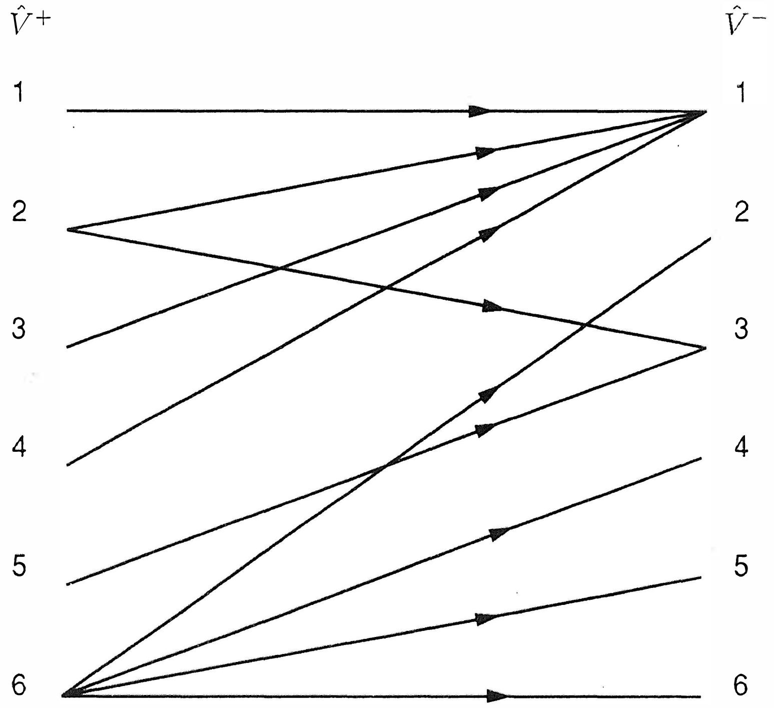

Figure 2: The bipartite graph $\hat{G}$

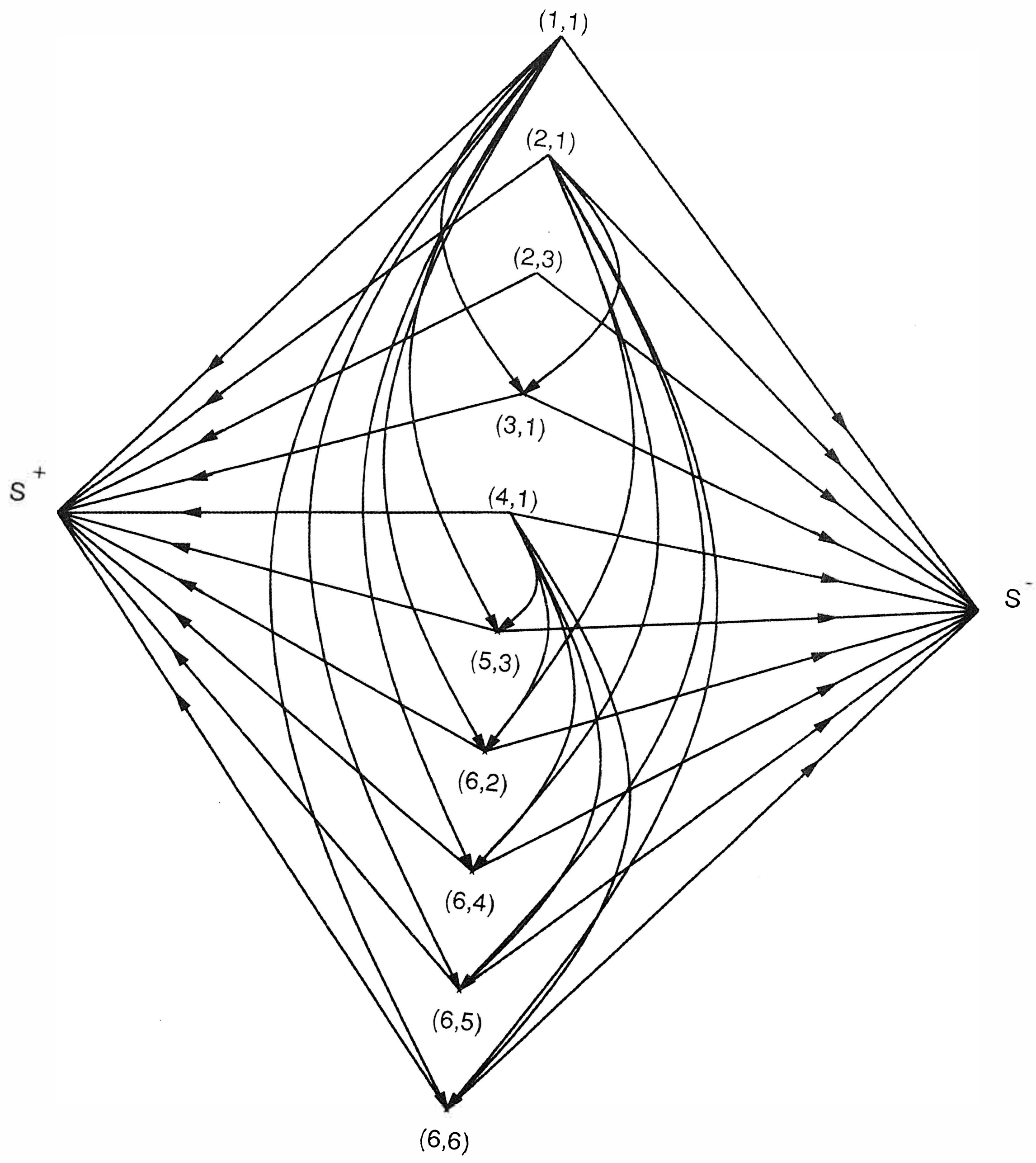

Figure 3: The initial auxiliary graph

Chapter 3.

An Efficient Cost Scaling Algorithm for the Independent Assignment Problem

3.1. Introduction

The independent assignment problem was formulated and solved by M. Iri and N. Tomizawa [53]. Given a bipartite graph with matroidal structures on both of the two sets of end-vertices, the independent assignment problem is to find a maximum independent matching [81] having the smallest total cost, where a cost is given to each arc. It is a natural extension of the ordinary assignment problem. The weighted matroid intersection problem considered by J. Edmonds [13], E. L. Lawler [58] and others is equivalent to the independent assignment problem.

The theoretical analyses and algorithms for the independent assignment problem can be found in [53] and [27]. Algorithms for the weighted matroid intersection problem were given by J. Edmonds [13], [15], E. L. Lawler [58], A. Frank [21], J. B. Orlin and J. Vande Vate [67], C. Brezovec, G. Cornuéjols and F. Glover [6], H. N. Gabow and Y. Xu [40] and others. Recently, M. Shigeno and S. Iwata [74] have proposed an approximate-weight-splitting algorithm for the weighted matroid intersection problem which is an approximate version of A. Frank's algorithm [21]. The cost scaling approach based on the approximate optimality plays a fundamental rôle in recent efficient algorithms for ordinary minimum-cost flows and bipartite matchings.

We propose an efficient cost scaling algorithm for the independent assignment problem. Our algorithm in general can be viewed as a generalization of the cost scaling algorithm, recently given by J. B. Orlin and R. K. Ahuja [66], for the ordinary assignment problem. The cost scaling technique is adopted in our algorithm. The procedure for each scaling phase can be decomposed into two parts: an auction-like algorithm (see [4], [66]) and a successive shortest path algorithm. We provide a complexity analysis under the independence oracle for matroids and we show that on a bipartite graph with n vertices and integer arc costs bounded by C, an optimal r-independent assignment can be found in $O(\sqrt{r}n^2 \log(rC))$ time by our algorithm under an independence oracle for matroids.

In Sections 3.2, 3.3 and 3.4 we give the definition of the independent assignment problem, some applications of the independent assignment problem and some further properties of matroids, respectively. The optimality and ε-optimality conditions for the problem are given in Section 3.5. Sections 3.6 and 3.7 describe the detail of our algorithm. Section 3.8, is concerned with the complexity analysis of the algorithm.

3.2. The Independent Assignment Problem

Let $G = (V^+, V^-; A)$ be a *bipartite graph* with the left (right) end-vertex set V^+ (V^-) and the arc set A. For any $a \in A$, $\partial^+ a$ ($\partial^- a$) is the initial (terminal) end-vertex of a. We assume that $\partial^+ a \in V^+$ and $\partial^- a \in V^-$ for each $a \in A$. Also for any $M \subseteq A$, $\partial^+ M$ ($\partial^- M$) denotes the set of the initial (terminal) end-vertices of arcs in M. A subset M of A is called a *matching* in the bipartite graph $G = (V^+, V^-; A)$ if $|\partial^+ M| = |M| = |\partial^- M|$.

Let $\mathcal{M}^+ = (V^+, \mathcal{I}^+)$ and $\mathcal{M}^- = (V^-, \mathcal{I}^-)$, respectively, be matroids on V^+ and V^- with families $\mathcal{I}^+ \subseteq 2^{V^+}$ and $\mathcal{I}^- \subseteq 2^{V^-}$ of independent sets. A cost function $c : A \rightarrow \mathrm{Z}$ is given, where Z is the set of all integers. We denote this network by $\mathcal{N} = (G = (V^+, V^-; A), \mathcal{M}^+, \mathcal{M}^-, c)$.

An *independent matching* $M \subseteq A$ in $\mathcal{N}$ is a matching in G such that $\partial^+ M \in \mathcal{I}^+$ and $\partial^- M \in \mathcal{I}^-$. The maximum independent matching problem is to find an independent matching M in $\mathcal{N}$ of the maximum cardinality.

For a positive integer k, a *k-independent matching* M in $\mathcal{N}$ is an independent matching of cardinality k. An *optimal k-independent assignment* in $\mathcal{N}$ is a k-independent matching M having the minimum cost $c(M) = \sum_{e \in M} c(e)$ among all the k-independent matchings in $\mathcal{N}$. When costs are taken into account, we use the term, assignment, instead of matching (in a bipartite graph).

Let B^+ (B^-) be any base of $\mathcal{M}^+$ ($\mathcal{M}^-$), where we assume that $|B^+| = |B^-| = r$.

In this chapter we will give a new efficient scaling algorithm for finding an optimal r-independent assignment if there exists one. The general optimal k-independent assignment problem can be solved by our algorithm after applying *k-truncation* (see [82]) of $\mathcal{M}^+$ and $\mathcal{M}^-$. Throughout this chapter, we denote the closure function of $\mathcal{M}^+$ ($\mathcal{M}^-$) by cl^+ (cl^-), the fundamental circuit with respect to I and e in $\mathcal{M}^+$ ($\mathcal{M}^-$) by $\mathrm{C}^+(I|e)$ ($\mathrm{C}^-(I|e)$) and the fundamental cocircuit with respect to B and e in $\mathcal{M}^+$ ($\mathcal{M}^-$) by $\mathrm{K}^+(B|e)$ ($\mathrm{K}^-(B|e)$). For convenience we assume that any single element subset is independent, i.e., there exist no self-loops in $\mathcal{M}$.

3.3. Applications of the Independent Assignment Problem

Minimum-cost spanning arborescence. Given a directed graph $\hat{G} = (V, \hat{A})$ with a cost function $c : \hat{A} \to \mathbf{R}$. A *spanning arborescence* is a spanning tree T of $\hat{G}$ satisfying that there exists a vertex $v \in V$ and for each other vertex u there exists a directed path from v to u consisting of only arcs in T. Let the cost of a tree be the sum of all costs of its arcs. The *minimum-cost spanning arborescence problem* is the problem to find a spanning arborescence of the minimum cost among all spanning arborescences. The problem has applications in optimal water supply networks and others. We can formulate this problem into an independent assignment problem. Let $G = (V^+, V^-; A, c')$ with $V^+ = \hat{A}$, $V^- = V$ and

$$A = \{(a, \partial^-_{\hat{G}} a) \mid a \in \hat{A}\}, \tag{3.3.1}$$

with $c'(a, \partial^-_{\hat{G}} a) = c(a)$. The matroid on V^+ is the graphic matroid of $\hat{G}$ and the matroid on V^- is a free matroid, i.e., any subset of V^- is defined to be an independent set. We have that a $(|V| - 1)$-independent assignment is a spanning arborescence of $\hat{G}$. Hence the minimum-cost spanning arborescence problem is reduced to the optimal $(|V| - 1)$-independent assignment problem.

Controllability/observability of a linear dynamical system with combinatorial constraints. Consider a discrete-time linear dynamical system represented by a system of difference equations:

$$x(t+1) = Ax(t) + Bu(t)$$

$$y(t) \quad = Cx(t) + Du(t) \tag{3.3.2}$$

for $t = 0, 1, \cdots, n-1$, where $x(t)$, $u(t)$ and $y(t)$ are the n-dimensional state vector, the r-dimensional control vector and the p-dimensional observation vector, respectively, at time t. A, B, C and D are constant matrices of appropriate sizes.

As is well known, the necessary and sufficient condition for System (3.3.2) to be controllable is

$$\mathrm{rank}[B, AB, \cdots, A^{n-1}B] = n \tag{3.3.3}$$

and that for System (3.3.2) to be observable is

$$\mathrm{rank}[C', A'C', \cdots, A'^{n-1}C'] = n \tag{3.3.4}$$

where the prime denotes the transposition of a matrix.

However, due to physical constraints or the need to decrease the cost, the following restrictions are frequently happens in practice.
(i) every control terminal can not be used more than a prescribed number of times,
(ii) at every time, at most a prescribed number of control terminals can be used,
(i') every observation terminal can not be used more than a prescribed number of times,
(ii') at every time, at most a prescribed number of observation terminals can be used.

Let $G = (V^+, V^-; A)$ be a bipartite graph with V^+ the set of all columns of the wide matrix in (3.3.4),

$$V^- = \{y(t)_i \mid t = 0, 1, \cdots, n-1, \ i = 1, 2, \cdots, p\} \tag{3.3.5}$$

and the arc set A is a corresponding from $y(t)_i$ to the ith column of the matrix A'^tC'. The matroid on V^+ is defined to be the linear matroid. Suppose, in constraint (ii'), the prescribed number for each period t is r_t $(t = 0, 1, \cdots, n-1)$. The matroid on V^- is defined to be the partition matroid with partition of V^- as $V_t^- = \{y(t)_i \mid i = 1, 2, \cdots, p\}$ $(t = 0, 1, \cdots, n-1)$. Consequently, the problem of determining whether System (3.3.2) is observable under constraints (ii') is reduced to the problem that whether there exists an n-independent assignment for G. Moreover, we can consider the problem of selecting the observation variables of the minimum cost under constraint (ii'). The problem is reduced to the optimal n-independent assignment problem. The problems for other types of constraints can be treated in a similar way.

The applications to the structural solvability and controllability of large-scale systems is extensively studied and presented by K. Murota in [64]. The applications in electric network theory can be found in A. Recski [70] and M. Iri [50].

3.4. Further Properties of Matroids

In this section we give some further properties of matroids which play a fundamental rôle in our algorithm. Some of them are adopted from M. Iri and N. Tomizawa [53]. We provide no proofs for those well known (see, e.g., [52] and [82]).

Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid with a family $\mathcal{I}$ of independent sets.

Lemma 3.4.1: *If $I \in \mathcal{I}$ and $v \in \mathrm{cl}(I) - I$, then for each $u \in \mathrm{C}(I|v)$, $I + v - u$ is an independent set.* □

Lemma 3.4.2: *For a given $I \in \mathcal{I}$, if $2q$ distinct elements $u_1, \cdots, u_q$ $(\in I)$ and $v_1, \cdots, v_q$ $(\in \mathrm{cl}(I) - I)$ satisfy the relations:*

$$u_j \in \mathrm{C}(I|v_j) \quad (j = 1, \cdots, q) \tag{3.4.6}$$

and

$$u_i \notin \mathrm{C}(I|v_j) \quad (1 \leq i < j \leq q), \tag{3.4.7}$$

then for each $m = 1, \cdots, q$ $I_m = (I - \{u_1, \cdots, u_m\}) \cup \{v_1, \cdots, v_m\}$ is also an independent set and $\mathrm{cl}(I_m) = \mathrm{cl}(I)$. Furthermore, for $m = 2, \cdots, q$, we have $u_m \in \mathrm{C}(I_{m-1}|v_m)$. □

Lemma 3.4.3: *For $I \in \mathcal{I}$ and a pair (u, v) such that $v \in \mathrm{cl}(I) - I$, $u \in \mathrm{C}(I|v) - v$, denote the independent set $I + v - u$ by I'. If (w, z) is a pair such that $z \in \mathrm{cl}(I') - I'$, $w \notin \mathrm{C}(I|z)$ and $w \in \mathrm{C}(I'|z) - z$ then we have*

(i) *either $u = z$ or $u \in \mathrm{C}(I|z) - z$,*

(ii) *either $v = w$ or $w \in \mathrm{C}(I|v) - v$.* □

For a family $\mathcal{S} = \{S_1, \cdots, S_t\}$ of subsets of E, a *transversal* of $\mathcal{S}$ is a set $\{e_1, \cdots, e_t\}$ of t distinct elements of E such that $e_i \in S_i$ for $i = 1, \cdots, t$.

Lemma 3.4.4: *Let B be a base of $\mathcal{M}$ and $v_1, \cdots, v_t$ be t elements of B, where $t \leq |B|$. Suppose that there are t circuits $C_1, \cdots, C_t$ of $\mathcal{M}$ such that (1) $v_k \in C_k$ for $k = 1, \cdots, t$ and (2) $v_k \notin C_l$ if $k \neq l$, for $k, l = 1, \cdots, t$. Denote $T_k = \mathrm{K}(B|v_k) - v_k$ for $k = 1, \cdots, t$. Then, there exists a common transversal of the families $\mathcal{C} = \{C_1, \cdots, C_t\}$ and $\mathcal{T} = \{T_1, \cdots, T_t\}$.*

Proof: It is known [8, p. 74] that there is a common transversal for $\mathcal{C}$ and $\mathcal{T}$ if and only if for each $X,\ Y \subseteq \{1, \cdots, t\}$ we have

$$|(\bigcup_{k \in X} C_k) \cap (\bigcup_{l \in Y} T_l)| \geq |X| + |Y| - t. \tag{3.4.8}$$

Since $|X \cap Y| \geq |X| + |Y| - t$, from (3.4.8) it is sufficient for us to verify that

$$|(\bigcup_{k \in X \cap Y} C_k) \cap (\bigcup_{l \in X \cap Y} T_l)| \geq |X \cap Y|. \tag{3.4.9}$$

Without loss of generality we prove (3.4.9) when $X \cap Y = \{1, \cdots, t\}$. That is, we prove that

$$|(\bigcup_{k=1}^{t} C_k) \cap (\bigcup_{k=1}^{t} T_k)| \geq t. \tag{3.4.10}$$

Denote $I_0 = \{v_1, \cdots, v_t\}$ and $I_1 = B - I_0$. Since $v_k \in \mathrm{cl}(C_k - v_k)$ for $k = 1, \cdots, t$, we have

$$B \subseteq \mathrm{cl}((\bigcup_{k=1}^{t} (C_k - v_k)) \cup I_1). \tag{3.4.11}$$

Hence, there exist $u_1, \cdots, u_t \in \cup_{k=1}^{t} (C_k - v_k)$ such that $I_1 + u_1 + \cdots + u_t$ is a base of $\mathcal{M}$. From the assumptions on $\mathcal{C}$ and $\{v_1, \cdots, v_t\}$ we have $\{u_1, \cdots, u_t\} \cap I_0 = \emptyset$. For each $1 \leq l \leq t$, since $I_1 + u_l \in \mathcal{I}$, we have $I_0 \supseteq \mathrm{C}(B|u_l) - (I_1 + u_l) \neq \emptyset$. Hence, there is some element $v_s \in I_0 \cap \mathrm{C}(B|u_l)$. This implies that $B + u_l - v_s$ is also a base, i.e., $u_l \notin \mathrm{cl}(B - v_s)$. Thus $u_l \in T_s$. We have $u_1, \cdots, u_t \in \cup_{k=1}^{t} T_k$ and hence (3.4.10) is valid. □

3.5. The Exact and Approximate Optimality

In this section we define an auxiliary network and give optimality and ε-optimality conditions for the independent assignment problem in terms of auxiliary networks. The concept of ε-optimality was introduced by D. P. Bertsekas [4] and É. Tardos [76] for the minimum-cost flow problem and is essential in our cost scaling framework.

Given a network $\mathcal{N} = (G = (V^+, V^-; A), \mathcal{M}^+, \mathcal{M}^-, c)$ as in Section 3.2, let $\mathcal{B}^+$ ($\mathcal{B}^-$) be the family of bases of $\mathcal{M}^+$ ($\mathcal{M}^-$). Consider a triple $\Delta = (B^+, M, B^-)$ that satisfies the following conditions:

$$M \text{ is a matching of } G = (V^+, V^-; A), \tag{3.5.12}$$

$$B^+ \in \mathcal{B}^+ \text{ and } B^- \in \mathcal{B}^-, \tag{3.5.13}$$

$$\partial^+ M \subseteq B^+ \text{ and } \partial^- M \subseteq B^-. \tag{3.5.14}$$

We define the *auxiliary network* $\mathcal{N}_\Delta$ associated with $\Delta = (B^+, M, B^-)$ as $\mathcal{N}_\Delta = (G_\Delta = (V^*, A_\Delta), \mathcal{M}^+, \mathcal{M}^-, c_\Delta)$ with vertex set $V^* = V^+ \cup V^-$ and arc set $A_\Delta = A_{B^+} \cup A \cup \tilde{M} \cup A_{B^-}$, where

$$A_{B^+} = \{(u, v) \mid v \in V^+ - B^+,\ u \in \mathrm{C}^+(B^+|v) - v\}, \tag{3.5.15}$$

$$A_{B^-} = \{(v, u) \mid v \in V^- - B^-,\ u \in \mathrm{C}^-(B^-|v) - v\}, \tag{3.5.16}$$

$$\tilde{M} = \{\bar{a} \mid a \in M\} \quad (\bar{a} : \text{a reorientation of } a) \tag{3.5.17}$$

and $c_\Delta : A_\Delta \to \mathrm{Z}$ is defined from $c : A \to \mathrm{Z}$ as

$$c_\Delta(a) = \begin{cases} c(a) & \text{if} \quad a \in A \\ 0 & \text{if} \quad a \in A_{B^+} \cup A_{B^-} \\ -c(\bar{a}) & \text{if} \quad a \in \tilde{M}. \end{cases} \tag{3.5.18}$$

A triple $\Delta = (B^+, M, B^-)$ that satisfies conditions (3.5.12)~(3.5.14) is called an *independent partial assignment* and if, in addition, $|M| = r$, then $\Delta = (B^+, M, B^-)$ is called an *independent assignment.*

In the auxiliary network $\mathcal{N}_\Delta$, we consider $c_\Delta(a)$ as the length of arc $a \in A_\Delta$. Then, we have the following.

Theorem 3.5.1 (Fujishige [27]): *An independent assignment $\Delta = (B^+, M, B^-)$ is an optimal independent assignment of $\mathcal{N}$ if and only if there is no negative directed cycle in $\mathcal{N}_\Delta$.* □

Given a function $p : V^* \to \mathrm{R}$, called a *potential*, we define $c_{\Delta,p}(a) = c_\Delta(a) + p(\partial^+ a) - p(\partial^- a)$ for each $a \in A_\Delta$. From Theorem 3.5.1 we also have

Theorem 3.5.2: *An independent assignment $\Delta = (B^+, M, B^-)$ is an optimal independent assignment of $\mathcal{N}$ if and only if there exists a potential p such that $c_{\Delta,p}(a) \geq 0$ for all $a \in A_\Delta$.* □

Definition 3.5.3: An independent partial assignment $\Delta = (B^+, M, B^-)$ is said to be *ε-optimal* if there exists a potential p such that $c_{\Delta,p}(a) \geq -\varepsilon$ for all $a \in A_\Delta$. □

Put $C = \max_{a \in A} |c(a)|$. Then, we have

Lemma 3.5.4: *Any independent assignment is ε-optimal for $\varepsilon \geq C$ and any ε-optimal independent assignment with $\varepsilon < 1/|4r|$ is an optimal independent assignment.*

Proof: The first part of the lemma can be verified by taking $p \equiv 0$. For the second part of the lemma, we see that if $\varepsilon < 1/|4r|$, then there is no negative directed cycle in $\mathcal{N}_\Delta$, since the length $\sum_{a \in C} c_\Delta(a) = \sum_{a \in C} c_{\Delta,p}(a)$ of each cycle C is an integer and is greater than or equal to $-\varepsilon|4r| > -1$. Hence, the optimality of the independent assignment follows from Theorem 3.5.1. □

3.6. A Cost Scaling Framework

We first give a higher-level description of our cost-scaling algorithm for finding an optimal independent assignment. Starting from $\varepsilon = C$, the algorithm proceeds by obtaining ε-optimal independent assignments for successively smaller values of ε until the value of ε is less than $1/|4r|$. Thus, at the end we have an optimal independent assignment due to Lemma 3.5.4. Therefore, the algorithm consists of a number of cost-scaling phases. In each cost-scaling phase, the algorithm performs procedure Refine which transforms a 2ε-optimal independent assignment to an ε-optimal independent assignment.

Procedure Refine consists of two subprocedures: Auction and SuccessiveShortest-Path. Procedure Auction can be viewed as a generalization of the auction procedure given by J. B. Orlin and R. K. Ahuja [66] which is designed for the optimal assignment problem on a bipartite graph without any additional matroid constraints. The auction procedure by Orlin and Ahuja [66] is a variation of the auction algorithm by D. P. Bertsekas and J. Eckstein [5]. Our procedure Auction starts with an ε-optimal independent assignment and first converts it into an $\varepsilon/4$-optimal independent partial assignment $\Delta = (B^+, M, B^-)$ with $|M| = \emptyset$. During the execution of the procedure, the $\varepsilon/4$-optimality of the independent partial assignment is maintained and at the termination the obtained independent partial assignment M satisfies $r - |M| \leq \sqrt{r}$. Procedure SuccessiveShortestPath starts with this independent partial assignment and further enlarges the size of M one by one through successive shortest path augmentation steps, which yields an $\varepsilon/2$-optimal independent assignment at the termination.

The optimal independent assignment algorithm is described as follows. Details of the subprocedures will be given in the next section. The value of the parameter ε is not changed during the execution of procedure Refine and its subprocedures Auction and SuccessiveShortestPath. The input L can be any positive integer and will be optimized later (in Section 3.8).

Algorithm Assignment

Input: $\mathcal{N} = (G = (V^+, V^-; A), \mathcal{M}^+, \mathcal{M}^-, c)$, a potential $p \equiv \mathbf{0}$, a positive integer L, and $\varepsilon = C = \max\{|c(a)| \mid a \in A\}$.
Output: An optimal independent assignment $\Delta = (B^+, M, B^-)$ of $\mathcal{N}$.
Step 1: While $\varepsilon \geq 1/|4r|$, put $\varepsilon' \leftarrow \varepsilon/4$, perform procedure Refine(ε', L, p) and put $\varepsilon \leftarrow \varepsilon/2$.
(End)

Procedure Refine(ε, L, p)

Input: $\mathcal{N}$, L, ε, and p such that there exists a 4ε-optimal independent assignment with respect to p.
Output: A potential p and a 2ε-optimal independent assignment $\Delta = (B^+, M, B^-)$ of

$\mathcal{N}$ with respect to p.

Step 1: Perform procedure Auction(ε, L, p).

Step 2: Perform procedure SuccessiveShortestPath($\varepsilon, p, \Delta = (B^+, M, B^-)$).

(End)

3.7. A Refinement of the Approximate Optimality

For an ε-optimal independent partial assignment $\Delta = (B^+, M, B^-)$ with respect to a potential p, an arc $a \in A_\Delta$ is called an *admissible arc* in $\mathcal{N}_\Delta$ if $-\varepsilon \le c_{\Delta,p}(a) < 0$. Each $v \in B^+ - \partial^+ M$ ($v \in B^- - \partial^- M$) is called a *source* (*sink*) vertex.

For each $v \in V^*$ we define a basic operation Relabel(v) for our procedure Refine.

Relabel(v): Applicability: $v \in V^*$ and for any $a \in A_\Delta$ with $\partial^+ a = v$ we have $c_{\Delta,p}(a) \ge 0$;

Action: Put $p(v) \leftarrow p(v) - \varepsilon$.

We can easily see the following.

Lemma 3.7.1: *The relabeling operation keeps the ε-optimality of $\Delta = (B^+, M, B^-)$ with respect to the updated potential p.*

A directed path of $\mathcal{N}_\Delta$ starting from a source vertex and consisting of only admissible arcs is called an *admissible path*. We consider the following three types of admissible path P. We denote by $(v \to w)$ a path consisting of a single arc (v, w) and by $(v \to w \to u)$ a path of arcs (v, w) and (w, u).

Type 1: $P = (v \to w)$ such that $(v, w) \in A_{B^+}$ and $p(w) = \max \{p(u) \mid (v, u) \in A_{B^+}\}$.

Type 2: $P = (v \to w)$ such that $v \in B^+ - \partial^+ M$ and $w \in B^-$.

Type 3: $P = (v \to w \to u)$ such that $(w, u) \in A_{B^-}$ and $p(u) = \max \{p(z) \mid (w, z) \in A_{B^-}\}$.

Three types of push operations are defined for our procedure Refine. Push operations are performed on admissible paths of the above three types.

Push1(P): Applicability: $P = (v \to w)$ is an admissible path of Type 1.
Action: Put $B^+ \leftarrow B^+ + w - v$.
Push2(P): Applicability: $P = (v \to w)$ is an admissible path of Type 2.
Action: Put $M \leftarrow M + (v, w) - \{a \in M \mid \partial^- a = w\}$.
Push3(P): Applicability: $P = (v \to w \to u)$ is an admissible path of Type 3.
Action: Put $B^- \leftarrow B^- + w - u$ and $M \leftarrow M + (v, w) - \{a \in M \mid \partial^- a = u\}$.

Lemma 3.7.2: *All three types of pushes maintain conditions* (3.5.12)~(3.5.14) *and the ε-optimality of $\Delta = (B^+, M, B^-)$ with respect to the current potential p.*

Proof: It is easily verified that Δ satisfies conditions (3.5.12)~(3.5.14) after the push operations. We prove that the ε-optimality is also maintained. Suppose that by Push1(P) the action $B_1^+ \leftarrow B^+ + w - v$ yields a new arc $(u, z) \in A_{B_1^+}$. From Lemma 3.4.3 we have

(i) $v = z$ or $(v, z) \in A_{B^+}$,

(ii) $u = w$ or $(u, w) \in A_{B^+}$.

(i), (ii) and the ε-optimality imply that $p(v) - p(z) \geq -\varepsilon$ and $p(u) - p(w) \geq -\varepsilon$. If $v \neq z$, then the selection of w implies $p(z) \leq p(w)$. Hence, $p(u) - p(z) \geq p(u) - p(w) \geq -\varepsilon$. If $v = z$, then

$$p(u) - p(z) = p(u) - p(v) \geq p(w) - p(v) - \varepsilon > -\varepsilon, \tag{3.7.19}$$

where note that $p(w) - p(v) > 0$ due to the admissibility of arc (v, w).

For Push2(P), the only new arc introduced is (w, v). Since (v, w) is admissible before the push, we have $c(w, v) + p(w) - p(v) > 0 > -\varepsilon$.

The case of Push3(P) can be verified similarly. □

The first subprocedure of procedure Refine is Auction. The basic operations in procedure Auction are Relabelings and Pushes. In procedure Auction, the number of Relabelings on each source vertex is not more than $L + 4$. The size of M is enlarged through push operations. If L is selected large enough, procedure Auction will terminate with an ε-optimal independent assignment (see Lemma 3.8.5). Procedure Auction is described as follows.

Procedure Auction(ε, L, p)

Input: L and p such that there exists a 4ε-optimal independent assignment with respect to p.
Output: A potential p and an ε-optimal independent partial assignment $\Delta = (B^+, M, B^-)$ with respect to p.
Step 1: Put $M = \emptyset$ and $p(v) \leftarrow p(v) - 4\varepsilon$ for any $v \in V^-$. Find a base B^+ of $\mathcal{M}^+$ such that

$$p(B^+) = \max_{B \in \mathcal{B}^+} p(B) \tag{3.7.20}$$

and B^- of $\mathcal{M}^-$ such that

$$p(B^-) = \min_{B \in \mathcal{B}^-} p(B). \tag{3.7.21}$$

Step 2: If there exists no source vertex v which is relabeled less than $L{+}4$ times, then the procedure terminates and let the current $\Delta = (B^+, M, B^-)$ and p be the output. Otherwise, find a source vertex v relabeled less than $L + 4$ times.
Step 3: If for each $a \in A_\Delta$ with $\partial^+ a = v$ we have $c_{\Delta,p}(a) \geq 0$, then perform Relabel(v) and go to Step 2. Otherwise go to Step 4.
Step 4: Let $V(v) = \{u \mid (v, u) \in A_\Delta \text{ and is admissible}\}$ ($V(v)$ is not empty within this step).
(4-1) Applicability: $V(v) \cap B^- \neq \emptyset$.
Find $u \in V(v) \cap B^-$. Perform Push2$(v \rightarrow u)$ and Relabel(u), and go to Step 2.
(4-2) Applicability: $V(v) \cap B^- = \emptyset$ and $V(v) \cap (V^- - B^-) \neq \emptyset$.
Find $u \in V(v) \cap (V^- - B^-)$. If for each $a \in A_\Delta$ with $\partial^+ a = u$ we have $c_{\Delta,p}(a) \geq 0$, then perform Relabel(u) and go to Step 2. Otherwise find $w \in B^-$ such that $P = (v \rightarrow u \rightarrow w)$ is an admissible path of Type 3. Perform Push3(P) and Relabel(u), and go to Step 2.
(4-3) Applicability: $V(v) \cap V^- = \emptyset$ and $V(v) \cap (V^+ - B^+) \neq \emptyset$.
Find $w \in V^+ - B^+$ such that $P = (v \rightarrow w)$ is an admissible path of Type 1; perform

Push1(P), Relabel(v) and go to Step 2.
(End)

Lemma 3.7.3: *The relabeling operations in procedure Auction are performed only when they are applicable.*

Proof: The relabeling operation in Step 3 and the first one in Step (4-2) are obviously valid. For the relabeling operation in Step (4-1) and the second one in Step (4-2), note that there is only one arc (u, v) going out from u after the corresponding push operation. Before the push, (v, u) is an admissible arc, which implies $c(u, v) + p(u) - p(v) > 0$. Therefore, such relabeling operations are applicable. For the relabeling operation in Step (4-3), since all the arcs going out from v are from v to V^- and not admissible in this step. So the relabeling operation is valid. □

We can easily see that the independent partial assignment $\Delta = (B^+, M, B^-)$ defined in Step 1 of procedure Auction is 0-optimal. Because of Lemmas 3.7.1∼3.7.3, we get an ε-optimal independent partial assignment and the corresponding potential p at the end of procedure Auction. Starting with them, we perform procedure SuccessiveShortest-Path described below. We get a 2ε-optimal independent assignment at the termination of procedure SuccessiveShortestPath. In procedure SuccessiveShortestPath, the cost function c and the potential p obtained at the end of procedure Auction are modified into $\bar{c}$ and $\bar{p}$ such that the initial independent partial assignment in procedure SuccessiveShortestPath is 0-optimal with respect to $\bar{c}$ and $\bar{p}$, and then the size of the independent partial assignment is enlarged one by one through successive shortest path augmentation steps. The augmentation step is essentially the same as that of M. Iri and N. Tomizawa [53].

Procedure SuccessiveShortestPath($\varepsilon, p, \Delta = (B^+, M, B^-)$)

Input: A potential p and an ε-optimal independent partial assignment $\Delta = (B^+, M, B^-)$ with respect to p.
Output: A potential p and a 2ε-optimal independent assignment $\Delta = (B^+, M, B^-)$ with respect to p.

Step 1: Put

$$\bar{p}(v) = \begin{cases} p(v) - \varepsilon & \text{for } v \in (V^+ - B^+) \cup B^- \\ p(v) & \text{for } v \in (V^- - B^-) \cup B^+ \end{cases} \tag{3.7.22}$$

and

$$\bar{c}(a) = \begin{cases} \bar{p}(\partial^- a) - \bar{p}(\partial^+ a) & \text{for } a \in M \\ \max\{\bar{p}(\partial^- a) - \bar{p}(\partial^+ a), c(a)\} & \text{for } a \in A - M. \end{cases} \tag{3.7.23}$$

Similarly as (3.5.18) we define $\bar{c}_\Delta : A_\Delta \to \mathrm{R}$ in terms of $\bar{c}$ instead of c.

Step 2: For each $a \in A_\Delta$ let $l(a) = \bar{c}_\Delta(a) + \bar{p}(\partial^+ a) - \bar{p}(\partial^- a)$ be the length of arc a. For each $v \in V^*$ let $\tilde{p}(v)$ be the length of a shortest path from the source vertex set $S^+ = B^+ - \partial^+ M$ to vertex v in $\mathcal{N}_\Delta$. If there exists some sink vertex $u \in B^- - \partial^- M$ which is not reachable from S^+, stop (there is no r-independent matching in $\mathcal{N}$). Otherwise go to Step 3.

Step 3: Choose a fixed sink vertex w and find a shortest directed path P in $\mathcal{N}_\Delta$ from S^+ to w; if there are more than one such path, choose one which consists of the fewest number of arcs. Denote the arc set of P by A_P. Put

$$\begin{aligned} B^+ &\leftarrow (B^+ - \{\partial^+ a \mid a \in A_P \cap A_{B^+}\}) \cup \{\partial^- a \mid a \in A_P \cap A_{B^+}\}, && (3.7.24) \\ B^- &\leftarrow (B^- \cup \{\partial^+ a \mid a \in A_P \cap A_{B^-}\}) - \{\partial^- a \mid a \in A_P \cap A_{B^-}\}, && (3.7.25) \\ M &\leftarrow (M \cup (A_P \cap A)) - \{a \mid \bar{a} \in A_P \cap \tilde{M}\}, && (3.7.26) \\ \bar{p} &\leftarrow \bar{p} + \tilde{p}. && (3.7.27) \end{aligned}$$

Step 4: If $|M| = r$, then put $p \leftarrow \bar{p}$ and stop. Otherwise go to Step 2.
(End)

For simplifying our argument we assume in Step 3 that every $v \in V^*$ is reachable from $S^+ = B^+ - \partial^+ M$.

Lemma 3.7.4: *If procedure SuccessiveShortestPath stops at Step 2, then there is no r-independent matching in $\mathcal{N}$.* □

The proof of this lemma will be given in Section 3.8.

The remaining of this section is the proof of the validity of procedure SuccessiveShortestPath. The argument is similar to that of M. Iri and N. Tomizawa [53].

It is straightforward to see that in Step 1 of procedure SuccessiveShortestPath $\Delta = (B^+, M, B^-)$ is a 0-optimal independent partial assignment with respect to $\bar{p}$ and $\bar{c}$.

Lemma 3.7.5: *In Step 2, $\Delta = (B^+, M, B^-)$ is 0-optimal independent partial assignment with respect to the potential $\bar{p} + \tilde{p}$ and cost function $\bar{c}$.*

Proof: By the definition of $\tilde{p}$ we have $\tilde{p}(\partial^- a) \le \tilde{p}(\partial^+ a) + l(a)$ for each $a \in A_\Delta$, i.e., $\bar{c}_\Delta(a) + \tilde{p}(\partial^+ a) + \bar{p}(\partial^+ a) - (\tilde{p}(\partial^- a) + \bar{p}(\partial^- a)) \ge 0$. □

Lemma 3.7.6: *After an execution of Step 3 $\Delta = (B^+, M, B^-)$ satisfies conditions (3.5.12)~(3.5.14).*

Proof: We begin by verifying condition (3.5.13). From the definition of P we have for any $a \in A_P$ $\tilde{p}(\partial^- a) = \tilde{p}(\partial^+ a) + l(a)$. It follows that

$$\bar{p}(\partial^- a) - \bar{p}(\partial^+ a) = \bar{c}_\Delta(a). \tag{3.7.28}$$

Denote $\Delta = (B^+, M, B^-)$ obtained at the beginning of Step 3 by $\Delta_1 = (B_1^+, M_1, B_1^-)$. Suppose that the arc set $A_P \cap A_{B_1^+}$ is given by $\{a_1, \cdots, a_q\}$ with $a_i = (u_i, v_i)$ $(i = 1, \cdots, q)$. Since $\bar{c}_\Delta(a) = 0$ for $a \in A_P \cap A_{B_1^+}$, we have $\bar{p}(u_i) = \bar{p}(v_i)$ $(i = 1, \cdots, q)$ from (3.7.28). Also by definition, at the end of Step 3

$$B^+ = (B_1^+ - \{u_1, \cdots, u_q\}) \cup \{v_1, \cdots, v_q\}. \tag{3.7.29}$$

Without loss of generality, let u_i's and v_i's be numbered in such a way that

$$\bar{p}(u_i) = \bar{p}(v_i) \le \bar{p}(u_j) = \bar{p}(v_j) \quad (1 \le i < j \le q), \tag{3.7.30}$$

and that if $\bar{p}(u_i) = \bar{p}(v_i) = \bar{p}(u_j) = \bar{p}(v_j)$ $(i < j)$, then a_i lies nearer to the initial vertex of path P than a_j along P. From these assumptions it is seen that there exists no arc (u_i, v_j) in $A_{B_1^+}$ with $1 \le i < j \le q$ due to the 0-optimality and the way of selecting P. Hence, by Lemma 3.4.2 B^+ is a base of $\mathcal{M}^+$. By a similar reasoning, we can show that B^- is also a base of $\mathcal{M}^-$. The verification of conditions (3.5.12) and (3.5.14) on $\Delta = (B^+, M, B^-)$ is easy. □

Lemma 3.7.7: *After an execution of Step 3 $\Delta = (B^+, M, B^-)$ is a 0-optimal independent partial assignment with respect to the current potential $\bar{p}$ and cost function $\bar{c}$.*

Proof: The notations are the same as in the proof of Lemma 3.7.6. We prove that for each $a \in A_\Delta - A_{\Delta_1}$ we have $\bar{c}_\Delta(a) + \bar{p}(\partial^+ a) - \bar{p}(\partial^- a) \geq 0$. Here,

$$A_\Delta - A_{\Delta_1} = (A_{B^+} - A_{B_1^+}) \cup (A_{B^-} - A_{B_1^-}) \cup (\tilde{M} - \tilde{M}_1). \qquad (3.7.31)$$

For any $a \in \tilde{M} - \tilde{M}_1$ we have $\bar{a} \in A_P \cap A$. From (3.7.28) we get $\bar{c}_\Delta(a) + \bar{p}(\partial^+ a) - \bar{p}(\partial^- a) = 0$.

Next, consider the arcs in $A_{B^+} - A_{B_1^+}$. Define

$$I_m = (B_1^+ - \{u_1, \cdots, u_m\}) \cup \{v_1, \cdots, v_m\} \quad (m = 1, \cdots, q) \qquad (3.7.32)$$

and $I_0 = B_1^+$. Then, from Lemma 3.4.2 $I_m = I_{m-1} - u_m + v_m$ is a base of $\mathcal{M}^+$ for each $m = 1, \cdots, q$. Note that $I_q = B^+$. We prove by induction on $m = 0, \cdots, q$ that for each $m = 0, \cdots, q$ and $a \in A_{I_m}$ we have $\bar{p}(\partial^+ a) - \bar{p}(\partial^- a) \geq 0$. This is true for $m = 0$ due to Lemma 3.7.5. Suppose that it is true for $m = k - 1$ $(1 \leq k \leq q)$. For $m = k$, let $a = (w, z) \in A_{I_m} - A_{I_{m-1}}$. From Lemma 3.4.3 we have

(i) $u_m = z$ or $(u_m, z) \in A_{I_{m-1}}$,

(ii) $v_m = w$ or $(w, v_m) \in A_{I_{m-1}}$.

Therefore, $\bar{p}(u_m) \geq \bar{p}(z)$ and $\bar{p}(w) \geq \bar{p}(v_m)$. It follows that $\bar{p}(w) \geq \bar{p}(z)$ since $\bar{p}(u_m) = \bar{p}(v_m)$. Thus the induction assumption is true for $m = k$, which is the required conclusion.

For $a \in A_{B^-} - A_{B_1^-}$, the proof is similar. □

From Lemma 3.7.7, the arc length $l(a)$ defined in Step 2 is nonnegative for each $a \in A_\Delta$. Consequently, $\tilde{p}(v)$ is well defined and can be computed efficiently by Dijkstra's algorithm.

Lemma 3.7.8: *The output $\Delta = (B^+, M, B^-)$ of procedure SuccessiveShortestPath is a 2ε-optimal independent assignment with respect to the corresponding p and c.*

Proof: From (3.7.23) we have $|\bar{c}_\Delta(a) - c_\Delta(a)| \leq 2\varepsilon$ for all $a \in A_\Delta$. Hence, the present lemma follows from Lemma 3.7.7. □

3.8. The Complexity of the Algorithm

In the algorithm we assume an oracle, called an independence oracle, for testing whether a given set is independent. Clearly, in our algorithm procedure Refine is executed O(log(rC)) times. Procedure Refine is divided into procedures Auction and SuccessiveShortestPath. We first analyze procedure Auction.

Lemma 3.8.1: *During an execution of procedure Auction each vertex in V^* can be relabeled at most $L+4$ times and thus the total number of relabeling operations is at most $(L+4)|V^*|$.*

Proof: For each $v \in V^+$ we relabel v only when it is relabeled less than L+4 times. So, it is relabeled at most $L+4$ times. For each $u \in V^-$, at the moment of the last relabeling on u, there exists $v \in V^+$ such that (v,u) is admissible. Thus we have $p(v) - p(u) + c(v,u) < 0$. Denote the potential obtained at the end of Step 1 by $\bar{p}$. Then, $\Delta = (B^+, M, B^-)$ in Step 1 is a 0-optimal independent partial assignment associated with $\bar{p}$. Since $(v,u) \in A_\Delta$, we have $\bar{p}(v) - \bar{p}(u) + c(v,u) \geq 0$. Hence,

$$p(u) - \bar{p}(u) > p(v) + c(v,u) - \bar{p}(v) - c(v,u) \geq -(L+4)\varepsilon. \tag{3.8.33}$$

Therefore, u can be relabeled at most $L+4$ times. □

Lemma 3.8.2: *The total number of push operations during an execution of procedure Auction is at most $(L+4)|V^*|$.*

Proof: Since each push operation is followed by a relabeling operation, the present lemma follows from Lemma 3.8.1. □

Each push or relabeling operation requires O($|V^*|$) time for searching an arc. Hence, we have

Theorem 3.8.3: *The complexity of procedure Auction is* O($L|V^*|^2$). □

The efficiency of our algorithm depends on the size of the final M obtained by procedure Auction which can be controlled by the choice of L. A relation between L and the size of M will be given in Lemma 3.8.5.

Assume that the outputs of procedure Auction are $\Delta = (B^+, M, B^-)$ and p, and the input is p_1. Let $\Delta_1 = (B_1^+, M_1, B_1^-)$ be a 4ε-optimal independent assignment with respect to p_1. If $B^+ - B_1^+$ is not empty, suppose $B^+ - B_1^+ = \{v_1^+, \cdots, v_t^+\}$ and put $C_i^+ = \mathrm{C}^+(B_1^+|v_i^+)$ $(i = 1, \cdots, t)$. Also define

$$T_i^+ = \mathrm{K}^+(B^+|v_i^+) - v_i^+ \quad (i = 1, \cdots, t). \tag{3.8.34}$$

Apply Lemma 3.4.4 to these two families $\{C_i^+ \mid i = 1, \cdots, t\}$ and $\{T_i^+ \mid i = 1, \cdots, t\}$, let $\{u_1^+, \cdots, u_t^+\}$ $(= B_1^+ - B^+)$ be the common transversal of them. (It may be noted here that obtaining a transversal of each of the two families is enough for the present argument, but we use Lemma 3.4.4 that appeared in the original version of [38].) Then, for a bijection f from $\{1, \cdots, t\}$ to itself we have $u_i^+ \in C_{f(i)}^+ \cap T_i^+$ for each $i = 1, \cdots, t$, i.e., $(v_i^+, u_i^+) \in A_\Delta$ and $(u_i^+, v_{f(i)}^+) \in A_{\Delta_1}$ for each $i = 1, \cdots, t$. Similarly, if $B_1^- - B^-$ is not empty, suppose $B_1^- - B^- = \{v_1^-, \cdots, v_s^-\}$ and let $\{u_1^-, \cdots, u_s^-\}$ $(= B^- - B_1^-)$ be the common transversal of $C_i^- = \mathrm{C}^-(B^-|v_i^-)$ $(i = 1, \cdots, s)$ and

$$T_i^- = \mathrm{K}^-(B_1^-|v_i^-) - v_i^- \quad (i = 1, \cdots, s). \tag{3.8.35}$$

There exists a bijection g from $\{1, \cdots, s\}$ to itself such that $(v_i^-, u_i^-) \in A_\Delta$ and $(u_i^-, v_{g(i)}^-) \in A_{\Delta_1}$ for each $i = 1, \cdots, s$.

For each vertex $v \in V$ in a directed graph $G = (V, A)$ we define

$$\begin{aligned} \mathrm{d}_G^+ v &= |\{a \mid a \in A,\ \partial^+ a = v\}|, & (3.8.36)\\ \mathrm{d}_G^- v &= |\{a \mid a \in A,\ \partial^- a = v\}|. & (3.8.37) \end{aligned}$$

We consider two directed graphs $H = (W, F)$ and $H_1 = (W, F_1)$ that are, respectively, subgraphs of $\mathcal{N}_\Delta$ and $\mathcal{N}_{\Delta_1}$ with $W = B_1^+ \cup B^+ \cup B_1^- \cup B^-$ and

$$\begin{aligned} F &= M_1 \cup \tilde{M} \cup (\cup_{i=1}^t \{(v_i^+, u_i^+)\}) \cup (\cup_{i=1}^s \{(v_i^-, u_i^-)\}), & (3.8.38)\\ F_1 &= \tilde{M}_1 \cup M \cup (\cup_{i=1}^t \{(u_i^+, v_{f(i)}^+)\}) \cup (\cup_{i=1}^s \{(u_i^-, v_{g(i)}^-)\}). & (3.8.39) \end{aligned}$$

Let $S^+ = B^+ - \partial^+ M$ and $S^- = B^- - \partial^- M$. Note that $B_1^+ = \partial^+ M_1$ and $B_1^- = \partial^- M_1$. It is easy to verify the following properties of H and H_1:

$$\mathrm{d}_H^+ v = \mathrm{d}_H^- v = \mathrm{d}_{H_1}^+ v = \mathrm{d}_{H_1}^- v = 1 \quad \text{for } v \in W - S^+ - S^-, \tag{3.8.40}$$

$$\mathrm{d}^+_H v = \mathrm{d}^-_{H_1} v = \begin{cases} 1 & \text{for } v \in S^+ \\ 0 & \text{for } v \in S^- \end{cases} \tag{3.8.41}$$

and

$$\mathrm{d}^-_H v = \mathrm{d}^+_{H_1} v = \begin{cases} 0 & \text{for } v \in S^+ \\ 1 & \text{for } v \in S^-. \end{cases} \tag{3.8.42}$$

It follows from (3.8.40)$\sim$(3.8.42) that

the graph $H = (W, F)$ is decomposed into $|S^+|$ $(= |S^-|)$ vertex-disjoint directed paths from S^+ to S^- and some vertex-disjoint directed cycles, (3.8.43)

the graph $H_1 = (W, F_1)$ is decomposed into $|S^-|$ $(= |S^+|)$ vertex-disjoint directed paths from S^- to S^+ and some vertex-disjoint directed cycles. (3.8.44)

Lemma 3.7.4 follows from (3.8.43). Furthermore, we have

$$|F| = |F_1| = r + |M| + s + t \le 3r + |M|. \tag{3.8.45}$$

Let $\lfloor x \rfloor$ be the largest integer not exceeding x and $\lceil x \rceil$ be the smallest integer not less than x.

Lemma 3.8.4: *The arc set F defined in* (3.8.38) *has no less than* $\lceil (r - |M|)L/5 \rceil$ *arcs.*

Proof: Let $\Delta_1 = (B_1^+, M_1, B_1^-)$, $\Delta = (B^+, M, B^-)$, p_1 and p be those appearing above. From (3.8.43) and the ε-optimality of $\Delta = (B^+, M, B^-)$ in $\mathcal{N}_\Delta$ we have

$$\sum_{a \in F} c_{\Delta,p}(a) = \sum_{v \in S^+} p(v) - \sum_{v \in S^-} p(v) + \sum_{a \in F} c_\Delta(a) \ge -|F|\varepsilon. \tag{3.8.46}$$

On the other hand, from (3.8.44) and the 4ε-optimality of Δ_1 we have

$$\sum_{a \in F_1} c_{\Delta_1,p_1}(a) = \sum_{v \in S^-} p_1(v) - \sum_{v \in S^+} p_1(v) + \sum_{a \in F_1} c_{\Delta_1}(a) \ge -4|F_1|\varepsilon. \tag{3.8.47}$$

Note that $p(v) = p_1(v) - 4\varepsilon$ for $v \in S^-$, since the potential of sink vertex v decreases by 4ε at the beginning of procedure Auction and then remains unchanged. From the termination condition of procedure Auction, we have $p(v) = p_1(v) - (L+4)\varepsilon$ for $v \in S^+$. Also, $\sum_{a \in F_1} c_{\Delta_1}(a) = -\sum_{a \in F} c_\Delta(a)$. Hence, from (3.8.46) and (3.8.47),

$$\begin{aligned} -|F|\varepsilon &\le \sum_{v \in S^+} p(v) - \sum_{v \in S^-} p(v) + \sum_{a \in F} c_\Delta(a) \\ &= \sum_{v \in S^+} (p_1(v) - (L+4)\varepsilon) - \sum_{v \in S^-} (p_1(v) - 4\varepsilon) - \sum_{a \in F_1} c_{\Delta_1}(a) \\ &\le -(r - |M|)L\varepsilon + 4|F_1|\varepsilon. \end{aligned} \tag{3.8.48}$$

Consequently, $|F| \geq \lceil (r - |M|)L/5 \rceil$. □

From Lemma 3.8.4 we have

Lemma 3.8.5: *Let $\Delta = (B^+, M, B^-)$ be the output of procedure Auction. Then we have $|M| \geq \lceil (r - |M|)L/5 \rceil - 3r$. If we choose $L \geq 20r$, then $|M| = r$.*

Proof: It easily follows from (3.8.45) and Lemma 3.8.4 that $|M| \geq \lceil (r - |M|)L/5 \rceil - 3r$. Let L be not less than $20r$. If $|M| \neq r$, i.e., $r - |M| \geq 1$, then the above inequality would imply $|M| \geq \lceil L/5 \rceil - 3r = 4r - 3r = r$, a contradiction. Therefore, $|M| = r$. □

Next, we turn to analyze procedure SuccessiveShortestPath. In each iteration of Steps 2 and 3 in procedure SuccessiveShortestPath, M is augmented by one. Therefore, the number of such iterations is $r - |M|$, where $\Delta = (B^+, M, B^-)$ is the input. The dominating part of the computation is Dijkstra's algorithm for finding $\tilde{p}(v)$ and the required shortest directed path, which requires $\mathrm{O}(|V^*|^2)$ times. Hence, we get

Theorem 3.8.6: *The complexity of procedure SuccessiveShortestPath is $\mathrm{O}((r - |M|)$ $|V^*|^2)$ when M is the input.* □

Finally, we have

Theorem 3.8.7: *If we choose $L = \lceil 20\sqrt{r} - 5 \rceil$, then the complexity of the optimal independent assignment algorithm is $\mathrm{O}(\sqrt{r}|V^*|^2 \log(rC))$ with the independence oracle.*

Proof: Procedure Refine is executed $\mathrm{O}(\log(rC))$ times. From (3.8.45) and Lemma 3.8.4 we have $3r + |M| \geq \lceil (r - |M|)L/5 \rceil$. When $L = \lceil 20\sqrt{r} - 5 \rceil$, we have $r - |M| \leq \sqrt{r}$. Hence, Theorems 3.8.3 and 3.8.6 imply the present theorem. □

Chapter 4.

A Cost Scaling Algorithm for Minimum-Cost Submodular Flows

4.1. The Minimum-Cost Submodular Flow Problem and the Optimality Condition

In this chapter, we consider the problem of finding minimum-cost submodular flows. An algorithm for the problem is constructed which is a generalization of the algorithm for minimum-cost flows devised by A.V. Goldberg and R. E. Tarjan ([44]). Our algorithm uses the technique of cost scaling, the complexity of the algorithm is proved to be pseudo-polynomial and in case of minimum-cost 0-1 submodular flows the complexity is polynomial.

The definition of the minimum-cost submodular flow problem is given below. Let $G = (V, A)$ be a directed graph with a vertex set V ($|V| = n$) and an arc set A ($|A| = m$) with no self-loop and no multiple edges. Also let function $\bar{c} : A \rightarrow \mathrm{Z}$ (the set of all integers) be an upper capacity function, $\underline{c} : A \rightarrow \mathrm{Z}$ be a lower capacity function and $\gamma : A \rightarrow \mathrm{Z}$ be a cost function. $\mathcal{D} \subseteq 2^V$ is a family with $\emptyset, V \in \mathcal{D}$ and closed with respect to the set union and intersection. $f : \mathcal{D} \rightarrow \mathrm{Z}$ is a submodular function on $\mathcal{D}$ with $f(\emptyset) = f(V) = 0$. Given a flow function $\varphi : A \rightarrow \mathrm{R}$, if

$$\underline{c}(a) \leq \varphi(a) \leq \bar{c}(a) \quad (a \in A), \tag{4.1.1}$$

then φ is said to be a *feasible flow*.

A *minimum-cost submodular flow* (or an *optimal submodular flow*) is a feasible flow

$\varphi : A \to \mathrm{R}$ which is a solution to the following Problem $(\mathrm{P_s})$:

$$(\mathrm{P_s}) \quad \text{Minimize} \quad \sum_{a\in A} \gamma(a)\varphi(a) \tag{4.1.2}$$

$$\text{subject to} \quad \underline{c}(a) \le \varphi(a) \le \bar{c}(a) \ (a \in A), \tag{4.1.3}$$

$$\partial\varphi \in \mathrm{B}(f). \tag{4.1.4}$$

A flow φ satisfying (4.1.3) and (4.1.4) is called a *submodular flow.* The following theorem is concerned with the existence of a submodular flow φ.

Theorem 4.1.1 ([24]): *There exists a submodular flow for the submodular flow problem* $(\mathrm{P_s})$ *if and only if*

$$\forall X \in \mathcal{D}: \ (\kappa_{\underline{c},\bar{c}})^{\#}(X) \le f(X) \tag{4.1.5}$$

or

$$\forall X \in \mathcal{D}: \ \bar{c}(\Delta^- X) - \underline{c}(\Delta^+ X) + f(X) \ge 0, \tag{4.1.6}$$

where for each $X \subseteq V$ $\Delta^+ X = \{a \mid a \in A, \ \partial^+ a \in X, \ \partial^- a \in V - X\}$ *and* $\Delta^- X = \{a \mid a \in A, \ \partial^- a \in X, \ \partial^+ a \in V - X\}$. □

Since the capacity functions take finite real values, the set of all feasible solution of Problem $(\mathrm{P_s})$ is a bounded and closed set. It follows that there exists an optimal solution of Problem $(\mathrm{P_s})$ if the problem is feasible. We adopt a theorem from [34, p. 136], which shows an optimality condition for submodular flows.

Theorem 4.1.2: *A submodular flow* $\varphi : A \to \mathrm{R}$ *for Problem* $(\mathrm{P_s})$ *is optimal if and only if there exists a function* $p : V \to \mathrm{R}$ *such that, defining* $\gamma_p : A \to \mathrm{R}$ *by*

$$\gamma_p(a) = \gamma(a) + p(\partial^+ a) - p(\partial^- a) \ \ (a \in A), \tag{4.1.7}$$

we have for each $a \in A$

$$\gamma_p(a) > 0 \Longrightarrow \varphi(a) = \underline{c}(a), \tag{4.1.8}$$

$$\gamma_p(a) < 0 \Longrightarrow \varphi(a) = \bar{c}(a) \tag{4.1.9}$$

and such that the boundary $\partial\varphi : V \to \mathrm{R}$ is a maximum weight base of $\mathrm{B}(f)$ *with respect to the weight function p i.e.,*

$$\sum_{v\in V} \partial\varphi(v)p(v) = \max\{\sum_{v\in V} x(v)p(v) \mid x \in \mathrm{B}(f)\}. \tag{4.1.10}$$

$\square$

Given a pair $\Delta = (\varphi, z)$ called a *submodular pseudo-flow* formed by a feasible flow φ and a base $z \in \mathrm{B}(f)$, we define an auxiliary network $\mathcal{N}_\Delta = (G_\Delta = (V, A_\Delta), c_\Delta, \gamma_\Delta)$ where G_Δ is a directed graph with vertex set V and arc set A_Δ. A_Δ is defined as

$$\begin{aligned} A_\Delta &= A_\varphi \cup B_\varphi \cup C_z & (4.1.11)\\ A_\varphi &= \{a \mid a \in A, \varphi(a) < \bar{c}(a)\}, & (4.1.12)\\ B_\varphi &= \{a \mid \bar{a} \in A, \varphi(\bar{a}) > \underline{c}(\bar{a})\} \ \ (\bar{a} : \text{a reorientation of } a), & (4.1.13)\\ C_z &= \{(u,v) \mid u, v \in V,\ u \in \mathrm{dep}(z,v) - \{v\}\}. & (4.1.14) \end{aligned}$$

The capacity function $c_\Delta : A_\Delta \to \mathrm{R}$ is given by

$$c_\Delta(a) = \begin{cases} \bar{c}(a) - \varphi(a) & (a \in A_\varphi) \\ \varphi(\bar{a}) - \underline{c}(\bar{a}) & (a \in B_\varphi,\ \bar{a}\ (\in A) : \text{a reorientation of } a) \\ \tilde{c}(z, v, u) & (a = (u,v) \in C_z) \end{cases} \tag{4.1.15}$$

and $\gamma_\Delta : A_\Delta \to \mathrm{R}$ is the length function given by

$$\gamma_\Delta(a) = \begin{cases} \gamma(a) & (a \in A_\varphi) \\ -\gamma(\bar{a}) & (a \in B_\varphi,\ \bar{a}\ (\in A) : \text{a reorientation of } a) \\ 0 & (a = (u,v) \in C_z). \end{cases} \tag{4.1.16}$$

Theorem 4.1.3 ([34, p. 137]): *A submodular flow $\varphi : A \to \mathrm{R}$ for Problem* $(\mathrm{P_s})$ *is optimal if and only if there exists no directed cycle of negative length, relative to the length function γ_Δ, in the auxiliary network $\mathcal{N}_\Delta = (G_\Delta = (V, A_\Delta), c_\Delta, \gamma_\Delta)$ where $\Delta = (\varphi, z)$ with $z = \partial\varphi$.* $\square$

Using the auxiliary network $\mathcal{N}_\Delta = (G_\Delta = (V, A_\Delta), c_\Delta, \gamma_\Delta)$ and Lemma 1.5.5, we can rewrite the optimality condition given by Theorem 4.1.2 as follows.

Theorem 4.1.4: *A submodular flow $\varphi : A \to \mathrm{R}$ for Problem* $(\mathrm{P_s})$ *is optimal if and only if there exists a function $p : V \to \mathrm{R}$ such that, defining $\gamma_{\Delta,p} : A_\Delta \to \mathrm{R}$ by*

$$\gamma_{\Delta,p}(a) = \gamma_\Delta(a) + p(\partial^+ a) - p(\partial^- a) \quad (a \in A_\Delta), \tag{4.1.17}$$

we have for each $a \in A_\Delta$, $\gamma_{\Delta,p}(a) \geq 0$, where $\Delta = (\varphi, z)$ with $z = \partial\varphi$. □

For a positive real number ε we define ε-optimality for a submodular pseudo-flow $\Delta = (\varphi, z)$. This concept is fundamental for our cost scaling algorithm.

Definition 4.1.5: A submodular pseudo-flow $\Delta = (\varphi, z)$ is said to be ε-*optimal* if there exists a potential function $p : V \to \mathrm{R}$ such that $\gamma_{\Delta,p}(a) \geq -\varepsilon$ for all $a \in A_\Delta$. □

Put $\Gamma = \max_{a \in A} |\gamma(a)|$. Then, we have

Lemma 4.1.6: *Any submodular pseudo-flow is ε-optimal for $\varepsilon \geq \Gamma$ and any ε-optimal submodular flow ($z = \partial\varphi$) with $\varepsilon < 1/|V|$ is an optimal submodular flow.*

Proof: The first part of the lemma can be verified by taking $p \equiv \mathbf{0}$. For the second part of the lemma, we see that if $\varepsilon < 1/|V|$, then there is no negative directed cycle in $\mathcal{N}_\Delta = (G_\Delta = (V, A_\Delta), c_\Delta, \gamma_\Delta)$, since the length $\sum_{a \in C} \gamma_\Delta(a) = \sum_{a \in C} \gamma_{\Delta,p}(a)$ of each cycle C is an integer and is greater than or equal to $-\varepsilon|V| > -1$. Hence, the optimality of the submodular flow follows from Theorem 4.1.3. □

4.2. Examples of Submodular Flow Problems

In this section we describe several examples of submodular flow problems.

Strongly-connected orientation. Consider an undirected graph $G_u = (V, E)$, the *k-strongly-connected orientation problem* is to orient all edges of G_u to make it k-strongly-connected for a given positive integer k or show this is impossible. In the *minimum-cost k-strongly-connected orientation problem* each orientation of an edge has a cost and we seek a k-strongly-connected orientation of smallest possible cost. In [23], A. Frank has formulated this problem as a minimum-cost 0-1 submodular flow problem with the submodular function $\rho(X) - k$ (cf. Theorem 1.4.2).

Reachability orientation. Consider a mixed graph, with both directed arcs and undirected edges. Let s be a fixed vertex of the graph. The *k-reachability orientation problem* is to orient the undirected edges so that each vertex can be reached by k arc-disjoint directed path from s. The *minimum-cost k-reachability orientation problem* (defined similarly) can be formulated as a minimum-cost 0-1 submodular flow problem.

Both problems are related to the reliability theory.

Vertex reachability. Consider a directed graph $G = (V, A)$ with nonnegative arc-costs and a distinguished vertex s. The *minimum-cost k-vertex-reachability problem* is to find a minimum-cost set B of arcs such that any vertex can be reached from s by k vertex-disjoint directed paths consisting only arcs in B. A. Frank and É. Tardos [26] reduce this problem to a minimum-cost 0-1 submodular flow problem. It is a natural extension of the minimum-cost spanning arborescence problem.

Independent flow problem. Consider a network

$$\mathcal{N} = (G = (V, A), S^+, S^-, \bar{c}, \underline{c}, (\mathcal{D}^+, f^+), (\mathcal{D}^-, f^-), \gamma), \tag{4.2.18}$$

where $S^+, S^- \subset V$ with $S^+ \cap S^- = \emptyset$, $\bar{c}$ ($\underline{c}$) is the upper (lower) capacity function on A, $(\mathcal{D}^+, f^+)$ ($(\mathcal{D}^-, f^-)$) is a submodular system on S^+ (S^-), and γ is the cost function on A. The *minimum-cost independent flow problem* is defined by

$$\begin{aligned}
&\text{Minimize} && \textstyle\sum_{a \in A} \gamma(a)\varphi(a) && (4.2.19)\\
&\text{subject to} && \underline{c}(a) \le \varphi(a) \le \bar{c}(a) \ (a \in A), && (4.2.20)\\
& && (\partial\varphi)^{S^+} \in \mathrm{B}(f^+),\ (\partial\varphi)^{S^-} \in \mathrm{B}(f^-), && (4.2.21)\\
& && \partial\varphi(v) = 0 \ (v \in V - S^+ \cap S^-). && (4.2.22)
\end{aligned}$$

Here, $(\partial\varphi)^{S^+}$ ($(\partial\varphi)^{S^-}$) is the restriction of $\partial\varphi : V \to \mathbf{R}$ to S^+ (S^-).

The independent flow problem was first considered by S. Fujishige [28] which is the generalization of both the independent assignment problem and the submodular intersection problem. The problem has many engineering applications (see [51, 52]). Also, the independent flow problem can be reduced to a submodular flow problem and vice versa (see [34]).

Polymatroidal flow problem. Consider a network

$$\mathcal{N} = (G = (V, A), \underline{c}, \gamma, (\mathcal{D}_v^+, f_v^+), (\mathcal{D}_v^-, f_v^-)(v \in V)), \tag{4.2.23}$$

where $\underline{c}$ is a lower capacity function on A, γ is a cost function on A, and $(\mathcal{D}_v^+, f_v^+)$ $((\mathcal{D}_v^-, f_v^-))$ is a submodular system on δ^+v (δ^-v) for each $v \in V$ (see Section 1.2 for definitions of δ^+v and δ^-v). The *polymatroidal flow problem* (R. Hassin [48] and E. L. Lawler and C. U. Martel [60, 61]) is given as follows

$$\text{Minimize} \qquad \textstyle\sum_{a\in A} \gamma(a)\varphi(a) \tag{4.2.24}$$

$$\text{subject to} \qquad \underline{c}(a) \le \varphi(a) \ (a \in A), \tag{4.2.25}$$

$$\varphi^{\delta^+v} \in \mathrm{P}(f_v^+),\ \varphi^{\delta^-v} \in \mathrm{P}(f_v^-) \ (v \in V), \tag{4.2.26}$$

$$\partial\varphi(v) = 0 \ (v \in V), \tag{4.2.27}$$

where φ^{δ^+v} (φ^{δ^-v}) is the restriction of $\varphi : A \to \mathbf{R}$ to δ^+v (δ^-v). In [63] an application of the polymatroidal flow problem to multiprocessor scheduling problems has been studied. Also, the polymatroidal flow problem can be reduced to a submodular flow problem and vice versa (see [34]).

4.3. A Cost Scaling Algorithm

In our cost scaling algorithm, we execute a procedure called *Refine* which converts a 2ε-optimal submodular flow to an ε-optimal submodular pseudo-flow and then converts it to an ε-optimal submodular flow. Two basic operations called Relabel and Push are performed in procedure Refine. Given a submodular pseudo-flow $\Delta = (\varphi, z)$ and the corresponding auxiliary network $\mathcal{N}_\Delta = (G_\Delta = (V, A_\Delta), c_\Delta, \gamma_\Delta)$, suppose that we have a potential function p such that Δ is ε-optimal with respect to p. For each $v \in V$ let $e(v) = z(v) - \partial\varphi(v)$, which is called the *excess* on v. If $e(v) > 0$, then v is called an *active* vertex.

For an ε-optimal submodular pseudo-flow with respect to a potential p, an arc $a \in A_\Delta$ is called an *admissible arc* in $\mathcal{N}_\Delta = (G_\Delta = (V, A_\Delta), c_\Delta, \gamma_\Delta)$ if $-\varepsilon \le \gamma_{\Delta,p}(a) < 0$. Note that in our algorithm $p(v)/\varepsilon$ for any $v \in V$ is always an integer, and hence for each $a \in C_z$ a is an admissible arc if and only if $\gamma_{\Delta,p}(a) = -\varepsilon$.

The relabeling operation on $v \in V$ is defined as:

Relabel(v): Applicability: $e(v) > 0$ and for any $a \in A_\Delta$ with $\partial^+a = v$ we have $\gamma_{\Delta,p}(a) \ge 0$;

Action: $p(v) \leftarrow p(v) - \varepsilon$.

The push operation on $(v, w) \in A_\Delta$ is defined as:

Push1(v, w): Applicability: $e(v) > 0$, $\gamma_{\Delta,p}(v, w) < 0$ and $(v, w) \in A_\varphi \cup B_\varphi$;

Action:

If $(v, w) \in A_\varphi$, then $\varphi(v, w) \leftarrow \varphi(v, w) + \min(e(v), c_\Delta(v, w))$.

If $(v, w) \in B_\varphi$, then $\varphi(w, v) \leftarrow \varphi(w, v) - \min(e(v), c_\Delta(v, w))$.

Push2(v, w): Applicability: $e(v) > 0$, $(v, w) \in C_z$ and $\gamma_{\Delta,p}(v, w) = -\varepsilon$.

Action: $z \leftarrow z + \alpha(\chi_w - \chi_v)$ where $\alpha = \min(e(v), c_\Delta(v, w))$.

Lemma 4.3.1: *If v is an active vertex, then either a push for some $a \in A_\Delta$ with $\partial^+ a = v$ or a relabel of v is applicable.* □

An algorithm for the minimum-cost submodular flow problem is described as follows. The L in the input can be any positive integer at the moment and will be appropriately determined in the next section.

Algorithm Minimum-Cost Submodular Flow.

Input: $\mathcal{N} = (G = (V, A), \bar{c}, \underline{c}, \gamma, (\mathcal{D}, f))$, a positive integer L, a potential $p \equiv \mathbf{0}$ and $\varepsilon = \Gamma/2 = \max\{|\gamma(a)|/2 \mid a \in A\}$.

Output: A minimum-cost submodular flow φ in $\mathcal{N}$.

Step 1: While $\varepsilon \geq 1/(2|V|)$, perform procedure Refine(ε, L, p) and put $\varepsilon \leftarrow \varepsilon/2$.

(End)

Procedure Refine(ε, L, p).

Input: ε, L, and p such that there exists a 2ε-optimal submodular flow φ with respect to p.

Output: A potential p and an ε-optimal submodular flow φ with respect to p.

Step 0: For the current p, find an integer vector z_0 in $\mathrm{B}(f)$ such that

$$\sum_{v \in V} p(v) z_0(v) = \max_{z' \in \mathrm{B}(f)} \sum_{v \in V} p(v) z'(v). \tag{4.3.28}$$

Put $z \leftarrow z_0$. For each $(v, w) \in A$, if $\gamma_{\Delta,p}(v, w) < 0$ then put
$\varphi(v, w) \leftarrow \bar{c}(v, w)$,
otherwise put
$\varphi(v, w) \leftarrow \underline{c}(v, w)$.
Put $\Delta \leftarrow (\varphi, z)$.
Step 1: While there exists a vertex $v \in V$ that satisfies $e(v) > 0$ and has been relabeled less than L times, choose one such vertex v and do the following (1-1)~(1-3) (if there exists no such vertex, then the procedure terminates and let the current φ, ε and p be the output).
(1-1) Applicability: For any $a \in A_\Delta$ with $\partial^+ a = v$ we have $\gamma_{\Delta,p}(a) \geq 0$;
$p(v) \leftarrow p(v) - \varepsilon$.
(1-2) Applicability: $\gamma_{\Delta,p}(v, w) < 0$ and $(v, w) \in A_\varphi \cup B_\varphi$;
Perform Push1(v, w).
(1-3) Applicability: $\gamma_{\Delta,p}(v, w) = -\varepsilon$ and $(v, w) \in C_z$;
Perform Push2(v, w).
(End)

We have the following lemmas.

Lemma 4.3.2 (cf. Lemma 1.5.5): *The submodular pseudo-flow Δ defined in Step 0 of procedure Refine is 0-optimal with respect to the potential function in the input.*

Proof: $\gamma_{\Delta,p}(a) \geq 0$ for each $a \in A_\varphi \cup B_\varphi$ is directly from the definition of φ. $\gamma_{\Delta,p}(a) \geq 0$ for each $a \in C_z$ is from Lemma 1.5.5. □

Lemma 4.3.3: *The relabeling operation in procedure Refine keeps the ε-optimality of $\Delta = (\varphi, z)$ with respect to the updated potential p.* □

Lemma 4.3.4: *Both two types of push operations keep $\Delta = (\varphi, z)$ a submodular pseudo-flow and the ε-optimality of $\Delta = (\varphi, z)$ with respect to the current potential p.*

Proof: Since the potential function is not changed, it is enough to prove that $\gamma_{\Delta,p}(a) \geq -\varepsilon$ for any new arc generated by a push.

Suppose (w,s) is a new arc after a push operation on an admissible arc $(u,v) \in C_z$. By Lemma 1.4.12 we have
i) $u = s$ or $(u,s) \in C_z$ and
ii) $v = w$ or $(w,v) \in C_z$.

If $u \neq s$ and $w \neq v$, from i) we have $p(u) - p(s) \geq -\varepsilon$ and from ii) we have $p(w) - p(v) \geq -\varepsilon$. Hence

$$P(w) - p(s) \geq p(v) - p(s) - \varepsilon = p(u) + \varepsilon - p(s) - \varepsilon \geq -\varepsilon. \tag{4.3.29}$$

If $u = s$ and $w \neq v$, then

$$p(w) - p(s) \geq p(v) - p(s) - \varepsilon = p(v) - p(u) - \varepsilon = 0. \tag{4.3.30}$$

If $u \neq s$ and $w = v$, then

$$p(w) - p(s) \geq p(w) - p(u) - \varepsilon = p(v) - p(u) - \varepsilon = 0. \tag{4.3.31}$$

If $u = s$ and $w = v$, then $p(w) - p(s) = p(v) - p(u) = \varepsilon$.

Therefore $\gamma_{\Delta,p}(w,s) \geq -\varepsilon$ holds.

A push on an admissible arc $(u,v) \in A_\varphi \cup B_\varphi$ only produces a new arc (v,u) for which we have $\gamma_\Delta(v,u) + p(v) - p(u) > 0$. □

Lemma 4.3.5: *At the end of procedure Refine, the outputs are a potential p and an ε-optimal submodular flow with respect to the potential p.* □

Proof: The present lemma is from Lemma 4.3.3, 4.3.4, $\partial\varphi \geq z$ (since $e(v) \leq 0$) and $z(V) = f(V) = \partial\varphi(V) = 0$.

4.4. The Complexity of the Algorithm

Our cost scaling algorithm repeatedly performs procedure Refine. Obviously the iteration number of procedure Refine is $O(\log(|V|\Gamma))$ due to Lemmas 4.3.5 and 4.1.6. In this section, the complexity of procedure Refine is analyzed. We first estimate the number of relabelings on each vertex in V during an execution of procedure Refine and then estimate the number of pushes during an execution of procedure Refine.

We give a lemma and a corollary of it where $f(V) = 0$ is not assumed.

Lemma 4.4.1: *Let $z, z' \in \mathrm{B}(f)$ and $z(v), z'(v) \geq 0$ for all $v \in V$, where $(\mathcal{D}, f)$ is a submodular system on V. Suppose that for a potential function $p : V \to \mathrm{R}$ and $\varepsilon > 0$ we have $p(u) - p(v) \geq -\varepsilon$ for any $u, v \in V$ with $u \in \mathrm{dep}(z, v) - \{v\}$. Then,*

$$\sum_{v \in V} p(v)(z(v) - z'(v)) \geq -\varepsilon f(V). \tag{4.4.32}$$

Proof: Define a bipartite graph $G_b = (V, V'; A_z)$ where V' is a copy of V and $A_z = \{(u, v') \mid u, v \in V, u \in \mathrm{dep}(z, v)\}$. The upper capacities of the arcs in A_z are assumed to be infinity and the lower capacities of the arcs in A_z are assumed to be zero. For any subset U of V let $W = \{w \mid w \in V,\ u' \in U',\ (w, u') \in A_z\}$. It follows from the definition of A_z, W, Lemma 1.4.3 and Equation (1.4.11) that $z(W) = f(W)$ and $U \subseteq W$. Hence $z'(U) \leq z'(W) \leq f(W) = z(W)$. Consequently, from Theorem 1.2.2, there exists a function $g : A_z \to \mathrm{R}_+$ such that

$$g(\delta^+ u) = z(u) \quad (u \in V), \qquad g(\delta^- v') = z'(v') \quad (v' \in V'), \tag{4.4.33}$$

where

$$\delta^+ u = \{(u, v') \mid v' \in V', (u, v') \in A_z\}, \tag{4.4.34}$$

$$\delta^- v' = \{(u, v') \mid u \in V, (u, v') \in A_z\}, \tag{4.4.35}$$

and $z'(v') = z'(v)$ for $v' \in V'$.

Define $p(v') = p(v)$ for $v' \in V'$. Then,

$$\begin{aligned}
&\sum_{v \in V} p(v)(z(v) - z'(v)) \\
&= \sum_{u \in V} z(u)p(u) - \sum_{v' \in V'} z'(v')p(v') \\
&= \sum_{u \in V} g(\delta^+ u)p(u) - \sum_{v' \in V'} g(\delta^- v')p(v') \\
&= \sum_{a \in A_z} (p(\partial^+ a) - p(\partial^- a))g(a) \\
&\geq -\varepsilon \sum_{a \in A_z} g(a) \\
&= -\varepsilon \sum_{u \in V} z(u) \\
&= -\varepsilon f(V).
\end{aligned} \tag{4.4.36}$$

□

Corollary 4.4.2: *Let $z, z' \in \mathrm{B}(f)$ and $d \in \mathrm{R}^V$ be such that $z(v)+d(v), z'(v)+d(v) \geq 0$ for all $v \in V$, where $(\mathcal{D}, f)$ is a submodular system on V. Suppose that for a potential function $p : V \to \mathrm{R}$ and $\varepsilon > 0$ we have $p(u) - p(v) \geq -\varepsilon$ for any $u, v \in V$ with $u \in \mathrm{dep}(z, v) - \{v\}$. Then,*

$$\sum_{v \in V} p(v)(z(v) - z'(v)) \geq -\varepsilon(f(V) + d(V)). \tag{4.4.37}$$

□

Let $\Delta' = (\varphi', z' = \partial\varphi')$ be a 2ε-optimal submodular flow with respect to p' and $\Delta = (\varphi, z)$ be an ε-optimal submodular pseudo-flow with respect to p, where p' is the input of an execution of procedure Refine and $\Delta = (\varphi, z)$ and p are the current submodular pseudo-flow and the corresponding potential function within the execution of procedure Refine. Define

$$\begin{aligned} S^+ &= \{v \in V \mid z(v) - \partial\varphi(v) > 0\}, & (4.4.38)\\ S^- &= \{v \in V \mid z(v) - \partial\varphi(v) < 0\}, & (4.4.39)\\ E_+ &= \{(u,v) \in A_\varphi \mid \varphi'(u,v) > \varphi(u,v)\} \\ &\quad \cup\{(u,v) \in B_\varphi \mid \varphi(v,u) > \varphi'(v,u)\}, & (4.4.40)\\ E_- &= \{(u,v) \in A_{\varphi'} \mid \varphi(u,v) > \varphi'(u,v)\} \\ &\quad \cup\{(u,v) \in B_{\varphi'} \mid \varphi'(v,u) > \varphi(v,u)\}. & (4.4.41) \end{aligned}$$

Note that $p'(v) = p(v)$ for $v \in S^-$ since we only relabel active vertices. Then,

$$\begin{aligned} &\sum_{a \in E_+ \cap A_\varphi} \gamma_{\Delta,p}(a)(\varphi'(a) - \varphi(a)) + \sum_{a \in E_+ \cap B_\varphi} \gamma_{\Delta,p}(a)(\varphi(\bar{a}) - \varphi'(\bar{a})) \\ &= \sum_{a \in E_+ \cap A_\varphi} \gamma_\Delta(a)(\varphi'(a) - \varphi(a)) + \sum_{a \in E_+ \cap B_\varphi} \gamma_\Delta(a)(\varphi(\bar{a}) - \varphi'(\bar{a})) \\ &\quad + \sum_{a \in E_+ \cap A_\varphi} (p(\partial^+ a) - p(\partial^- a))(\varphi'(a) - \varphi(a)) \\ &\quad + \sum_{a \in E_+ \cap B_\varphi} (p(\partial^+ a) - p(\partial^- a))(\varphi(\bar{a}) - \varphi'(\bar{a})) \\ &= \sum_{a \in E_+ \cap A_\varphi} \gamma_\Delta(a)(\varphi'(a) - \varphi(a)) + \sum_{a \in E_+ \cap B_\varphi} \gamma_\Delta(a)(\varphi(\bar{a}) - \varphi'(\bar{a})) \\ &\quad + \sum_{v \in V} p(v)\Bigl(\sum_{a \in E_+ \cap A_\varphi,\ \partial^+ a = v} (\varphi'(a) - \varphi(a)) + \sum_{a \in E_+ \cap B_\varphi,\ \partial^+ a = v} (\varphi(\bar{a}) - \varphi'(\bar{a}))\Bigr) \end{aligned}$$

$$
\begin{aligned}
&-\sum_{v\in V} p(v)\Big(\sum_{a\in E_+\cap A_\varphi,\ \partial^- a=v} (\varphi'(a)-\varphi(a)) + \sum_{a\in E_+\cap B_\varphi,\ \partial^- a=v} (\varphi(\bar a)-\varphi'(\bar a))\Big)\\
&= \sum_{a\in E_+\cap A_\varphi} \gamma_\Delta(a)(\varphi'(a)-\varphi(a)) + \sum_{a\in E_+\cap B_\varphi} \gamma_\Delta(a)(\varphi(\bar a)-\varphi'(\bar a))\\
&\quad -\sum_{v\in V} p(v)(\partial\varphi(v)-\partial\varphi'(v))\\
&= \sum_{a\in E_+\cap A_\varphi} \gamma_\Delta(a)(\varphi'(a)-\varphi(a)) + \sum_{a\in E_+\cap B_\varphi} \gamma_\Delta(a)(\varphi(\bar a)-\varphi'(\bar a))\\
&\quad -\sum_{v\in V} p(v)(\partial\varphi(v)-z(v)) - \sum_{v\in V} p(v)(z(v)-z'(v))\\
&= \sum_{a\in E_+\cap A_\varphi} \gamma_\Delta(a)(\varphi'(a)-\varphi(a)) + \sum_{a\in E_+\cap B_\varphi} \gamma_\Delta(a)(\varphi(\bar a)-\varphi'(\bar a))\\
&\quad -\sum_{v\in S^+} p(v)(\partial\varphi(v)-z(v)) - \sum_{v\in S^-} p(v)(\partial\varphi(v)-z(v))\\
&\quad -\sum_{v\in V} p(v)(z(v)-z'(v)). \qquad (4.4.42)
\end{aligned}
$$

On the other hand, we have

$$
\begin{aligned}
&\sum_{a\in E_-\cap A_{\varphi'}} \gamma_{\Delta',p'}(a)(\varphi(a)-\varphi'(a)) + \sum_{a\in E_-\cap B_{\varphi'}} \gamma_{\Delta',p'}(a)(\varphi'(\bar a)-\varphi(\bar a))\\
&= \sum_{a\in E_-\cap A_{\varphi'}} \gamma_{\Delta'}(a)(\varphi(a)-\varphi'(a)) + \sum_{a\in E_-\cap B_{\varphi'}} \gamma_{\Delta'}(a)(\varphi'(\bar a)-\varphi(\bar a))\\
&\quad + \sum_{a\in E_-\cap A_{\varphi'}} (p'(\partial^+ a)-p'(\partial^- a))(\varphi(a)-\varphi'(a))\\
&\quad + \sum_{a\in E_-\cap B_{\varphi'}} (p'(\partial^+ a)-p'(\partial^- a))(\varphi'(\bar a)-\varphi(\bar a))\\
&= \sum_{a\in E_-\cap A_{\varphi'}} \gamma_{\Delta'}(a)(\varphi(a)-\varphi'(a)) + \sum_{a\in E_-\cap B_{\varphi'}} \gamma_{\Delta'}(a)(\varphi'(\bar a)-\varphi(\bar a))\\
&\quad +\sum_{v\in V} p'(v)\Big(\sum_{a\in E_-\cap A_{\varphi'},\ \partial^+ a=v} (\varphi(a)-\varphi'(a)) + \sum_{a\in E_-\cap B_{\varphi'},\ \partial^+ a=v} (\varphi'(\bar a)-\varphi(\bar a))\Big)\\
&\quad -\sum_{v\in V} p'(v)\Big(\sum_{a\in E_-\cap A_{\varphi'},\ \partial^- a=v} (\varphi(a)-\varphi'(a)) + \sum_{a\in E_-\cap B_{\varphi'},\ \partial^- a=v} (\varphi'(\bar a)-\varphi(\bar a))\Big)\\
&= \sum_{a\in E_-\cap A_{\varphi'}} \gamma_{\Delta'}(a)(\varphi(a)-\varphi'(a)) + \sum_{a\in E_-\cap B_{\varphi'}} \gamma_{\Delta'}(a)(\varphi'(\bar a)-\varphi(\bar a))\\
&\quad r -\sum_{v\in V} p'(v)(\partial\varphi'(v)-\partial\varphi(v))\\
&= \sum_{a\in E_-\cap A_{\varphi'}} \gamma_{\Delta'}(a)(\varphi(a)-\varphi'(a)) + \sum_{a\in E_-\cap B_{\varphi'}} \gamma_{\Delta'}(a)(\varphi'(\bar a)-\varphi(\bar a))\\
&\quad -\sum_{v\in V} p'(v)(z(v)-\partial\varphi(v)) - \sum_{v\in V} p'(v)(z'(v)-z(v))
\end{aligned}
$$

$$\begin{aligned}
&= \sum_{a\in E_-\cap A_{\varphi'}} \gamma_{\Delta'}(a)(\varphi(a)-\varphi'(a)) + \sum_{a\in E_-\cap B_{\varphi'}} \gamma_{\Delta'}(a)(\varphi'(\bar a)-\varphi(\bar a)) \\
&\quad - \sum_{v\in S^+} p'(v)(z(v)-\partial\varphi(v)) - \sum_{v\in S^-} p'(v)(z(v)-\partial\varphi(v)) \\
&\quad - \sum_{v\in V} p'(v)(z'(v)-z(v)). \qquad (4.4.43)
\end{aligned}$$

From (4.4.42),

$$\begin{aligned}
&\sum_{a\in E_+\cap A_\varphi} \gamma_{\Delta}(a)(\varphi'(a)-\varphi(a)) + \sum_{a\in E_+\cap B_\varphi} \gamma_{\Delta}(a)(\varphi(\bar a)-\varphi'(\bar a)) \\
&\quad - \sum_{v\in S^+} p(v)(\partial\varphi(v)-z(v)) - \sum_{v\in S^-} p(v)(\partial\varphi(v)-z(v)) \\
&= \sum_{a\in E_+\cap A_\varphi} \gamma_{\Delta,p}(a)(\varphi'(a)-\varphi(a)) + \sum_{a\in E_+\cap B_\varphi} \gamma_{\Delta,p}(a)(\varphi(\bar a)-\varphi'(\bar a)) \\
&\quad + \sum_{v\in V} p(v)(z(v)-z'(v)) \\
&\geq -\varepsilon d(V) - \varepsilon\Big(\sum_{a\in E_+\cap A_\varphi} (\varphi'(a)-\varphi(a)) + \sum_{a\in E_+\cap B_\varphi} (\varphi(\bar a)-\varphi'(\bar a))\Big). \qquad (4.4.44)
\end{aligned}$$

From (4.4.43),

$$\begin{aligned}
&\sum_{a\in E_-\cap A_{\varphi'}} \gamma_{\Delta'}(a)(\varphi(a)-\varphi'(a)) + \sum_{a\in E_-\cap B_{\varphi'}} \gamma_{\Delta'}(a)(\varphi'(\bar a)-\varphi(\bar a)) \\
&\quad - \sum_{v\in S^+} p'(v)(z(v)-\partial\varphi(v)) - \sum_{v\in S^-} p'(v)(z(v)-\partial\varphi(v)) \\
&= \sum_{a\in E_-\cap A_{\varphi'}} \gamma_{\Delta',p'}(a)(\varphi(a)-\varphi'(a)) + \sum_{a\in E_-\cap B_{\varphi'}} \gamma_{\Delta',p'}(a)(\varphi'(\bar a)-\varphi(\bar a)) \\
&\quad + \sum_{v\in V} p'(v)(z'(v)-z(v)) \\
&\geq -2\varepsilon d(V) - 2\varepsilon\Big(\sum_{a\in E_-\cap A_{\varphi'}} (\varphi(a)-\varphi'(a)) + \sum_{a\in E_-\cap B_{\varphi'}} (\varphi'(\bar a)-\varphi(\bar a))\Big). \qquad (4.4.45)
\end{aligned}$$

Putting $U = \max_{a\in A}(|\bar c(a)| + |\underline{c}(a)|)$ and adding the above two inequalities, we have

$$\sum_{v\in S^+} (p'(v)-p(v))(z(v)-\partial\varphi(v)) \leq 3\varepsilon d(V) + 12\varepsilon m U. \qquad (4.4.46)$$

Lemma 4.4.3: *If for a submodular pseudo-flow $\Delta = (\varphi, z)$ in procedure Refine, there exists $v \in V$ such that $e(v) > 0$ and $\{a \mid a \in A_\Delta, \partial^+ a = v\} = \emptyset$, then Problem* $(\mathrm{P_s})$ *is infeasible.*

Proof: If Problem ($\mathrm{P_s}$) is feasible, then procedure Refine is well defined since any submodular flow is Γ-optimal with respect to $p \equiv \mathbf{0}$ which is the input for the first procedure Refine. Let the current potential function be p and the potential function in inputs be p'. The inequality (4.4.46) holds. But $\{a \,|\, a \in A_\Delta, \partial^+ a = v\} = \emptyset$ implies that relabeling operations can be carried out any times on v. This is impossible by (4.4.46). Therefore, Problem ($\mathrm{P_s}$) is infeasible. □

If $\bar{c}$, $\underline{c}$, z and φ are integer vectors, f is an integer valued function and procedure Refine terminates when each active vertex is relabeled L (an integer) times, then from (4.4.46)

$$\sum_{v \in S^+} L\varepsilon \le \sum_{v \in S^+} (p'(v) - p(v))(z(v) - \partial\varphi(v)). \tag{4.4.47}$$

Hence,

$$L \le (3d(V) + 12mU)/|S^+|. \tag{4.4.48}$$

Theorem 4.4.4: *If we take $L > 3d(V) + 12mU$ and relabel each vertex at most L times, then procedure Refine terminates at (φ, z) such that $z = \partial\varphi$.*

Proof: Suppose on the contrary that procedure Refine terminates with (φ, z) such that $z \ne \partial\varphi$, i.e., $S^+ \ne \emptyset$. Then, the remaining active vertices are all relabeled exactly L times. Hence,

$$L \le (3d(V) + 12mU)/|S^+| \le 3d(V) + 12mU. \tag{4.4.49}$$

This is a contradiction. □

Define $x(v) = -U|\delta v|$ for each $v \in V$, where

$$\delta v = \{(u, w) \mid (u, w) \in A, u = v \text{ or } w = v\}. \tag{4.4.50}$$

If Problem ($\mathrm{P_s}$) has a feasible solution φ, then $x \in \mathrm{P}(f)$ since $x \le \partial\varphi \in \mathrm{B}(f)$. Let f_x be the contraction of f by the vector x. Replacing f by f_x in Problem ($\mathrm{P_s}$) does not change the set of all feasible submodular flows. For given z, $z' \in \mathrm{B}(f_x)$ as above, we have $z - x \ge \mathbf{0}$ and $z' - x \ge \mathbf{0}$ from Lemma 1.4.13. Then, from (4.4.46) and $-x(V) = \sum_{v \in V} U|\delta v| \le 2mU$ we have

$$\sum_{v \in S^+} (p'(v) - p(v))(z(v) - \partial\varphi(v)) \le 18\varepsilon mU. \tag{4.4.51}$$

Theorem 4.4.5: *If we take $L = 18mU + 1$ and relabel each vertex at most L times, then procedure Refine terminates at (φ, z) such that $z = \partial\varphi$.*

Proof: Let $d = -x$, the present theorem follows from Theorem 4.4.4. □

For a 0-1 minimum-cost submodular flow problem, we have

$$\sum_{v \in S^+} (p'(v) - p(v))(z(v) - \partial\varphi(v)) \leq 18\varepsilon m. \tag{4.4.52}$$

Inequality (4.4.52) is essential to construct a hybrid version algorithm for a 0-1 minimum-cost submodular flow problem which is similar to the approach in Chapter 3.

To estimate the number of pushes, we must make refinements on the order of basic operations to be carried out. First we make refinements on the order of selecting active vertices for operations of relabeling and push. We use a topological numbering on V which was initiated by H. N. Gabow and Y. Xu [40] for constructing a valid augmenting path.

Let $\tau : V \to \mathrm{Z}_+$ be a nonnegative integer function on V. τ is said to be a *topological numbering* on V if it satisfies the following:
(T1) If a is an admissible arc, $\tau(\partial^+ a) \geq \tau(\partial^- a)$,
(T2) If a is an admissible arc in $A_\varphi \cup B_\varphi$, $\tau(\partial^+ a) > \tau(\partial^- a)$.

Definition 4.4.6: An admissible arc $a \in A_\Delta$ is called a *strongly admissible arc* if $a \in A_\varphi \cup B_\varphi$, or $a \in C_z$ with $\tau(\partial^+ a) = \tau(\partial^- a)$. □

Lemma 4.4.7: *A push operation on a strongly admissible arc keeps τ a topological numbering on V.*

Proof: Suppose (w, s) is a new admissible arc after a push operation on a strongly admissible arc $(u, v) \in C_z$. By Lemma 1.4.12, we have
i) $u = s$ or $(u, s) \in C_z$ and
ii) $v = w$ or $(w, v) \in C_z$.

If $u \neq s$ and $w \neq v$, from i) we have $p(u) - p(s) \geq -\varepsilon$ and from ii) we have $p(w) - p(v) \geq -\varepsilon$. Since (w, s) is admissible, so

$$p(u) \geq p(s) - \varepsilon = p(w) \geq p(v) - \varepsilon = p(u). \tag{4.4.53}$$

Therefore, equality holds throughout the above inequalities and consequently (u, s) and (w, v) are admissible arcs. Thus $\tau(w) \geq \tau(v) = \tau(u) \geq \tau(s)$.

If $u = s$ and $w \neq v$, then $p(u) = p(s) = p(w) + \varepsilon \geq p(v) = p(u) + \varepsilon$, a contradiction.

If $u \neq s$ and $w = v$, then $p(u) \geq p(s) - \varepsilon = p(w) = p(v) = p(u) + \varepsilon$, a contradiction.

If $u = s$ and $w = v$, then $p(w) - p(s) = p(v) - p(u) = \varepsilon$, which implies that (w, s) is not an admissible arc in this case.

A push on an admissible arc $(u, v) \in A_\varphi \cup B_\varphi$ only produces a new arc (v, u) which is not an admissible arc since $\gamma_\Delta(v, u) + p(v) - p(u) > 0$. □

Suppose a relabeling is performed on v. Let T be the maximum value of current τ. Put $T \leftarrow T + 1$ and then put $\tau(v) \leftarrow T$. The modified τ is a valid topological numbering on V, since there is no admissible arc going into v right after v is relabeled. If during the algorithm we find that there is no strong admissible arc going out from v, then $\tau(v)$ is replaced by $\max\{\tau(w) \mid (v, w) \in C_z \text{ is admissible}\}$. The modified τ is still a valid topological numbering on V.

Now, we define a list $L_{(\tau,p)}$ of V. The vertices of V in $L_{(\tau,p)}$ are arranged in the order of the decreasing magnitude of the values of τ. For the vertices having the same value of τ, we arrange these vertices in the order of the increasing magnitude of the values of p. For the vertices having the same values of τ and of p, they can be ordered in any way. In list $L_{(\tau,p)}$ there is a vertex called the current vertex of $L_{(\tau,p)}$. Since $\tau(v) > \tau(w)$ holds for each admissible arc $(v, w) \in A_\varphi \cup B_\varphi$ and $\tau(v) \geq \tau(w)$, $p(v) = p(w) - \varepsilon$ holds for each admissible arc $(v, w) \in C_z$, we have an important property of list $L_{(\tau,p)}$, i.e., if (v, w) in A_Δ is an admissible arc, then v appears before w in $L_{(\tau,p)}$.

Initially, $\tau(v) = 0$ $(v \in V)$, $L_{(\tau,p)}$ contains the vertices in V in the order of the increasing magnitude of the values of p and the current vertex of $L_{(\tau,p)}$ is the first vertex of $L_{(\tau,p)}$. $\tau \equiv \mathbf{0}$ is a topological numbering for the initial Δ of procedure Refine since Δ is 0-optimal by Lemma 4.3.2 and thus there is no admissible arc in $\mathcal{N}_\Delta = (G_\Delta = (V, A_\Delta), c_\Delta, \gamma_\Delta)$.

In the following we make refinements on the order of the push operations to be carried out. Let $\pi : V \rightarrow \{1, 2, ..., n\}$ be a one-to-one mapping, i.e., a numbering of the vertices in V. Based on this π, we define a list L_π on V by the rule:

$$v \text{ is before } w \text{ on } L_\pi \iff \pi(v) > \pi(w) \tag{4.4.54}$$

for any $v, w \in V$. For a fixed u in V, we define a list $List_\pi(u)$ on the ordered pair set $\{(u, v) \mid v \in V - \{u\}\}$ by

$$(u, v) \text{ is before } (u, w) \text{ on } List_\pi(u) \Longleftrightarrow v \text{ is before } w \text{ on } L_\pi \quad (v,\ w \neq u). \tag{4.4.55}$$

In $List_\pi(u)$ there is an element called the current arc of u. Initially, the current arc of u is the first element in $List_\pi(u)$. We should point out that list $L_{(\tau,p)}$ is used in selecting active vertices and has no relation to the definition of current arcs. We give an implementation of procedure Refine which is called the first-active implementation.

Procedure Refine(ε, L, p) (first-active implementation).

Input: ε, $L = 18mU + 1$, and p such that there exists a 2ε-optimal submodular flow φ with respect to p.

Output: A potential p and an ε-optimal submodular flow φ with respect to p.

Step 0: For the current p, find an integer vector z_0 in $\mathrm{B}(f)$ such that

$$\sum_{v \in V} p(v) z_0(v) = \max_{z' \in \mathrm{B}(f)} \sum_{v \in V} p(v) z'(v). \tag{4.4.56}$$

Put $z \leftarrow z_0$. For each $(v, w) \in A$, if $\gamma_{\Delta,p}(v, w) < 0$ then put
$\varphi(v, w) \leftarrow \bar{c}(v, w)$,
otherwise put
$\varphi(v, w) \leftarrow \underline{c}(v, w)$.
Put $\Delta \leftarrow (\varphi, z)$.

Step 1: Let $L_{(\tau,p)}$ be the list of V and v be the current vertex. If v is an active vertex and relabeled less than L times, go to Step 2. If v is not the last vertex of $L_{(\tau,p)}$, replace v as the current vertex by the vertex right after v on $L_{(\tau,p)}$ and go to the beginning of Step 1. If v is the last vertex of $L_{(\tau,p)}$, then output the current potential p and ε-optimal submodular flow φ with respect to p and stop.

Step 2: Let (v, w) be the current arc of v.

(2-1) Applicability: $\gamma_{\Delta,p}(v, w) < 0$ and $(v, w) \in A_\varphi \cup B_\varphi$;
Perform Push1(v, w). Go to Step 1.

(2-2) Applicability: $\gamma_{\Delta,p}(v, w) = -\varepsilon$ and $(v, w) \in C_z$ is strongly admissible;
Perform Push2(v, w). Go to Step 1.

(2-3) Applicability: $(v, w) \notin A_\Delta$, or (v, w) is not strongly admissible and not the last element in $List_\pi(v)$;

Replace the current arc (v, w) by (v, u) which is next to (v, w) in $List_\pi(v)$. Go to the beginning of Step 2.

(2-4) Applicability: (v, w) is the last element in $List_\pi(v)$, $(v, w) \notin A_\Delta$ or (v, w) is not strongly admissible, and there is at least one admissible arc in $List_\pi(v)$;

Put $\tau(v) = \max\{\tau(w) \mid (v, w) \in C_z$ is admissible$\}$ and let (v, w) be the current arc of v where (v, w) is the first admissible arc in $List_\pi(v)$ achieving the value $\max\{\tau(u) \mid (v, u) \in C_z$ is admissible$\}$ with $u = w$. Put v at the right position in $L_{(\tau,p)}$ based on the new values of $\tau(v)$ and $p(v)$. If this changes the position of v, then let the vertex next to v before the replacement be the current vertex and go to Step 1. Otherwise, perform Push2(v, w) and go to Step 1.

(2-5) Applicability: (v, w) is the last element in $List_\pi(v)$ and there is no admissible arc in $List_\pi(v)$;

$p(v) \leftarrow p(v) - \varepsilon$, $T \leftarrow T + 1$ and $\tau(v) \leftarrow T$. Place v at the beginning vertex of $L_{(\tau,p)}$ and let it be the current vertex. Go to Step 1.

(End)

Lemma 4.4.8: *Throughout the algorithm the following property* $(*)$ *is maintained:*
$(*)$ *For each* $v \in V$ *and the current arc* (v, w), *any arc* $(v, u) \in A_\Delta$ *lying before* (v, w) *in* $List_\pi(v)$ *is not a strongly admissible arc.*

Proof: Suppose that currently $(*)$ holds and that the next basic operation is a relabeling operation for a vertex u_0. This operation does not generate any new arc. Denote the current potential function by p, and that after the operation by p'. Note that $p'(w) \leq p(w)$ $(w \in V)$. For u_0, the current arc for it is made to be the first element in $List_\pi(u_0)$. Furthermore, for any other v $(v \neq u_0)$ let (v, w) be the current arc of v and $(v, u) \in A_\Delta$ be the predecessor of (v, w). We have

$$\gamma_{\Delta,p'}(v, u) = c_\Delta(v, u) + p'(v) - p'(u) \geq c_\Delta(v, u) + p(v) - p(u). \tag{4.4.57}$$

From (4.4.57) and since τ is not changed for $u \neq u_0$, (v, u) is still not a strongly admissible arc. If $u = u_0$, we have (v, u_0) is not admissible since $p(u_0)$ is reduced by ε.

Next, suppose that currently $(*)$ holds and that the next basic operation is a push for the current arc $(u, v) \in C_z$ where (φ, z) is the current submodular pseudo-flow.

We denote the submodular pseudo-flow after the push operation by (φ, z'). Note that potential p is not changed by the push. Therefore, it is sufficient to show that after the push operation any new arc $(w,s) \in C_{z'}$ placed before the current arc in $List_\pi(w)$ is not a strongly admissible arc. By Lemma 1.4.12, we have i) $u = s$ or $(u,s) \in C_z$ and ii) $v = w$ or $(w,v) \in C_z$.

If $u \neq s$ and $w \neq v$. Suppose on the contrary that $p(w)-p(s) = -\varepsilon$ and $\tau(w) = \tau(s)$. It is showed in the proof of Lemma 4.4.7 that (u,s) and (w,v) are all strongly admissible arcs. It follows that v is before s in L_π. Hence (w,s) is after (w,v). By the induction assumption, (w,v) is before the current arc of w and (w,v) is not a strongly admissible arc before the push. It is a contradiction.

If $u = s$ and $w \neq v$, then $p(w) \geq p(v) - \varepsilon = p(u) = p(s)$.

If $u \neq s$ and $w = v$, then $p(w) = p(v) = p(u) + \varepsilon \geq p(s)$.

If $u = s$ and $w = v$, then $p(w) - p(s) = p(v) - p(u) = \varepsilon > 0$.

Hence, $(*)$ holds after the push operation of type Push2.

For the push operation on (u,v) of type Push1, the only new arc is (v,u) which is not an admissible arc.

Finally, we note that the replacements in (2-4) of Step 2 maintain property $(*)$. □

In the beginning of (2-4) of Step 2, there is at least one admissible arc in $List_\pi(v)$ and the current arc is the last element of $List_\pi(v)$. From Lemma 4.4.8, there exists no strongly admissible arc in $List_\pi(v)$. Since each admissible arc in $A_\varphi \cup B_\varphi$ is a strongly admissible arc. Consequently, in $List_\pi(v)$ there is no admissible arc in $A_\varphi \cup B_\varphi$ and there is at least one admissible arc in C_z. Hence, (2-4) of Step 2 is well defined. We note that when entering Step 2, one and only one of Step (2-1)∼ Step (2-5) is applicable.

Lemma 4.4.9: *Procedure Refine terminates with an ε-optimal submodular flow φ with respect to the obtained potential function p.*

Proof: From the property of $L_{(\tau,p)}$ and the last vertex on $L_{(\tau,p)}$ is inactive at the termination, there is no active vertex in V. The present lemma follows from Lemmas 4.3.2∼4.3.4. □

Definition 4.4.10: A push on $a = (v, w) \in A_\Delta$ is called a *saturating push* if $e(v) \geq c_\Delta(a)$. If a push on $a = (v, w) \in A_\Delta$ is not a saturating push, then it is called a *nonsaturating push*. □

Define a *pass* over $L_{(\tau,p)}$ as a period of the algorithm that begins with the first vertex on $L_{(\tau,p)}$ and ends when a relabeling is performed or when the algorithm terminates.

Lemma 4.4.11: *During a procedure Refine there are at most $O(nmU)$ passes before the algorithm terminates.*

Proof: Since there is a relabeling operation in each pass except the last one, the present lemma follows from Theorem 4.4.5. □

Lemma 4.4.12: *The number of nonsaturating pushes during a procedure Refine (first-active implementation) is at most* $\mathrm{O}(n^2mU)$.

Proof: In each pass, a nonsaturating push on v makes v an inactive vertex and the current vertex shifts. So, there are at most $n - 1$ nonsaturating pushes in each pass. □

Lemma 4.4.13: *The number of saturating push operations during a procedure Refine (first-active implementation) is at most* $\mathrm{O}(n^3mU)$.

Proof: Note that after an execution of (2-4) of Step 2 the current arc of v is shifted backward. Between two successive relabelings on v, (2-4) of Step 2 is carried out at most $n-1$ times since the number of the different values of $\tau(w)$ ($w \in V$, $\tau(w) < \tau(v)$) is less than n before the first execution of (2-4) of Step 2 and each execution of (2-4) of Step 2 reduces the number of the different values of $\tau(w)$ ($w \in V$, $\tau(w) < \tau(v)$) by at least one. By a saturating push on an arc in $List_\pi(v)$ the current arc shifts to the next arc. From Lemma 4.4.8 we see that between two successive relabelings on v there are at most $2n(n-1)$ saturating pushes on arcs going out from v. It follows form Theorem 4.4.5 that there are at most $\mathrm{O}(n^2mU)$ saturating pushes for each vertex v during a procedure Refine . The present lemma follows from the fact that the cardinality of V is n. □

Finally, we have

Theorem 4.4.14: *The complexity of the minimum-cost submodular flow algorithm by the first-active implementation is* $\mathrm{O}(n^3 mU \log(n\Gamma))$ *with the oracles for dependence functions and exchange capacities of the given submodular system.* □

Chapter 5.

Conclusion

In this final chapter we discuss some future topics in theories and algorithms, summarize the computational complexity of some existing algorithms and make some comparisons of our proposed algorithms with other ones from the point of view of theoretical and computational aspects.

Submodular (or supermodular) functions on distributive lattices share similar structures with convex (or concave) functions on convex sets. The reason for the analogy between a submodular function and a convex function is nicely explained by the Lovász extension of a submodular function. Such an analogy has been recognized as a reason of the importance of submodular functions in the analysis of combinatorial systems such as graphs, networks and algebraic systems. Applications of matroids and submodular systems to practical engineering problems and information and system theories have been extensively made by M. Iri, A, Recski, S. Fujishige, K. Murota and others. Recently, uses of submodular systems and their extensions in scheduling problems have been extensively studied. The theory of matroids and submodular systems provides us a systematic mathematical foundation which enables us to better recognize the essential structures of the fields of applications. We believe that there will be more and more applications of matroids and submodular systems since these reveal combinatorially nice and deep structures of combinatorial systems. Also, such applications would require further developments in constructing efficient and practical algorithms. Our algorithms provide a general basis for applications.

The algorithms in this thesis are designed for a larger class of problems described by abstract structures of matroids or submodular systems. However, if the algorithms are to be applied to some specific problems such as the analysis/design of very large systems in physical and engineering applications, the algorithm should be refined to adopt to the characteristic features of such systems. In other words, our algorithms are likely to be less efficient for a narrower class if the refinements are not made, but they give a general basis for devising efficient ones for specific problems. Such refinements are of practical importance. The difference between our algorithms and others in performance for specific problems should be studied. Computationally, the structure of our algorithm is new and quite different from those of the existing previous algorithms. In this thesis, we give time complexities on the basis of some oracles. In applying our algorithms to practical problems we must be very careful in constructing a subroutine for an oracle. The following three are the most crucial for the practical efficiency: (1) what kind of oracle to adopt, (2) how to represent a matroid or a submodular system and (3) how to correlate with the representations in different stages of the solution process. What kind of oracles to adopt in our algorithms in order to improve the practical efficiency is one of the important future studies.

In the following we examine the difference between our algorithm for the independent assignment problem and other existing algorithms. There are two algorithms with computational complexities similar to ours, one proposed by H. N. Gabow and Y. Xu [40] and the other recently by M. Shigeno and S. Iwata [74]. Both are given for the weighted matroid intersection problem. H. N. Gabow and Y. Xu uses A. Frank's version of linear programming dual problem for the weighted matroid intersection problem [21]. The algorithm also makes scaling on weights. Each scaling adds one bit of precision to the original weights and finds a 1-optimal solution for these weights. The algorithm for a specified scale consists of two steps: the Hungarian search step and the augmenting step. The former step changes the values of dual variables (potentials) so as to make the admissible graph (consisting of only admissible arcs) have augmenting paths. The latter step consists of a sequence of augmentations. The augmenting paths used in the augmenting step are not required to be a kind of shortest paths. To make an augmenting path a valid one, they assign a topological numbering to the vertex set and require the numbering in the path satisfying certain conditions. To perform the Hungarian search step and the augmenting step, they use several lists of size $O(n)$ for each vertex. It is necessary in the algorithm to store the auxiliary network for improving complexity. Besides making modifications on the potential for each vertex, the algorithm keeps a topological numbering on the vertex set which also changes frequently.

Our algorithm is similar to the algorithm of H. N. Gabow and Y. Xu in the sense that both algorithms are based on the cost scaling. Computationally, our algorithm is quite different from their algorithm. In our algorithm we need two lists for the source vertex set and sink vertex set. Storing the whole auxiliary network is not necessary since for each local operation related to a vertex v, we only need to identify arcs which are incident to v. Therefore, H. N. Gabow and Y. Xu's algorithm uses a space which is at least three times larger than that used in our algorithm, disregarding the space required by the oracles.

The approximate-weight-splitting algorithm of M. Shigeno and S. Iwata uses a different optimality condition. The weight vector w is split into any two weight vectors u and v such that $w = u + v$. The optimality condition states that if B is a minimum-weight k-independent set of a matroid $\mathcal{M}_1$ on a finite set E with respect to the weight vector u and B is also a minimum-weight k-independent set of a matroid $\mathcal{M}_2$ on the set E with respect to the weight vector v, then B is a minimum-weight common k-independent set of the two matroids $\mathcal{M}_1$ and $\mathcal{M}_2$ with respect to the weight vector w. This condition is an elementary one. The theoretical background of their algorithm is simpler than that of our algorithm. They also only use local operations of relabeling split weights and elementary transformations for an independent set. Hence, the computation structure is almost the same as ours. We will make a comparison only on theoretical complexity in the following.

Both algorithms have been generalized to the minimum-cost 0-1 submodular flow problem (see [75] and [39]). In our generalization, we have shown that the number of relabelings of potential for each vertex is $\mathrm{O}(m)$, where there are n vertices. In their generalization, they proved that the number of relabelings of the split weight for each arc is $\mathrm{O}(m)$, there are m arcs in original graph. Therefore, our algorithm runs in $\mathrm{O}(mn^2 \log(nC))$ time while their algorithm runs in $\mathrm{O}(m^2 n \log(nC))$ time. Both time bounds are for the case where we assume an oracle for exchange capacities of a given submodular system without using a hybrid version. Also, when considering the hybrid versions, in our algorithm we can shift to a procedure SuccessiveShortestPath by choosing $L = \mathrm{O}(\sqrt{m})$ while the optimal selection for their algorithm is $L = \mathrm{O}(\sqrt{n})$.

Therefore, the complexity of our hybrid version becomes $O(\sqrt{m}n^2\log(nC))$, which is lower than the complexity $O(\sqrt{n}nm\log(nC))$ of their hybrid version. As to the case of the independent assignment problem, when their algorithm for the weighted matroid intersection problem is generalized to the independent assignment problem, the algorithm should deal with the split costs defined on arc set A while our potential function is defined on vertex set $V^+ \cup V^-$. We can make a similar argument on the complexity comparison for this case.

H. N. Gabow and Y. Xu in [41] have given an efficient algorithm for the independent assignment problem on graphic matroids with a dynamic tree data structure for the independence oracle. It will be an important and interesting work to find an efficient independence oracle for our hybrid version algorithm for linear or graphic matroid, since the structure of operations in the algorithm is quite different from that of H. N. Gabow and Y. Xu's. To apply our algorithm for the general graph matching problem is also an interesting future work.

There are many combinatorial optimization problems that can not be modeled by matroidal structures. We mention a few new structures as generalizations of a matroid and related concepts. Here, it is worth considering whether the approach of our algorithm is still applicable to such new structures. R. Chandrasekaran and S. N. Kabadi [7] introduced the concept of pseudomatroid. It is a common generalization of a family of independent sets of a matroid, a family of bases of a matroid and a generalized matroid [24]. They also defined a polyhedral version of a pseudomatroid called a polypseudomatroid, which has nice properties such as the total dual integrality of the defining inequalities and the validity of a greedy algorithm. Another generalization of matroid independence systems, called a $(2,2)$-system, is given by R. Euler in [18]. The defined independence system gives a unified formulation of matroid intersection problems, vertex-packing independence systems and b-matchings problems. R. L. Rardin and M. Sudit have introduced a new structure called a paroid. A paroid is formed by a matroid and a partition of the ground finite set. In [69] they show that paroid optimization formulations are suitable for several classical combinatorial problems such as k-matroid intersection, general graph matching, traveling salesman, vertex packing, satisfiability, graph partitioning, and knapsack problems.

In the following we briefly describe a few existing algorithms for minimum-cost submodular flow problems and point out some interesting problems for future research. The first polynomial algorithm for the minimum-cost submodular flow problem of Grötschel, Lovász and Schrijver [46] is based on the ellipsoid method which seems to have only theoretical significance. W. H. Cunningham and A. Frank [11] proposed the first polynomial algorithm which is purely combinatorial. The optimality theorem, Theorem 4.1.2, is fundamental to their algorithm. In the algorithm, a feasible submodular is maintained while we try to modify potentials to satisfy the optimalty conditions in Theorem 4.1.2. Given a feasible submodular flow φ and a potential function p, choose an arc a in $A_\varphi \cup B_\varphi$ with a negative reduced cost, i.e., an arc a violating the optimality conditions. A procedure, called Inner Algorithm in W. H. Cunningham and A. Frank's algorithm, finds a new pair (φ, p) such that the arc a no longer violates the optimality conditions while at the same time the other arcs satisfying the optimality conditions before do not violate the optimality conditions. Inner Algorithm consists of two parts. One performs flow changes only which sends a maximum flow on arc a under certain constraints. Any maximum submodular flow algorithm can work for this purpose. Our algorithm in Chapter 2 can also be used. The other part of Inner Algorithm is used to change the potential, when the changes on the flow fail to make arc a satisfy the optimality conditions. Their algorithm also uses cost scaling and requires in total $O(mn^3 \log(\Gamma))$ path augmentations.

In stead of reducing the discrepancy from the optimality conditions on each arc, one may use the method of canceling negative cycles in an auxiliary network with reduced costs as arc lengths. The first negative cycle method for the independent flow problem is due to S. Fujishige [28]. U. Zimmermann [85] also gave an algorithm for the submodular flow problem by a similar approach. Their algorithms may not terminate in finitely many steps. With minimum-mean cycle selection, W. Cui and S. Fujishige [10] proposed a finite algorithm and U. Zimmermann [86] developed a pseudo-polynomial variant that finds $O(mU)$ negative cycles. With a certain weight function, the minimum-ratio cycle selection method also leads to a pseudo-polynomial algorithm (see C. Wallacher [79]). In [80] C. Wallacher and U. Zimmermann showed that for a large number of weight functions, the cycle canceling method for submodular flow problems terminates after finding $O(mU \log(m\Gamma U))$ negative cycles. They also presented a polynomial variant that solves the problem by finding $O(n^2 m \log(m\Gamma U))$

negative cycles, which is the first polynomial algorithm by canceling negative cycles.

Another algorithm of strongly polynomial complexity is due to S. Fujishige, H. Röck and U. Zimmermann [35]. The algorithm adopts a tree projection approach for the calculation of the current cost approximation. (For the classical minimum-cost flow problem the tree projection technique was given by S. Fujishige [31] and was independently noticed by H. Röck and U. Zimmermann.) For a sequence of approximations of the cost function γ, a minimum-cost submodular flow problem for each approximation cost function γ' is solved by means of the primal-dual method of W. H. Cunningham and A. Frank [11] with or without scaling of γ'. The number of approximations is bounded by $m + n(n-1)$ and each approximation cost function γ' satisfies $|\gamma'(a)| \leq n^2$ for all $a \in A$. Therefore, W. H. Cunningham and A. Frank's algorithm for such cost function runs in strongly polynomial time with or without scaling.

The algorithm given in Chapter 4 is not a polynomial one. The improvement on its complexity seems to be related to a capacity scaling technique and other ideas. To devise an efficient algorithm base on a capacity scaling for submodular flows is still an unsolved problem.

Finally, we mention a few developments in theories related to submodular systems. A concept, called a generalized polymatroid, that is very similar to submodular base polyhedra is defined by A. Frank [24]. S. Fujishige [30] has shown that every generalized polymatroid can be obtained from a base polyhedron by a simple construction. Also, a bisubmodular system is first considered in [7], which is a natural generalization of a polypseudomatroid. The class of polyhedra associated with bisubmodular systems includes submodular polyhedra, base polyhedra, generalized polymatroids [24], and polypseudomatroids [7]. In [3] K. Ando, S. Fujishige and T. Naitoh have defined and studied a subclass of bisubmodular systems called the proper bisubmodular system which also includes base polyhedra, polypseudomatroids and a class of generalized polymatroids. The bisubmodular function in a proper bisubmodular system is defined in a domain which can be represented by a distributive lattice. Also they consider bidirected flows in a bidirected networks as an application of a proper bisubmodular system. To construct efficient algorithm for bidirected flows with possible bisubmodular constraints is an interesting future research subject.

References

[1] R. K. Ahuja and J. B. Orlin: A fast and simple algorithm for the maximum flow problem. *Operations Research* **37** (1990) 748-759.

[2] R. K. Ahuja, T. L. Magnanti and J. B. Orlin: Network Flows. In: *Handbooks in OR & MS* Vol. 1 (G. L. Nemhauser et al., eds., Elsevier Science Publishers B. V., North-Holland, 1989).

[3] K. Ando, S. Fujishige and T. Naitoh: Proper bisubmodular systems and bidirected flows. Discussion Paper No. 532, Institute of Socio-Economic Planning, University of Tsukuba (1993).

[4] D. P. Bertsekas: A distributed algorithm for the assignment problem. Working Paper, Laboratory for Information and Decision Systems, MIT (Cambridge, MA, 1979).

[5] D. P. Bertsekas and J. Eckstein: Dual coordinate step methods for linear network flow problems. *Mathematical Programming* **42** (1988) 203-243.

[6] C. Brezovec, G. Cornuéjols and F. Glover: Two algorithms for weighted matroid intersection. *Mathematical Programming* **36** (1986) 39-53.

[7] R. Chandrasekaran and S. N. Kabadi: Pseudomatroids. *Discrete Mathematics* **71** (1988) 205-217.

[8] J. Cheriyan and T. Hagerup: A randomized maximum-flow algorithm, *Proceedings of the IEEE 30th Annual Symposium on Foundations of Computer Science* (1989) 118-123.

[9] J. Cheriyan and S. N. Maheshwari: Analysis of preflow push algorithms for maximum network flows. *SIAM Journal on Computing* **18** (1989) 1057-1086.

[10] W. Cui and S. Fujishige: A primal algorithm for the submodular flow problem with minimum-mean cycle selection. *Journal of the Operations Research Society of Japan* **31** (1988) 431-440.

[11] W. H. Cunningham and A. Frank: A primal-dual algorithm for submodular flows. *Mathematics of Operations Research* **10** (1985) 251-262.

[12] E. A. Dinits: Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Mathematics* **11** (1970) 1277-1280.

[13] J. Edmonds: Submodular functions, matroids, and certain polyhedra. *Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications* (R. Guy, H. Hanani, N. Sauer and J. Schönheim, eds., Gordon and Breach, New York, 1970), 69-87.

[14] J. Edmonds: Matroids and the greedy algorithm. *Mathematical Programming* **1** (1971) 127-136.

[15] J. Edmonds: Matroid intersection algorithm. *Annals of Discrete Mathematics* **4** (1979) 39-49.

[16] J. Edmonds and R. Giles: A min-max relation for submodular functions on graphs. *Annals of Discrete Mathematics* **1** (1977) 185-204.

[17] J. Edmonds and R. M. Karp: Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of ACM* **19** (1972) 248-264.

[18] R. Euler: Augmenting paths and a class of independence systems. *Bonn Workshop on Combinatorial Optimization* (A. Bachem, M. Grötschel and B. Korte, eds., North Holland, Amsterdam, 1981) 69-82.

[19] L. R. Ford, Jr., and D. R. Fulkerson: Maximal flow through a network. *Canadian Journal of Mathematics* **8** (1956) 399-404.

[20] L. R. Ford, Jr., and D. R. Fulkerson: *Flows in Networks* (Princeton University Press, Princeton, N. J., 1962).

[21] A. Frank: A weighted matroid intersection algorithm. *Journal of Algorithms* **2** (1981) 328-336.

[22] A. Frank: How to make a digraph strongly connected. *Combinatorica* **1** (1981) 145-153.

[23] A. Frank: An algorithm for submodular functions on graphs. *Annals of Discrete Mathematics* **16** (1982) 189-212.

[24] A. Frank: Generalized polymatroids. In :A. Hajnal et. al., eds., Finite and Infinite Sets (North-Holland, Amsterdam-New York, 1984) 285-294.

[25] A. Frank: Augmenting graphs to meet edge-connectivity requirements. *SIAM Journal on Discrete Mathematics* **5** (1992) 25-53.

[26] A. Frank and É. Tardos: An application of submodular flows. *Linear Algebra and its Applications* **114/115** (1989) 329-348.

[27] S. Fujishige: A primal approach to the independent assignment problem. *Journal of the Operations Research Society of Japan* **20** (1977) 1-15.

[28] S. Fujishige: Algorithms for solving the independent-flow problems. *Journal of the Operations Research Society of Japan* **21** (1978) 189-204.

[29] S. Fujishige: Polymatroidal dependence structure of a set of random variables. *Information and Control* **39** (1978) 55-72.

[30] S. Fujishige: A note on Frank's generalized polymatroids. *Discrete Applied Mathematics* **7** (1984) 105-109.

[31] S. Fujishige: A capacity-rounding algorithm for the minimum cost circulation problem– A dual framework of the Tardos algorithm. *Mathematical Programming* **35** (1986) 298-308.

[32] S. Fujishige: From classical flow problems to the "neoflow" problem (in Japanese). *Transaction of the Institute of Electronics, Information and Communication Engineers of Japan* **J70-A** 139-145.

[33] S. Fujishige: An out-of-kilter method for submodular flows. *Discrete Applied Mathematics* **17** (1987) 3-16.

[34] S. Fujishige: *Submodular Functions and Optimization* (North-Holland, Amsterdam, 1991).

[35] S. Fujishige, H. Röck and U. Zimmermann: A strongly polynomial algorithm for minimum cost submodular flow problems. *Mathematics of Operations Research* **14** (1989) 60-69.

[36] S. Fujishige and N. Tomizawa: A note on submodular functions on distributive lattices. *Journal of the Operations Research Society of Japan* **26** (1983) 309-318.

[37] S. Fujishige and X. Zhang: New algorithms for the intersection problem of submodular systems. *Japan Journal of Industrial and Applied Mathematics* **9** (1992) 369-382.

[38] S. Fujishige and X. Zhang: An efficient cost scaling algorithm for the independent assignment problem. *Journal of the Operations Research Society of Japan* (to appear).

[39] S. Fujishige and X. Zhang: A cost scaling framework for submodular flows and its refinement for 0-1 submodular flows. Manuscript (1994).

[40] H. N. Gabow and Y. Xu: Efficient theoretic and practical algorithms for linear matroid intersection problems. Technical Report No. CU-CS-424-89, Computer Science Department, University of Colorado, (1989).

[41] H. N. Gabow and Y. Xu: Efficient algorithms for independent assignment on graphic Matroids. Technical Report No. CU-CS-468-90, Computer Science Department, University of Colorado, (1990).

[42] A. V. Goldberg: Processor-efficient implementation of a maximum flow algorithm. *Information Processing Letters* (1991) 179-185.

[43] A. V. Goldberg and R. E. Tarjan: A new approach to the maximum flow problem. *Journal of the Association for Computing Machinery* **35** (1988) 921-940.

[44] A. V. Goldberg and R. E. Tarjan: Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research* **15** (1990) 430-466.

[45] M. Grötschel, L. Lovász and A. Schrijver: The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* **1** (1981) 169-197.

[46] M. Grötschel, L. Lovász and A. Schrijver: *Geometric Algorithms and Combinatorial Optimization* (Algorithms and Combinatorics **2**) (Springer, Berlin, 1988).

[47] T. S. Han: The capacity region of general multiple-access channel with certain correlated sources. *Information and Control* **40** (1979) 37-80.

[48] R. Hassin: Minimum-cost flow with set-constraints. *Networks* **12** (1982) 1-21.

[49] R. Hassin: Algorithms for the minimum-cost circulations problem based on maximizing the mean improvement. Tel Aviv University Statistics Department Working Paper, Tel Aviv, Israel, (1991).

[50] M. Iri: Applications of matroid theory. *Mathematical Programming — The State of the Art* (A. Bachem, M. Grötschel and B. Korte, eds., Springer, Berlin, 1983), 158-201.

[51] M. Iri and S. Fujishige: Use of matroid theory in operations research, circuits and system theory. *International Journal of Systems Science* **12** (1981) 27-54.

[52] M. Iri, S. Fujishige and T. Oyama: *Graphs, Network and Matroids* (in Japanese) (Sangyo-Tosho ,Tokyo, 1986).

[53] M. Iri and N. Tomizawa: An algorithm for finding an optimal "independent assignment". *Journal of the Operations Research Society of Japan* **19** (1976) 32-57.

[54] D. S. Johnson and C. C. McGeoch (eds.): *Network Flows and Matching: First DIMACS Implementation Challenge* (DIMACS Series in Discrete Mathematics and Theoretical Computer Science Vol. 12) (American Mathematics Society, Princeton, N. J., 1992).

[55] A. V. Karzanov: Determining the maximal flow in a network by the method of preflows. *Soviet Mathematics Doklady* **15** (1974) 434-437.

[56] M. Klein: A primal method for minimal cost flows with applications to the assignment and transportation problems.. *Management Science* **14** (1967) 205-220.

[57] J. B. Kruskal: On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society* **7** (1956) 48-50.

[58] E. L. Lawler: Matroid intersection algorithms. *Mathematical Programming* **9** (1975) 31-56.

[59] E. L. Lawler: *Combinatorial Optimization — Networks and Matroids* (Holt, Rinehart and Winston, New York, 1976).

[60] E. L. Lawler and C. U. Martel: Computing maximal polymatroidal network flows. *Mathematics of Operations Research* **7** (1982) 334-347.

[61] E. L. Lawler and C. U. Martel: Flow network formulations of polymatroid optimization problems. *Annals of Discrete Mathematics* **16** 189-200.

[62] C. Lucchesi and D. H. Younger: A minimax relation for directed graphs. *Journal of the London Mathematical Society* **17** (1978) 369-374.

[63] C. U. Martel: Preemptive scheduling with release times, deadlines, and due times. *Journal of the Association for Computing Machinery* **29** (1982) 812-829.

[64] K. Murota: *Systems Analysis by Graphs and Matroids — Structural Solvability and Controllability* (Algorithms and Combinatorics, Vol. 3, Springer, Berlin, 1987).

[65] C. St. J. A. Nash-Williams: Edge-disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society* **36** (1961) 445-450.

[66] J. B. Orlin and R. K. Ahuja: New scaling algorithms for the assignment and minimum mean cycle problems. *Mathematical Programming* **54** (1992) 41-56.

[67] J. B. Orlin and J. Vande Vate: On a 'primal' matroid intersection algorithm. Sloan School Technical Report, MIT (Cambridge, 1984).

[68] R. Rado: Note on independence functions. *Proceedings of the London Mathematical Society* **7** (1957) 300-320.

[69] R. L. Rardin and M. Sudit: Paroids: a canonical format for combinatorial optimization. *Discrete Applied Mathematics* **39** (1992) 37-56.

[70] A. Recski: *Matroid Theory and Applications in Electric Network Theory and in Statics* (Algorithms and Combinatorics, Vol. 3, Springer, Berlin, 1989).

[71] H. Róck: Scaling techniques for minimal cost network flows. In:*Discrete Structures and Algorithms* (U. Papa, ed., Hanser, München, 1980), 181-191.

[72] P. Schönsleben: Ganzzahlige Polymatroid-Intresektions-Algorithmen, Dissertation, Eidgenössische Technisch Hochschule Zürich, 1980.

[73] L. S. Shapley: Cores of convex games. *International Journal of Game Theory* **1** (1971) 11-26.

[74] M. Shigeno and S. Iwata: Approximate-weight-splitting algorithm for a minimum common base of a pair of matroids. Research Report No. B-278, Department of Information Sciences, Tokyo Institute of Technology (1993).

[75] M. Shigeno and S. Iwata: Approximate-cost-splitting algorithm for minimum cost 0-1 submodular flow problems. International Symposium on Mathematical Programming (1994).

[76] É. Tardos: A strongly polynomial minimum cost circulation algorithm. *Combinatorica* **5** (1985) 247-256.

[77] É. Tardos, C. A. Tovey and M. A. Trick: Layered augmenting path algorithms. *Mathematics of Operations Research* **11** (1986) 362-370.

[78] B. L. Van der Waerden: *Moderne Algebra* (second edition) (Springer, Berlin, 1937).

[79] C. Wallacher: A generalization of the minimum-mean cycle selection rule in cycle canceling algorithms. Report, Inst. für Ang. Matht., Braunschweig, (1991).

[80] C. Wallacher and U. Zimmermann: A polynomial cycle canceling algorithm for submodular flows. Discussion paper (1994)

[81] D. J. A. Welsh: On matroid theorems of Edmonds and Rado. *Journal of the London Mathematical Society* **2** (1970) 251-256.

[82] D. J. A. Welsh: *Matroid Theory* (Academic Press, London, 1976).

[83] H. Whitney: On the abstract properties of linear dependence. *American Journal of Mathematics* **57** (1935) 509-533.

[84] X. Zhang: A cost scaling algorithm for minimum-cost submodular flows. Manuscript (1994).

[85] U. Zimmermann: Minimization on submodular flows. *Discrete Applied Mathematics* 4 (1982) 303-323.

[86] U. Zimmermann: Negative circuits for flows and submodular flows. *Discrete Applied Mathematics* **36** (1992) 179-189.