RIMS Kôkyûroku Bessatsu
B23 (2010), 63–79

Neighbor Systems and the Greedy Algorithm (Extended Abstract)

By

David HARTVIGSEN*

Abstract

A *neighbor system*, introduced in this paper, is a collection of integral vectors in $\Re^n$ with some special structure. Such collections (slightly) generalize jump systems, which, in turn, generalize integral bisubmodular polyhedra, integral polymatroids, delta-matroids, matroids, and other structures. We show that neighbor systems provide a systematic and simple way to characterize these structures. A main result of the paper is a simple greedy algorithm for optimizing over (finite) neighbor systems starting from any feasible vector. The algorithm is (essentially) identical to the usual greedy algorithm on matroids and integral polymatroids when the starting vector is zero. But in all other cases, from matroids through jump systems, it appears to be a new greedy algorithm.

§ 1. Introduction

This paper introduces a new structure, called a *neighbor system*, over which a straightforward greedy algorithm always finds an optimal solution. This system generalizes a variety of structures (matroids and generalizations of matroids) that have been developed since the 1930s and gives a new, standardized way of defining them. This paper is an extended abstract/excerpt of the full version of the paper; in particular, all non-trivial proofs have been removed. The full version will appear elsewhere. Before discussing our results in more detail, let us briefly review some key structures and results from the literature. (Except where noted, the greedy algorithms discussed below optimize these structures over linear objective functions.)

Matroids. Hassler Whitney in 1935 [27] introduced both the structure of matroids and the basic greedy algorithm for optimizing over them.

Generalized matroids. This structure was introduced by Tardos [26], along with a greedy algorithm, as a generalization of matroids.

Received September 15, 2008. Revised June 12, 2009.
2000 Mathematics Subject Classification(s): 90C27, 05B35
*Mendoza College of Business, University of Notre Dame, Notre Dame, IN, USA.

Integral polymatroids. Edmonds [11] introduced this structure as a generalization of matroids together with a greedy algorithm.

Delta-matroids. This structure was introduced, with slight variations, by Bouchet [4], Dress and Havel [9], and by Chandrasekaran and Kabadi [7]. Greedy algorithms were also presented in these papers. Delta-matroids generalize generalized matroids. An interesting example of a delta-matroid is the collection of node sets of the subgraphs of a graph that are perfectly matchable. This collection and the associated optimization problem were first studied by Balas and Pulleyblank [3] on bipartite graphs. This example was further studied on general graphs by Bouchet [5].

Bisubmodular polyhedra. This structure was introduced by Dunstan and Welsh [10] along with a greedy algorithm. Bisubmodular polyhedra are a common generalization of integral polymatroids and delta-matroids. Closely related structures were also studied by Chandrasekaran and Kabadi [7], Nakamura [22], and Qi [23]. A different greedy algorithm for bisubmodular polyhedra was also presented in Ando, Fujishige, and Naitoh [1]; this algorithm also works for the more general class of separable convex objective functions.

Jump systems. Bouchet and Cunningham [6] introduced this structure along with a greedy algorithm. (See also [20] and [16].) A jump system is a collection of integral vectors in $\Re^n$ with some special structure and is a generalization of bisubmodular polyhedra. An important example is the set of degree sequences of the subgraphs of a graph. An interesting fact about jump systems, which sets them apart from the above matroid generalizations, is that a jump system need not be equal to all the integral vectors in its convex hull; hence jump systems may have small "holes" in them. Another greedy algorithm for jump systems was discovered by Ando, Fujishige, and Naitoh [2]; this algorithm also works for the more general class of separable convex objective functions. A variation on this greedy algorithm, which also works for separable convex objective functions, was discovered by Shioura and Tanaka [25]; it has better (that is, polynomial) worst case complexity than the algorithm in [2]. (A further discussion appears below.)

In this paper we introduce a structure called *neighbor systems*. Neighbor systems are sets of integral vectors in $\Re^n$; they are slightly more general than jump systems, hence, they properly include all the examples above. A key feature in the definition of neighbor systems is the *neighbor function*. This notion allows us to systematically categorize, in a new way, the variety of matroid generalizations discussed above. It also allows us to state a greedy algorithm as a straightforward type of neighborhood search. An interesting property of neighbor systems is that they allow "holes" that are arbitrarily large. A main result of this paper is a greedy algorithm for optimizing linear objective functions over neighbor systems starting from any feasible vector. The algorithm is (essentially) identical to the usual greedy algorithm on matroids and integral polymatroids when the starting vector is zero. But in all other cases, from matroids through jump systems, it appears to be a new greedy algorithm. Furthermore, the validity of the algorithm is proved in a completely different way from related greedy algorithms. It is not known if our algorithm works for more general objective functions, such as the separable convex functions.

Another contribution of this paper is a simple characterization of neighbor systems in terms of linear algebra (i.e., cones). This theorem is used to prove the validity of the algorithm and provides an avenue for defining more general structures over which essentially the same greedy algorithm works (see Section 8). The theorem also provides a new way of defining the less general structures discussed above.

Let us discuss greedy algorithms in a bit more detail. Greedy algorithms for matroids and their generalizations have two basic forms in the literature. One type of algorithm *builds up* an optimal solution, one component at a time, using an oracle that (essentially) checks if a given partial vector is contained in (or can be completed to) a feasible solution. The algorithms for delta-matroids in [4], [9], and [7], and for integral bisubmodular polyhedra in [10], as well as the algorithms in [6] and [16] for jump systems, work in this way. The second type of algorithm moves in an *incremental* fashion from feasible solution to feasible solution making calls to a weaker oracle that checks only if a given vector is feasible. The standard greedy algorithm for matroids and the algorithm for integral polymatroids are of this type (where they begin from the origin). The algorithm in [26] for generalized matroids and the algorithms in [1], [2], and [25] for integral bisubmodular polyhedra and jump systems are also of this type (where they begin from an arbitrary feasible vector). The algorithm presented in this paper is of the incremental type and can begin from an arbitrary feasible vector.

Finally, let us mention some closely related work in the literature. Federgruen and Groenevelt [12] presented an incremental-style algorithm, that starts from the origin, for optimizing *weakly concave* functions over integral polymatroids and Groenevelt [17] presented a similar-style algorithm for optimizing separable concave functions over integral polymatroids. Murota [21] presented an incremental-style algorithm for optimizing M-convex functions over constant-parity jump systems. Frank [13] introduced generalized polymatroids together with a greedy algorithm (see also [14]). Shenmaier [24] has studied greedy algorithms over a generalization of matroids called *accessible vector systems*. Another generalization of matroids called *greedoids* is surveyed in Korte et al [18]. An excellent survey of matoid generalizations can be found in the book of Fujishige [15].

The paper is organized as follows. Section 2 contains the definition of neighbor systems. Section 3 contains a variety of examples of neighbor systems and presents a systematic characterization of the matroid generalizations discussed above in terms of neighbor systems. Section 4 contains the statement of the new greedy algorithm. Section 5 contains a characterization of neighbor systems in terms of linear algebra. Section 6 contains a discussion of the complexity of the algorithm and Section 7 contains a more efficient way to implement the algorithm. Section 8 discusses a way that neighbor systems can be further generalized.

§ 2. Definitions

In this section we present the definition of neighbor systems. In the next section we present a number of examples and compare neighbor systems to some of the related matroid generalizations that were referenced in the introduction.

Throughout the paper we let E denote a nonempty finite set; we let $\mathbf{Z}^E$ denote the integral vectors indexed by E; and we let $\mathcal{F}$ denote an arbitrary nonempty finite set of vectors in $\mathbf{Z}^E$. For $x, y, z \in \mathbf{Z}^E$, we say that z is *between* x and y if, for each $i \in E$, $x(i) \le z(i) \le y(i)$ or $x(i) \ge z(i) \ge y(i)$. A *direction* is a $\{0, \pm 1\}$-vector in $\mathbf{Z}^E$ with exactly one or two nonzero components.

For $x, y \in \mathcal{F}$, we say that y is a *neighbor* of x if there exists a direction d and a positive integer b such that $y = x + bd$ and there exists no positive integer $b' < b$ such that $x + b'd \in \mathcal{F}$. Roughly speaking, y is a neighbor of x if y is the "closest" vector of $\mathcal{F} \backslash x$ to x in some direction d.

A *neighbor function*, denoted by N, is a function that takes as input any set $\mathcal{F}$ with any $x \in \mathcal{F}$ and outputs a subset of the neighbors of x in $\mathcal{F}$; the output is denoted $N(\mathcal{F}, x)$.

Definition 2.1. For N a neighbor function, a set $\mathcal{F}$ is called an N-*neighbor system* if it satisfies the following condition:

- For every pair $x, y \in \mathcal{F}$, and for every $i \in E$, where $x(i) \neq y(i)$, there exists $z \in N(\mathcal{F}, x)$ such that z is between x and y, and $z(i) \neq x(i)$.

An N-neighbor system $\mathcal{F}$ is called *finite* if $\mathcal{F}$ is finite.

The idea is the following: If $\mathcal{F}$ is an N-neighbor system, then, for every pair of vectors x and y in $\mathcal{F}$, and every component on which x and y differ, there is a vector in $N(\mathcal{F}, x)$ that is between x and y and is not equal to x on this component. Notice that any choice of N partitions the collection of all sets of vectors in $\mathbf{Z}^E$ into two collections: the N-neighbor systems and the rest.

In the next section we consider a number of examples of neighbor systems, but let us mention here one important example. For every $\mathcal{F}$ and $x \in \mathcal{F}$, let $N^a(\mathcal{F}, x)$ be *all* the neighbors of x in $\mathcal{F}$. We refer to an N^a-neighbor system as an *all-neighbor system.*

Then, for any neighbor function N, the collection of N-neighbor systems is contained in the collection of all-neighbor systems. We will show that the collection of jump systems is properly contained in the collection of all-neighbor systems. Hence the notion of neighbor systems properly generalizes essentially all of the examples discussed in the introduction.

A main concern of the paper is the following optimization problem: Given a finite N-neighbor system $\mathcal{F}$ and a linear objective function $w \in \Re^E$, find a vector $x \in \mathcal{F}$ such that wx is a maximum. One of our main results is a greedy algorithm for solving this problem starting from any vector in $\mathcal{F}$.

Roughly speaking, our greedy algorithm starts by putting the directions into decreasing order by their "slope," which depends on w. The algorithm then considers the directions in this order and moves from vector to vector in the current direction, within the neighborhoods, as far as possible.

The concept of neighbors is useful in two ways. First, it yields a standardized means of defining and comparing various matroid generalizations (see Section 3); and second, it helps in analyzing the complexity of the greedy algorithm (see Sections 6 and 7).

§ 3. Examples of neighbor Systems

In this section we present some examples of neighbor systems. In so doing, a series of propositions is presented that systematically characterizes a number of well-known matroid generalizations in terms of neighbor systems. Finally, we show that neighbor systems are slightly more general than jump systems.

Before listing our examples, let us make the following definition of a *distance function d*: For $x, y \in \mathcal{F}$, let $d(x,y) \equiv \sum_{i \in E} |x(i) - y(i)|$. This yields the following simple neighbor function, for any positive integer k:

$$N_k(\mathcal{F}, x) = \{\text{neighbors } y \text{ of } x \text{ in } \mathcal{F} : d(x,y) \le k\}.$$

Distance-1 neighbor systems: $\mathcal{F}$ is an N_1-neighbor system if and only if it is the integral vectors x that satisfy a system of inequalities of the form: for each $i \in E$, $a(i) \le x(i) \le b(i)$, where $a(i)$ and $b(i)$ are integers.

Distance-2 neighbor systems: $\mathcal{F}$ is an N_2-neighbor system if and only if $\mathcal{F}$ is a jump system. (This example is discussed in more detail below.)

One dimension: Let $|E| = 1$ and let $\mathcal{F}$ be a set of integers $\{x_1, \ldots, x_n\}$, where we assume $x_1 < x_2 < \cdots < x_n$. Then $\mathcal{F}$ is an N-neighbor system if and only if $N(\mathcal{F}, x_1) = \{x_2\}$, $N(\mathcal{F}, x_n) = \{x_{n-1}\}$, and $N(\mathcal{F}, x_i) = \{x_{i-1}, x_{i+1}\}$, otherwise. Thus, $\mathcal{F}$ is an N_k-neighbor system if and only if, for all $x_i, x_{i+1} \in \mathcal{F}$, $x_{i+1} - x_i \le k$.

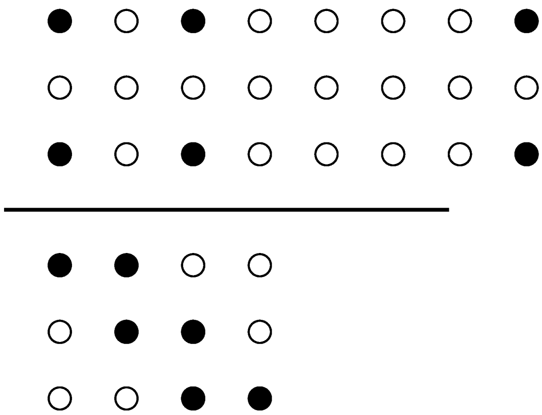

Figure 1. Two examples of 2-dimensional neighbor systems

Two dimensions: Figure 1 contains two examples of all-neighbor systems where $|E| = 2$. The black dots represent grid points in the corresponding set $\mathcal{F}$; white dots represent grid points not in $\mathcal{F}$. Note that if any single white dot in the top example is added to the system, the resulting set of points is no longer an all-neighbor system. For example, suppose we add to $\mathcal{F}$ the dot in the center of the four black dots on the left. If we call this dot y and the lower, far-right dot x, then the pair x, y fails the condition in the definition. Observe that the top example is also an N_k-neighbor system if and only if $k \geq 5$.

The remainder of this section focuses primarily on the examples of neighbor systems mentioned in the introduction. The relationships between the examples are illustrated in Figure 2. An arrow from one box to another means the example at the tail properly generalizes the example at the head. These relationships should be evident from the characterizations in the following propositions.

Matroids: The following proposition shows how matroids are related to neighbor systems.

Proposition 3.1. *For all $\mathcal{F}$ and $x \in \mathcal{F}$, let $N(\mathcal{F}, x)$ be the vectors in $\mathcal{F}$ of the form $x + d$, where d is a direction with one nonzero component or two nonzero components with opposite signs. Then $\mathcal{F} \subseteq \{0,1\}^E$, with $0 \in \mathcal{F}$, is the set of incidence vectors of a matroid if and only if $\mathcal{F}$ is an N-neighbor system.*

Proof. This follows easily from the standard definition of matroids (e.g., see page 268 in Lawler [19]). □

Generalized matroids: These structures were introduced by Tardos [26] (see also Chandrasekaran and Kabadi [7]). From the definition (as given in [7]), it is immediate that generalized matroids are characterized by the above proposition, where the condition $0 \in \mathcal{F}$ is removed and "matroid" is replaced with "generalized matroid."

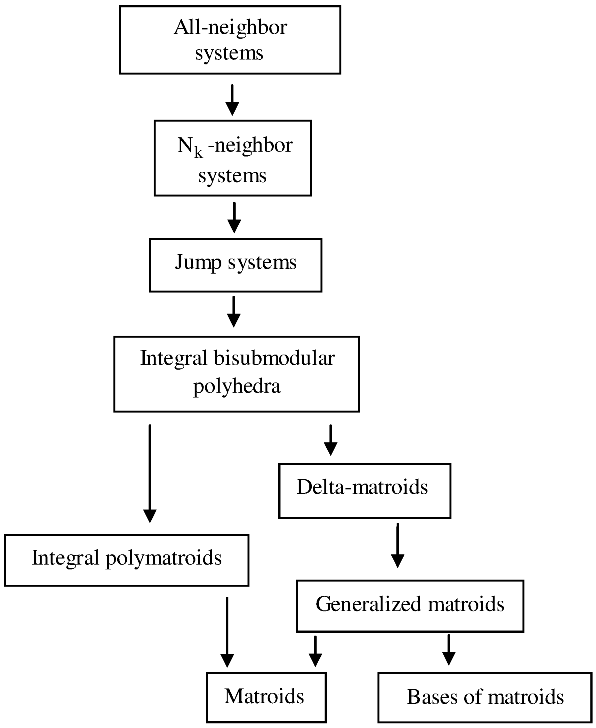

Figure 2. Hierarchy of matroid generalizations

Bases of a matroid: We next consider how the bases of a matroid are related to neighbor systems. A well-known example of the bases of a matroid is the collection of edge sets of spanning forests in a graph.

Proposition 3.2. *For all $\mathcal{F}$ and $x \in \mathcal{F}$, let $N(\mathcal{F}, x)$ be the vectors in $\mathcal{F}$ of the form $x + d$, where d is a direction with two nonzero components with opposite signs. Then $\mathcal{F} \subseteq \{0,1\}^E$ is the set of incidence vectors of the bases of a matroid if and only if $\mathcal{F}$ is an N-neighbor system.*

Proof. This follows immediately from the well-known theorem in matroid theory that says the bases of a matroid are characterized by an exchange property (e.g., see Theorem 5.3 on page 274 in [19]). □

Delta-matroids: Delta-matroids were introduced by Bouchet [4]. An example of a delta-matroid is the collection of node sets of those subgraphs of a graph that are perfectly matchable (see [5]). The following definition appears in [6]:

Definition: Let F be a family of subsets of a finite set E. Then (E, F) is a *delta-matroid* if the following *symmetric exchange axiom* is satisfied:

(SEA) If $F_1, F_2 \in F$ and $j \in F_1 \triangle F_2$, then there is $k \in F_1 \triangle F_2$ such that $F_1 \triangle \{j, k\} \in F$.

Note that $\triangle$ denotes symmetric difference.

Proposition 3.3. *For all $\mathcal{F}$ and $x \in \mathcal{F}$, let $N(\mathcal{F}, x)$ be the vectors in $\mathcal{F}$ of the form $x + d$, where d is a direction. Then $\mathcal{F} \subseteq \{0,1\}^E$ is the set of incidence vectors of a delta-matroid if and only if $\mathcal{F}$ is an N-neighbor system.*

Proof. This follows immediately from the definition of delta-matroids. □

Integral polymatroids: Integral polymatroids were introduced by Edmonds [11]. The standard definition requires the definition of submodular functions and is not needed here. We note that an integral polymatroid can be viewed as a special polyhedron or the integral vectors contained in such a polyhedron. We take the latter view here.

Proposition 3.4. *For all $\mathcal{F}$ and $x \in \mathcal{F}$, let $N(\mathcal{F}, x)$ be the vectors in $\mathcal{F}$ of the form $x + d$, where d is a direction with one nonzero component or two nonzero components with opposite signs. Then $\mathcal{F}$,with $0 \in \mathcal{F}$, is an integral polymatroid if and only if $\mathcal{F}$ is an N-neighbor system.*

Integral bisubmodular polyhedra: Bisubmodular polyhedra were first studied by Dunstan and Welsh [10], Chandrasekaran and Kabadi [7], Nakamura [22], and Qi [23]. The standard definition requires the definition of bisubmodular functions and is not needed here. As with the last example, an integral bisubmodular polyhedron can be viewed as a special polyhedron or as the integral vectors contained in such a polyhedron. We again take the latter view here.

Proposition 3.5. *For all $\mathcal{F}$ and $x \in \mathcal{F}$, let $N(\mathcal{F}, x)$ be the vectors in $\mathcal{F}$ of the form $x + d$, where d is a direction. Then $\mathcal{F}$ is an integral bisubmodular polyhedron if and only if $\mathcal{F}$ is an N-neighbor system.*

Jump systems: Jump systems were introduced by Bouchet and Cunningham [6]. The following definition appears in Ando et al [2].

Definition: Let E denote a nonempty finite set. A *step* is a $\{0, \pm 1\}$-vector in $\mathbf{Z}^E$ with exactly one nonzero component. For any $x, y \in \mathbf{Z}^E$ a *step u from x to y* is a step such that

$$\sum_{e \in E} |x(e) + u(e) - y(e)| = \sum_{e \in E} |x(e) - y(e)| - 1 \tag{3.1}$$

Let $St(x, y)$ denote the set of all steps from x to y. Let $\mathcal{F}$ denote a nonempty set of vectors in $\mathbf{Z}^E$. $(E, \mathcal{F})$ is called a *jump system* if the following *2-step axiom* is satisfied:

(2-SA) For any $x, y \in \mathcal{F}$ and $u \in St(x, y)$ with $x+u \notin \mathcal{F}$, there exists $v \in St(x+u, y)$ such that $x + u + v \in \mathcal{F}$.

An important example of a jump system (which is not an integral bisubmodular polyhedron) is the set of degree sequences of the subgraphs of a graph (see [6]), where loops are allowed and contribute 2 to the degree of a node. A more general class of jump systems using bidirected graphs also appears in [6].

Proposition 3.6. *For all $\mathcal{F}$ and $x \in \mathcal{F}$, let $N(\mathcal{F}, x)$ be the neighbors of x in $\mathcal{F}$ of the form $x + bd$, where d is a direction and if d has exactly one nonzero component, then $b = 1$ or $b = 2$; otherwise, $b = 1$. Then $\mathcal{F}$ is a jump system if and only if $\mathcal{F}$ is an N-neighbor system.*

We can restate the above proposition in the following more compact way.

Proposition 3.7. *$\mathcal{F}$ is a jump system if and only if $\mathcal{F}$ is an N_2-neighbor system.*

Constant parity jump systems: A jump system $\mathcal{F}$ is said to have *constant parity* if, for all $x \in \mathcal{F}$, the summations $\sum_{i \in E} x(i)$ have the same parity (see, e.g., [16]). Here is a simple example: Choose $x \in \mathbf{Z}^E$ and let $\mathcal{F}$ be all the vectors $y \in \mathbf{Z}^E$ such that y is between 0 and x, and $\sum_{i \in E} y(i)$ has the same parity as $\sum_{i \in E} x(i)$. Observe that the degree sequence jump systems defined above provide another example. The following proposition follows easily from Proposition 3.7 for jump systems.

Proposition 3.8. *For all $\mathcal{F}$ and $x \in \mathcal{F}$, let*

$$N(\mathcal{F}, x) = \{\textit{neighbors } y \textit{ of } x \textit{ in } \mathcal{F} : d(x, y) = 2\}.$$

Then $\mathcal{F}$ is a constant parity jump system if and only if $\mathcal{F}$ is an N-neighbor system.

Differences between neighbor systems and jump systems. We show here that neighbor systems are more general than jump systems. However, the difference does not seem to be large and is not the point of this paper.

In Figure 1, both examples are all-neighbor systems and the bottom example is a jump system. However, the top example is not a jump system, since the two points on the far right are too far away from the other points (see Proposition 3.7). Hence we have the following.

Remark: All-neighbor systems are strictly more general than jump systems.

For a general class of examples, let N be a neighbor function; let $\mathcal{F}$ be an N-neighbor system; and let m be an integer. Define $m\mathcal{F} = \{mx : x \in \mathcal{F}\}$ and define the neighbor function mN as follows: $mN(\mathcal{F}, x) = \{my : y \in N(\mathcal{F}, x)\}$. Then it is easy to see that $m\mathcal{F}$ is an mN-neighbor system. Furthermore, if $\mathcal{F}$ is a (non-trivial) jump system and if $|m| \geq 3$, then $m\mathcal{F}$ is not a jump system (see Proposition 3.7). It is also easy to see that the top example in Figure 1 is a neighbor system, say $\mathcal{F}$, and that there is no jump system, say $\mathcal{F}'$, and integer m, such that $\mathcal{F} = m\mathcal{F}'$.

§ 4. The Greedy Algorithm

In this section we present a greedy algorithm for optimizing a linear function $w \in \Re^E$ over a finite neighbor system $\mathcal{F}$ starting from any feasible vector. We assume we have a membership oracle that, in constant time, tells us if any given vector is in $\mathcal{F}$. The main idea is the use of a function w^p (which is analogous to a "slope") for choosing the move to make at any stage.

For each direction d, we define $w^p : \Re^E \to \Re$ as follows:

$$w^p(d) = \frac{wd}{\|d\|}$$

where $\|d\| = \sum_{i \in E} |d_i|$. In other words, $\|d\|$ is the number of nonzero components of d.

We assume, for the remainder of the paper, that w satisfies the following.

A1 For all pairs of distinct directions d' and d'', we have $w^p(d') \neq w^p(d'')$. (Note that it follows, by the definition of directions, that $|w(i)| \neq |w(j)|$, for all $i, j \in E$, where $i \neq j$.)

It is easy to show that there is no loss of generality in making this assumption in the sense that any objective function w can easily be "perturbed" so that A1 is satisfied, without affecting the validity or complexity of the greedy algorithm.

Our greedy algorithm begins by putting the direction vectors into decreasing order by their "slope." The algorithm then considers the direction vectors in this order and tries to improve the current solution by moving in the current direction within the current solution's neighborhood.

Greedy Algorithm :

Input : A finite N-neighbor system $\mathcal{F}$; a vector $x_0 \in \mathcal{F}$; and $w \in \Re^E$.

Output : A vector $x^* \in \mathcal{F}$ that maximizes wx.

Step 0 : Set $x \leftarrow x_0$. Label the input directions so that $w^p(d_1) > w^p(d_2) > \cdots$ and let s be the largest index i for which $w^p(d_i) > 0$.

Step 1 : For $i = 1, \ldots, s$, do the following:

Search $N(\mathcal{F}, x)$ (using the membership oracle) for a vector of the form $x + bd_i$, where b is a positive integer. If such a vector is found, set $x \leftarrow x + bd_i$ and repeat the search.

Step 2 : Set $x^* \leftarrow x$ and output x^*.

End .

Remark: There must exist an s as described in Step 0 by assumption A1; that is, there must exist a direction d such that $w^p(d) > 0$.

Other, simpler, greedy-type algorithms can be stated for this problem. For example, we could drop the labeling operation in Step 0 and replace Step 1 with one of the following variants:

1. Search in $N(\mathcal{F}, x)$ for any vector with a bigger objective value; make it the current x.
2. Search in $N(\mathcal{F}, x)$ for a vector that improves the objective value the most; make it the current x.
3. Search in $N(\mathcal{F}, x)$ for a vector $x + bd$ that improves the objective value and maximizes $w^p(d)$; make it the current x (this is a steepest ascent algorithm).

It follows easily from Corollary 5.2 that variant 1 produces an optimal solution in finite time, which implies the same for variants 2 and 3. Variant 3 is considered in detail in the proof of the validity of the Greedy Algorithm; we show that the behavior of this variant is essentially the same as that of the Greedy Algorithm. The advantage of our Greedy Algorithm over the above variants is that it makes better use of the structure of neighbor systems. In particular, for the (typically) low cost of a sort, the searches in $N(\mathcal{F}, x)$ may be confined to a smaller set of vectors. That is, for each search we make in $N(\mathcal{F}, x)$, we only look in one direction for improving vectors and, under w^p, these directions are decreasing during the algorithm, so once we finish looking in a direction, we never need to look again in that direction. This results in an algorithm with better complexity than the variants for neighbor systems (see the discussion in Section 6). Recall that an initial sort of this type is also carried out in the standard version of the greedy algorithm for matroids and this also results in an algorithm that is more efficient than each of the comparable variants above. In fact, an initial sort is made in all of the greedy algorithms discussed in the introduction (except for the algorithms of Ando et al in [1] and [2], although it turns out that the efficiency of these two algorithms also benefits from a sort).

Let us end this section with a quick example that illustrates how the Greedy Algorithm works and why it would not work if we had chosen $w^p(d) = wd$. Consider the all-neighbor system in Figure 3 (which is also a two-dimensional jump system). Let $x_0 = (0, 0)$ be the starting point and let $w = (2, 1)$ be the objective function; hence $x_3 = (3, 1)$ is the optimal solution. (Although these choices of w and w^p do not satisfy A1, it does not matter for this example.) Using $w^p(d) = wd$, the first move of the algorithm would be from x_0 to x_1, using direction $d' = (1, 0)$, where $w^p(d') = 2$. The second move would be from x_1 to x_2 using direction $d'' = (0, 1)$, where $w^p(d'') = 1$. The second

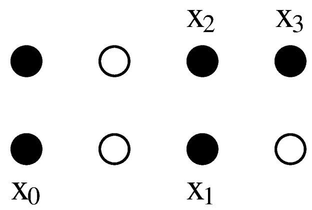

Figure 3. Greedy algorithm example

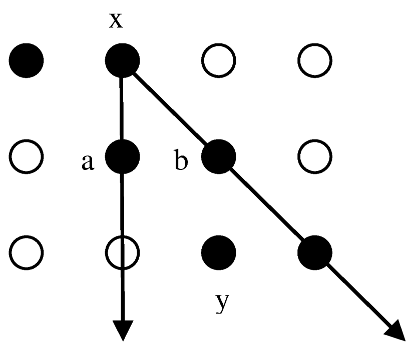

Figure 4. An illustration of Theorem 5.1

move cannot be from x_1 to x_3 using $d''' = (1,1)$, since $w^p(d') < w^p(d''') = 3$. Finally, observe that moving from x_2 to x_3 would require again using direction d'; but this is not allowed since $w^p(d'') < w^p(d')$. Hence the algorithm stops at x_2. Using $w^p(d) = \frac{wd}{\|d\|}$, the first move would again be from x_0 to x_1, using direction d', where $w^p(d') = 2$. In this case the second move would be from x_1 to x_3 using d''', where $w^p(d''') = 1.5$, which takes us to the optimum solution.

§ 5. Characterizing neighbor Systems

In this section we present the key result (Theorem 5.1) used to prove the validity of the greedy algorithm for neighbor systems. This result is interesting in its own right in that it serves as an algebraic characterization of neighbor systems. Roughly speaking, the theorem says that $\mathcal{F}$ is an N-neighbor system if and only if for every $x, y \in \mathcal{F}$, y is in the cone generated by the vectors from x to its neighbors that are between x and y. A 2-dimensional example of an all-neighbor system appears in Figure 4: a and b are the neighbors of x between x and y and the cone generated by them contains y.

Theorem 5.1. *$\mathcal{F}$ is an N-neighbor system if and only if for all $x, y \in \mathcal{F}$, where $x \neq y$, there exists $s_1, \ldots, s_n \in N(\mathcal{F}, x)$ and positive reals $a_1, \ldots, a_n$, such that: s_i is* *between x and y, for $i = 1, \ldots, n$; and*

$$x + \sum_{i=1}^{n} a_i(s_i - x) = y.$$

Corollary 5.2. *Let $\mathcal{F}$ be an N-neighbor system; let $w \in \Re^E$; and let $x, y \in \mathcal{F}$ such that $wx < wy$. Then there exists $s \in N(\mathcal{F}, x)$, such that s is between x and y, and $wx < ws$.*

Remark: The values a in Theorem 5.1 cannot, in general, be chosen to be integral. To see this, consider the following all-neighbor system: $\mathcal{F} = \{x = (0,0,0), x_1 = (1,1,0), x_2 = (1,0,1), x_3 = (0,1,1), y = (1,1,1)\}$, where $N(\mathcal{F}, x) = \{x_1, x_2, x_3\}$. Observe that $y = x + \frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3$ and that the coefficients of $\frac{1}{2}$ are unique.

§ 6. Complexity of the Algorithm

In this section we analyze the complexity of the Greedy Algorithm. We begin with a few definitions.

Let $\mathcal{F}$ be a finite N-neighbor system and define

$$\begin{aligned} u(e) &= \max_{x \in \mathcal{F}} x(e), \\ l(e) &= \min_{x \in \mathcal{F}} x(e), \\ size\,(\mathcal{F}) &= \max_{e \in E} \left(u(e) - l(e) \right). \end{aligned}$$

We assume we have a membership oracle that, in constant time, tells us if a given vector is in a set $\mathcal{F}$.

We have the following complexity result.

Proposition 6.1. *Let $\mathcal{F}$ be an N-neighbor system. Then the Greedy Algorithm applied to $\mathcal{F}$ can be implemented with a worst case time complexity of $O(|E|^2 \log|E| + |E|^2 size(\mathcal{F}))$. If $\mathcal{F}$ is an N_k-neighbor system, then the Greedy Algorithm applied to $\mathcal{F}$ can be implemented with a worst case time complexity of $O(|E|^2 \log|E| + k|E|^2 \log(size\,(\mathcal{F})))$.*

Remark: When the parameter k is fixed, then the complexity of the algorithm is improved for N_k-neighbor systems. This is the case, for example, for jump systems.

In the case of matroids, bases of matroids, generalized matroids, and delta-matroids, $size\,(\mathcal{F}) = 1$, hence the complexity further simplifies to $O\left(|E|^2 \log|E|\right)$. The complexity of the standard "build-up" greedy algorithms for delta-matroids, and integral bisubmodular polyhedra is $O\left(|E| \log|E|\right)$, which is the cost of sorting the weights in w. However, the algorithms for these structures depend upon calls to a stronger oracle than the membership oracle. In particular, the oracle checks if there exists a feasible solution for which a subset of the coordinates are set to specified values (whereas our algorithm needs only to be able to perform this check when all the coordinates are set to specified values). The complexity of the standard greedy algorithms for matroids and integral polymatroids is, again, $O(|E| \log |E|)$, since only a sort of the weights in w is required; in these cases the membership oracle is sufficient, however, the initial vector is 0.

We discuss a faster variation of the Greedy Algorithm, as well as the complexity of greedy algorithms for jump systems, in the following section.

§ 7. Strengthening of the Greedy Algorithm

In this section we show how the Greedy Algorithm can be altered slightly to obtain a better complexity. The resulting variation is closely related to the greedy algorithms of Ando et al [2] and Shioura et al [25] for jump systems. The variation hinges on the following proposition.

We say two vectors $f_1, f_2 \in \Re^E$ are *similarly signed* if, for all $i \in E$, $f_1(i) \neq 0$ and $f_2(i) \neq 0$ imply $f_1(i)$ and $f_2(i)$ have the same sign.

Proposition 7.1. *Let f and g be two objective functions for a finite N-neighbor system $\mathcal{F}$. Suppose f and g satisfy the following conditions:*

1. *f and g are similarly signed;*
2. *$|f(i)| > |f(j)|$ implies $|g(i)| > |g(j)|$, for all $i, j \in E$; and*
3. *$|f(i)| > 0$ implies $|g(i)| > 0$, for all $i \in E$.*

Then x is an optimal solution over $\mathcal{F}$ with g, implies x is an optimal solution over $\mathcal{F}$ with f.

This proposition leads to a variation of the Greedy Algorithm, called the *Altered Greedy Algorithm*, where Step 0 in the Greedy Algorithm is replaced with the following version. The notation $a \gg b$ means the real number a is "very much" larger than the real number b. We continue to assume Assumption A1 holds.

Step 0′ : Set $x \leftarrow x_0$. Label the members of E with $1, \ldots, |E|$ so that $|w(1)| > |w(2)| > \cdots > |w(|E|)|$. Choose w^* such that $|w^*(1)| \gg |w^*(2)| \gg \cdots \gg |w^*(|E|)|$ and so that w and w^* (playing the roles of f and g, respectively) satisfy conditions 1 and 3 of Proposition 7.1. Label the input directions so that $w^p(d_1) > w^p(d_2) > \cdots$ (where w^p is based on w^*) and let s be the maximum index i for which $w^p(d_i) > 0$.

Observe that, in general, the labeling/ordering of the directions in the Altered Greedy Algorithm will be different from the labeling in the Greedy Algorithm.

It is easy to show that the sequence of solutions produced by an application of the Altered Greedy Algorithm for $\mathcal{F}$ with w^* is a sequence of solutions that can result from an application of the algorithms in [2] and [25] for jump systems $\mathcal{F}$ with w. Furthermore, the algorithm behaves here very much like the greedy algorithm in [6] for jump systems, except that we are actually tracking the feasible vectors produced by the algorithm. It follows that our analysis provides a new proof of the validity of these algorithms.

The fact that the Altered Greedy Algorithm finds an optimal solution, and does so with a better complexity than the original Greedy Algorithm, is expressed in the following proposition.

Proposition 7.2. *Let $\mathcal{F}$ be an N-neighbor system. Then, the Altered Greedy Algorithm applied to $\mathcal{F}$ yields an optimal solution and can be implemented with a worst case time complexity of $O(|E| \log |E| + |E|^2 size(\mathcal{F}))$. If $\mathcal{F}$ is an N_k-neighbor system, then the Altered Greedy Algorithm applied to $\mathcal{F}$ can be implemented with a worst case time complexity of $O(|E| \log |E| + k|E|^2 \log(size\,(\mathcal{F}))$.*

The complexity of the Altered Greedy Algorithm, for the case of jump systems, is the same as the complexity of the greedy algorithm in Shioura et al [25], which is the best-known complexity of an incremental-style greedy algorithm for jump systems.

§ 8. Further Generalizations

In this section we briefly mention a generalization of the notion of neighbor systems that allows us to optimize, with a simple greedy algorithm, over more sets of vectors $\mathcal{F}$. Let us begin by generalizing the notion of directions to be an arbitrary set of vectors in $\mathbf{Z}^E$, which we call $\mathcal{D} = \{d_1, \ldots, d_p\}$. For $x, y \in \mathcal{F}$, let us say that y is a *$\mathcal{D}$-neighbor* of x if there exists a vector $d \in \mathcal{D}$ and a positive integer b such that $y = x + bd$ and there exists no positive integer $b' < b$ such that $x + b'd \in \mathcal{F}$.

A *$\mathcal{D}$-neighbor function* is a function that takes as input any set $\mathcal{F}$ with any $x \in \mathcal{F}$ and outputs a subset of the neighbors of x in $\mathcal{F}$; it is denoted $N_\mathcal{D}(\mathcal{F}, x)$. We use Theorem 5.1 to make the following definition.

Definition 8.1. $\mathcal{F}$ is an $N_\mathcal{D}$-neighbor system if for all $x, y \in \mathcal{F}$, where $x \neq y$, there exists $s_1, \ldots, s_n \in N_\mathcal{D}(\mathcal{F}, x)$ and positive reals $a_1, \ldots, a_n$, such that: s_i is between x and y, for $i = 1, \ldots, n$; and

$$x + \sum_{i=1}^{n} a_i(s_i - x) = y.$$

With essentially no changes, the definition of w^p and the Greedy Algorithm can be generalized and the proof of the Greedy Algorithm's validity can immediately be adapted to prove that this generalized greedy algorithm works.

References

[1] K. Ando, S. Fujishige, and T. Naitoh. A greedy algorithm for minimizing a separable convex function over an integral bisubmodular polyhedron. *J. Oper. Res. Soc. of Japan* Vol. 37, No. 3 (Sept. 1994), 188-196.

[2] K. Ando, S. Fujishige, and T. Naitoh. A greedy algorithm for minimizing a separable convex function over a finite jump system. *J. Oper. Res. Soc. of Japan* Vol. 38, No. 3 (Sept. 1995), 362-375.

[3] E. Balas and W. R. Pulleyblank. The perfectly matchable subgraph polytope of a bipartite graph. *Networks* 13 (1983), 495-516.

[4] A. Bouchet. Greedy algorithm and symmetric matroids. *Math. Prog.* 38 (1987), 147-159.

[5] A. Bouchet. Matchings and $\triangle$-matroids. *Discr. Appl. Math.* 24 (1989), 55-62.

[6] A. Bouchet and W. H. Cunningham. Delta-matroids, jump systems, and bisubmodular polyhedra. *SIAM J. Disc. Math.* Vol. 8, No. 1 (1995), 17-32.

[7] R. Chandrasekaran and S. N. Kabadi. Pseudomatroids. *Discr. Math.* 71 (1988), 55-62.

[8] W. H. Cunningham. Matching, matroids, and extensions. *Math. Prog.*, Series B 91 (2002), 515-542.

[9] A. Dress and T. Havel. Some combinatorial properties of discriminants in metric vector spaces. *Adv. Math.* 62 (1986), 285-312.

[10] F. D. J. Dunstan and D. J. A. Welsh. A greedy algorithm solving a certain class of linear programmes. *Math. Prog.* 5 (1973), 338-353.

[11] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Structures and their Applications.* R. K. Guy, H. Hanani, N. Sauer, and J. Schönheim, eds. Gordon and Breach, New York (1970), 69-87.

[12] A. Federgruen and H. Groenevelt. The greedy procedure for resource allocation problems: necessary and sufficient conditions for optimality. *Oper. Res.* Vol. 34, No. 6 (1986), 909-918.

[13] A. Frank. Generalized polymatroids. In A. Hajnal et. al., eds., *Finite and Infinite Sets.* North-Holland, Amsterdam-New York (1984), 285-294.

[14] A. Frank and É. Tardos. Generalized polymatroids and submodular flows. *Math. Prog. Ser. B.* 42 (1988), 489-563.

[15] S. Fujishige. *Submodular Functions and Optimization, Second Edition.* (*Annals of Discrete Mathematics*, Vol. 58) Elsevier, Amsterdam (2005).

[16] J. F. Geelen. Lectures on Jump Systems. Center of Parallel Computing, University of Cologne, Germany (1997).

[17] H. Groenevelt. Two algorithms for maximizing a separable concave function over a polymatroid feasible region. *Eur. J. Oper. Res.* Vol 54, Issue 2 (Sept. 1991), 227-236.

[18] B. Korte, L. Lovász, and R. Schrader. *Greedoids.* Springer-Verlag, New York, Berlin (1991).

[19] E. Lawler. *Combinatorial Optimization, Networks and Matroids.* Holt, Rinehart, and Winston, New York (1976).

[20] L. Lovasz. The membership problem in jump systems. *Journal of Comb. Theory, B* 70 (1997), 45-66.

[21] K. Murota. M-convex functions on jump systems: A general framework for minsquare graph factor problem. *SIAM J. Disc. Math.* Vol. 20, No. 1 (2006), 213-226.

[22] M. Nakamura. A characterization of those polytopes in which the greedy algorithm works. Abstract: 13th International Symposium on Mathematical Programming, Tokyo. 1988.

[23] L. Qi. Directed submodularity, ditroids and directed submodular flows. *Math. Prog.* 42 (1988), 579-599.

[24] V.V. Shenmaier. A greedy algorithm for maximizing a linear objective function. *Disc. Appl. Math.* Vol. 135 (2004), 267-279.

[25] A. Shioura and K. Tanaka, Polynomial-time algorithms for linear and convex optimization on jump systems. *SIAM J. Disc. Math.* Vol. 21, No. 2 (2007), 504-522.

[26] É. Tardos. Generalized matroids and supermodular colourings. Colloquia Mathematica Societatis Janos Bolyai 40. Matroid Theory, Szeged (Hungary), 1982.

[27] H. Whitney. On the abstract properties of linear dependence. *Amer. J. Math.* 57 (1935), 507-533.