ORSA Journal on Computing
Vol. 1, No. 3, Summer 1989

0899-1499/89/0103-0181 $01.25

On the Greedy Solution of Ordering Problems

AVI DECHTER *Department of Management Science, California State University, Northridge, CA 91330*

RINA DECHTER *Cognitive Systems Laboratory, Computer Science Department, University of California, Los Angeles, CA 90024*

(Received: January 1988; final revision received: September 1989; accepted: February 1989)

The greedy method is a well-known approach for problem solving directed mainly at the solution of optimization problems. Leading theoretical frameworks dealing with the optimality of greedy solutions (e.g., the matroid and greedoid theories) tacitly assume that the greedy algorithm is always guided by the cost function to be optimized, namely, it builds a solution by adding, in each step, an element that contributes the most to the value of the cost function. This paper studies a class of problems for which this type of a greedy algorithm does not optimize the given cost function, but for which there exists a secondary objective function, called a greedy rule, such that applying the greedy algorithm to the secondary objective function yields a solution which is optimal with respect to the original cost function.

The greedy method is a well-known approach for problem solving directed mainly at the solution of optimization problems involving the selection and/or the ordering of elements from a given set so as to maximize or minimize a given objective function. Nilsson[9] views the greedy algorithm as an irrevocable (i.e., without backtracking) search strategy that uses local knowledge to construct a solution in a "hill climbing" process. The greedy control strategy selects the next state so as to achieve the largest possible improvement in the value of some measure that, as pointed out by Horowitz and Sahni,[4] may or may not be the objective function.

Our interest in greedy methods originated in earlier research in the area of heuristic problem solving.[2] The connection between these two subjects is twofold. First, greedy schemes are probably the closest to emulate human problem-solving strategies because they require only a minimum amount of memory space and because they often produce adequate (if not optimal) results. (Due to the small size of human short-term memory, it is very hard to conceive of a human conducting best-first or even backtracking search, both requiring retention of some properties of previously suspended alternatives.) Second, **greedily optimized** problems (i.e., for which a greedy algorithm produces optimal solutions) represent a class of relatively **easy** problems. Pearl[10] has demonstrated that many heuristics used in the solution of hard problems are based on simplified models of the problem domain, which admit easy solutions. Therefore, the ability to characterize classes of easy problems is important, particularly if the process of discovering heuristics is to be mechanized.

This paper is concerned exclusively with **ordering problems**, involving a set of elements and a cost function defined on all permutations of the elements, where the task is to order the elements so as to maximize (or minimize) the value of the cost function. Job sequencing on a single machine and the traveling salesman problems are two examples of this class of problems.

A theoretical framework, called greedoid theory, which characterizes a class of ordering problems that can be solved optimally by greedy algorithms, is due to Korte and Lovasz.[6] The greedoid structure is a generalization of the matroid structure which provides a theoretical foundation for the optimality of the greedy algorithm **on selection problems**. (In contrast with ordering problems, selection problems involve a set of elements and a cost function defined on all **unordered** subsets of elements, where the task is to select a subset of elements which satisfies some property, so as to maximize (or minimize) the value of the cost function. The minimum weight spanning tree problem is a well known example of this class of problems. For further details on matroids refer, for example, to Lawler[8] or Welsh[12]).

The greedoid theory (as well as the matroid theory) considers only greedy algorithms that use the cost function to be optimized as their selection criterion, namely, which build the solution by adding, at each step, that element which results in maximum improvement in the value of that cost function. The appendix to this paper lists a number of known ordering problems for which this greedy algorithm does not optimize the cost function, but for which there exists a **secondary objective function**, which we call a **greedy rule**, such that

Subject classification: Analysis of algorithms: optimality of greedy algorithms.

applying the greedy algorithm to the greedy rule yields a solution which is optimal with respect to the original cost function. Our objective in this paper is to characterize these problems, which are not covered by the greedoid theory.

The remainder of the paper is organized as follows. In Section 1 we briefly review the basic facts of the greedoid theory and demonstrate that it does not cover all greedily optimized ordering problems. In Section 2 we generalize the notion of greedy optimization, define ranking functions and state the necessary and sufficient conditions for a class of problems to be greedily optimized by such functions. Section 3 discusses a class of ranking functions, called uniform ranking functions, which are guaranteed to be optimizing. Necessary and sufficient conditions for a class of problems to have a uniform ranking function and methods for its discovery are also discussed. In Section 4 we contrast problems which are greedily optimized by a ranking function with those which are greedily optimized by the cost function. Two examples are given in Section 5 to demonstrate the results and methods developed in the paper. A summary and conclusions are given in Section 6.

1. Greedoids, Greedoid Costs, and Greedoid Optimization

Greedoids is a mathematical structure that was proposed by Korte and Lovasz[6] for the study of the optimality of greedy solutions. A greedoid is a pair (E, L), where E is a finite set and L is a collection of finite sequences (without repetitions) of elements of E, also called *strings* or *words*, which satisfy the following axioms:

1. $\Phi \in L$
2. if $\alpha \in L$ and $\alpha = \beta \cdot \gamma$ then $\beta \in L$
3. if $\alpha, \beta \in L$ and $|\alpha| > |\beta|$, then there exists an $x \in \alpha$ such that $\beta \cdot \langle x \rangle \in L$,

where $\alpha \cdot \beta$ denotes the concatenation of two words α, β, and $|\alpha|$ is the length of α.

A word is called *feasible* if it belongs to L. Maximal feasible words are called *basic words*. The optimization problem P considered by the greedoid theory is a pair $((E, L), w)$, where (E, L) is a greedoid and $w: L \rightarrow R$ is real valued objective function. The task associated with P is:

(O) $\max\{w(\alpha) \mid \alpha$ is a basic word in $L\}$.

Let E^* denote the collection of all possible strings over E. Then, (E, E^*) is a greedoid, and each ordering problem of the type defined in the previous section can be viewed as a greedoid optimization problem $P = ((E, E^*), w)$, where $w: E^* \rightarrow R$.

The greedy algorithm considered by the greedoid theory for (O) is described as follows:

(GA) For $i = 1, 2, \ldots$ choose an element $x_i \in E$ such that

$$\langle x_1 \ldots x_i \rangle \in L$$

and

$$w(\langle x_1 \ldots x_i \rangle) = \max\{w(\langle x_1 \ldots x_{i-1}, y \rangle \mid \langle x_1 \ldots x_{i-1}, y \rangle \in L)\}$$

Stop if no such x_i exists.

Korte and Lovasz provide the following sufficient condition on the objective function w that guarantees that the greedy algorithm produces an optimal solution to (O).

(W) Let be $\alpha \in L$ and $\alpha \cdot \langle x \rangle \in L$.

If for all $y \in E$ such that $\alpha \cdot \langle y \rangle \in L$

$$w(\alpha \cdot \langle x \rangle) \geq w(\alpha \cdot \langle y \rangle),$$

then

(i) for all $y \in E$ and $\beta, \gamma \in E^*$ such that $\alpha \cdot \beta \cdot \langle x \rangle \cdot \gamma$, $\alpha \cdot \beta \cdot \langle y \rangle \cdot \gamma \in L$: $w(\alpha \cdot \beta \cdot \langle x \rangle \cdot \gamma) \geq w(\alpha \cdot \beta \cdot \langle y \rangle \cdot \gamma)$,

and

(ii) for all $y \in E$ and $\beta, \gamma \in E^*$ such that $\alpha \cdot \langle x \rangle \cdot \beta \cdot \langle y \rangle \cdot \gamma$, $\alpha \cdot \langle y \rangle \cdot \beta \cdot \langle x \rangle \cdot \gamma \in L$: $w(\alpha \cdot \langle x \rangle \cdot \beta \cdot \langle y \rangle \cdot \gamma) \geq w(\alpha \cdot \langle y \rangle \cdot \beta \cdot \langle x \rangle \cdot \gamma)$.

While the underlying structure of all the problems in the appendix is a greedoid, their objective functions do not satisfy the sufficient condition, and, indeed, they cannot be optimally solved by the greedy algorithm (GA). Yet, each of these problems can be solved by a greedy algorithm applied to a secondary function.

For example, consider the problem of minimizing weighted average flow-time on a single processor (problem #5 in the appendix). This problem can be formulated in terms of a pair $((E, E^*), C)$, where E is a set of jobs, and the cost function (to be minimized), C, is defined on any sequence $\sigma \in E^*$ of n jobs as follows:

$$C(\sigma) = \sum_{i=1}^{n} u_i \sum_{j=1}^{i} p_j, \tag{1}$$

where p_i and u_i, respectively, associate with the ith job in the sequence a **processing time** and an **importance weight**. Applying the greedy algorithm (GA) does not guarantee an optimal solution. However, it is well known that an optimal solution is easily obtained by a greedy algorithm which, in each step, adds to the sequence a job with a minimum value of p_i/u_i.

This requires a slightly different view of greedy algorithms than the one expressed by (GA). Greedy algorithms build solutions by, at each step, adding to a partial solution one element that, individually, has the **highest merit** among all remaining elements. This "figure of merit" is the value of some criterion function which we have called a **greedy rule**. In the next section we discuss briefly greedy rules in general and then shift our attention to a special type of greedy rules called **ranking functions**.

2. Greedy Rules and Ranking Functions

In general, greedy rules are real valued functions defined on the language L in precisely the same manner as the cost function w. Let $f: L \rightarrow R$ be a greedy rule. Then solving the problem (O) using the greedy rule f means using algorithm (GA) where w is replaced with f.

Each of the ordering problems in the appendix is optimized by a **single argument** greedy rule satisfying:

$$f(\langle x_1, \ldots, x_i \rangle) = g(x_i), \tag{2}$$

where g is a real-valued function defined on each element. Namely, evaluation of the greedy rule requires only information pertaining to the last element in the sequence. We refer to such rules as **ranking functions**. A ranking function induces a weak order on the elements of E and the greedy procedure simply chooses elements in a nonincreasing order of f. The remainder of this paper is concerned only with problems which are greedily optimized by ranking functions.

The greedoid theory does not explicitly make the distinction between a class of problems and its instances. However, observe that every greedoid optimization problem really represents a class of problems. For every subset $E_I \subset E$ of the ground set E of a greedoid (E, L) there is a corresponding structure (E_I, L_I), called a restriction, where $L_I = \{\alpha \in L \mid \alpha \in E_I^*\}$. It is easy to see that the restriction (E_I, L_I) is a greedoid. We refer to the problem $P_I = ((E_I, L_I), w)$, associated with each of the restrictions (E_I, L_I) of a greedoid (E, L), as a **subproblem** of the problem $P = ((E, L), w)$. If the sufficient condition (W) for greedoid optimization holds for a problem P, then it holds for any of its subproblems. We should expect any optimizing greedy rule to have a similar hereditary property.

Definition. A ranking function is **optimizing** for some problem P, if for every subproblem, P_I, a sequence is optimal if and only if it satisfies the weak order induced by it. A problem is said to be **greedily optimized** by a ranking function if it permits an optimizing ranking function. (In the sequel we will occasionally use the term *greedily optimized* to stand for *greedily optimized by a ranking function* when no confusion can arise.)

We now give a necessary and sufficient condition for a problem to be greedily optimized.

Theorem 1. *A necessary and sufficient condition for a problem P to be greedily optimized is that for any two elements a and b, and for all subproblems in which they both participate, one of the following situations must occur:*

1. *If there exist a subproblem in which a precedes b in all of its optimal sequences then a precedes b in all optimal sequences of all subproblems, or else,*
2. *In all subproblems where both elements a and b appear any optimal sequence in which a appears before b can be transformed into an optimal sequence in which b stands before a by exchanging the positions of a and b.*

Proof. Suppose the problem is greedily optimized by a ranking function and let a and b be two elements in some subproblem, P_I. If a precedes b in all optimal sequences, f must rank a strictly higher than b in order to generate this problem's optimal sequences and none else. Therefore, in all other subproblems f will generate only sequences where a precedes b. Since f is assumed to generate all optimal sequences, they all must have a before b. If a doesn't precede b in all optimal sequences (we are in part 2 of the condition) then there are at least two optimal sequences with both a before b and b before a. For an optimizing ranking function to yield these two sequences it must rank a and b as equal (i.e., $f(a) = f(b)$). Such a ranking function must generate all sequences in which a and b's locations are completely exchangeable, and since it is optimizing all such sequences must be optimal for all subproblems. This yields the necessary part.

The condition is sufficient since the relative positions of any pair of elements in all optimal sequences defines an order relationship among the elements. Namely, if a appears before b in some optimal sequences we say that $a \preccurlyeq b$. It is easy to show that the condition guarantees that this is a well defined relation which is also transitive and thus constitutes a weak order. There exists, therefore, a real function f, defined over all the elements which satisfy $f(a) \leq f(b)$ iff $a \preccurlyeq b$ (see [7]). The function f is the ranking function that yields the optimal solutions. ■

To illustrate the necessary condition of Theorem 1, consider a problem defined by a set of four elements $\{1, 2, 3, 4\}$ and some cost function. Suppose that the only optimal sequence for the subproblem defined by the set $\{1, 2, 3\}$ is $(3, 2, 1)$, and that the only optimal

sequence for the subproblem defined by the set {1, 2, 4} is (4, 1, 2). Then, this problem does not have any optimizing ranking function because elements 1 and 2 do not have the same ordering in the two optimal sequences.

The verification of the condition given in Theorem 1 usually depends on the knowledge of all optimal solutions in each subproblem, and cannot be carried out by simple manipulation of the problem representation. Therefore, its usefulness is very limited. In the next section we present stronger, potentially more easily verifiable, requirements that constitute sufficient (but not necessary) conditions for greedily optimized problems.

3. Uniform Ranking Functions

A reasonable approach for obtaining sufficient conditions for greedily optimizing problems is to extend the properties required by Theorem 1 for the optimal sequence to **all possible sequences**. This leads to the following definition.

Definition. Let $P = ((E, E^*), C)$ and let f be a ranking function. A ranking function f is called **uniform** relative to P, if for every sequence σ of elements in E,

$$C(\sigma) > C(\sigma^i) \quad \text{if } f(\sigma_i) > f(\sigma_{i+1}) \tag{3}$$

and

$$C(\sigma) = C(\sigma^i) \quad \text{if } f(\sigma_i) = f(\sigma_{i+1})$$

for all i, where σ_i is the ith element in σ and σ^i is the sequence resulting from the exchange of the ith and $(i+1)$st elements in σ.

Theorem 2. *Every problem P that has a uniform ranking function f, is greedily optimized by it.*

Proof. We have to show that if P has a uniform ranking function f then every sequence generated by it is optimal and that every optimal sequence can be generated by it. Let P_l be any subproblem of P, σ be any sequence generated by the uniform ranking function f, and σ^* be an arbitrary optimal sequence of P_l. To prove that σ is optimal we will show that σ^* could be transformed into σ by local exchange operations in a way that does not change the cost associated with intermediate sequences. Let i be the first location in which the two sequences differ, namely, for all $k \leq i-1$, $\sigma_k^* = \sigma_k$ and the sets of elements from i to n in both sequences are identical. Therefore, element σ_i appears in some subsequent location of σ^*. Assume $\sigma_i = \sigma_j^*$ for some $j > i$. Since σ's elements are sequenced by f, $f(\sigma_i)$ is greater than or equal to the f value of subsequent elements in σ, and therefore it is greater than or equal to subsequent elements of σ^*. In particular, $f(\sigma_i) = f(\sigma_j^*) \geq f(\sigma_{j-1}^*)$. If $f(\sigma_j^*) > f(\sigma_{j-1}^*)$ then the cost of σ^* must increase by exchanging σ_{j-1}^* with σ_j^*, but this would contradict the assumption that σ^* is optimal. Thus, it must be that $f(\sigma_j^*) = f(\sigma_{j-1}^*)$, which means that σ_{j-1}^* and σ_j^* may be exchanged without changing the value of the cost function. The resultant sequence is therefore also optimal. The (original) σ_j^* element which is now in location $j-1$ of σ^* can propagate down in this fashion with no change in the cost until it reaches the ith position. At this point the prefix of i elements of the resultant optimal sequence and the prefix of i elements of σ are identical. The next differing position between the two orderings, which may appear only in a position higher than i, will be identified, and the process will continue in the same way. Since the discrepancies between the sequences appear in increasing locations, the process is guaranteed to terminate.

To show that every optimal sequence may be generated by the uniform ranking function consider again the sequences σ and σ^* as defined in the first part. The procedure described above implies that $f(\sigma_i) = f(\sigma_i^*)$, which means that by applying the tie-breaking rule differently, the greedy algorithm using f could have picked σ_i^* for the ith position in σ. Thus, the ranking function f can be used to generate an optimal sequence with at least the first i elements identical to the first i elements of σ^*. Repeating this argument as necessary, it is easy to see that longer and longer prefixes of σ^*, and eventually the entire sequence, can be generated using f by breaking ties "correctly." ■

We next examine a necessary and sufficient condition on the cost function for there to exist a uniform ranking function. Let E_{ab}^* be the set of all permutations of subsets of E in problem P for which element a immediately precedes element b.

Definition. A cost function C is said to be **pairwise preferentially independent** (p.w.p.i.) if $\forall a, b$ either

$$C(\sigma) > C(\sigma^a) \quad \forall \sigma \in E_{ab}^*, \tag{4}$$

or

$$C(\sigma) < C(\sigma^a) \quad \forall \sigma \in E_{ab}^*,$$

or

$$C(\sigma) = C(\sigma^a) \quad \forall \sigma \in E_{ab}^*,$$

where σ^a is the sequence resulting by the exchange of the adjacent elements a and b in σ. In the first two cases we say that C prefers a over b (resp. b over a), and denote it by $a >_{\text{p.w.}} b$ (resp. $b >_{\text{p.w.}} a$). In the third case we say that C is indifferent between a and b and use the notation $a \sim_{\text{p.w.}} b$.

Definition. A pairwise preferentially independent cost function C is said to be acyclic if the relation $\gtrsim_{\text{p.w.}}$ is

transitive, i.e.,

$$\text{if } a \gtrsim_{\text{p.w.}} b \quad \text{and} \quad b \gtrsim_{\text{p.w.}} c \quad \text{then } a \gtrsim_{\text{p.w.}} c. \tag{5}$$

This last property (i.e., that the relation "C prefers a on b" satisfies transitivity) is required to assure a weak order[7] and does not follow automatically from the p.w.p.i. property. The following example shows that a cost function can be pairwise preferentially independent but not acyclic. Consider a subproblem defined by a set of three elements {1, 2, 3} with a cost function C that creates the following complete order among all different sequences:

$$(321) > (132) > (213) > (312) > (123) > (231).$$

For this subproblem, C is pairwise preferentially independent with 3 being preferred to 2 and 2 being preferred to 1. However, 1 is preferred to 3 thus violating transitivity.

Theorem 3. *A necessary and sufficient condition for a problem $P = ((E, E^*), C)$ to have a uniform ranking function is that C is p.w.p.i. and acyclic (denoted a.p.w.p.i.).*

Proof. It is obvious that the existence of a uniform ranking function for a problem P implies that the cost function is a.p.w.p.i. We will therefore show only the sufficiency part. Since C is a.p.w.p.i. it induces a weak order on all the elements of E. Therefore, there exists[7] a real function f on the elements of E that reflects this ordering, i.e.,

$$a >_{\text{p.w.}} b \quad \text{iff } f(a) > f(b) \tag{6}$$

and

$$a \sim_{\text{p.w.}} b \quad \text{iff } f(a) = f(b).$$

Obviously, f is a uniform ranking function, and, by Theorem 2, P is greedily optimized by it. ■

The next theorem suggests a process which has the potential of identifying an a.p.w.p.i. cost function and indicating its optimizing ranking function.

Theorem 4. *Let P be a problem $P = ((E, E^*), C)$ and σ any sequence of any subset of the elements in E. If the cost function C satisfies*

$$C(\sigma) - C(\sigma') = K(\sigma) \cdot (d(\sigma_i) - d(\sigma_{i+1})) \tag{7}$$

for all i, where $K(\sigma)$ is a nonnegative function defined on σ and d is a function defined on single elements, then d is an optimizing ranking function.

Proof. It is easily seen that d, satisfying (7), is a uniform ranking function and therefore, by Theorem 2, optimizing. ■

The process Theorem 4 suggests is: perform symbolic manipulation on the difference between the cost of an arbitrary sequence and that of the sequence which results from exchanging the ith and $(i + 1)$st elements. If you obtain an expression that satisfies condition (7) then an optimizing ranking function exists and is given by d in that expression.

As an example consider again the single-processor job sequencing problem whose cost function is given in (1). Let σ' be a sequence resulting from the exchange of the ith and $(i + 1)$st elements. For this cost function we get:

$$\begin{aligned} C(\sigma) - C(\sigma') &= (u_{i+1}p_i - u_i p_{i+1}) \\ &= u_{i+1}u_i\left(\frac{p_i}{u_i} - \frac{p_{i+1}}{u_{i+1}}\right). \end{aligned} \tag{8}$$

In this case the ranking function suggested from the above representation is $f(p_i, u_i) = p_i/u_i$.

The requirement that an optimizing ranking function must be optimal for all subproblems of a problem provides a simple means for screening potential ranking functions, which is summarized in the following result:

Theorem 5. *If P is any greedily optimized problem then an optimizing ranking function f has to agree with the ordering dictated by the cost function on pairs of elements, namely, for every two elements a and b, $C(a, b) > C(b, a)$ iff $f(a) > f(b)$.*

Proof. If a problem has an optimizing ranking function, then it also has to optimally solve all subproblems of two elements. To do that the ranking function has to satisfy the above condition, which will induce a weak order on the elements of E. ■

If a problem is known, or believed, to be greedily optimized, then Theorem 5 suggests a way for generating an optimal solution: sequence the element in an order satisfying the weak order dictated by applying the cost function to pairs of elements. When a problem is not known a priori to be greedily optimized, Theorem 5 may be used to screen candidate ranking functions by rejecting any candidate that does not satisfy $C(a, b) > C(b, a)$ iff $f(a) > f(b)$ for any two elements a and b of E. If the relation implied by the cost function on pairs is not transitive, we can conclude that the problem does not have any optimizing ranking function. Asserting that a candidate f is indeed optimizing requires other means (e.g., the a.p.w.p.i. property).

4. Dominant Ranking Functions

As noted earlier, a common greedy rule is the cost function itself, i.e.,

$$f(\langle x_1, \ldots, x_i \rangle) = C(\langle x_1, \ldots, x_i \rangle). \tag{9}$$

This means that at each step the algorithm chooses that element which, if it were the last, would yield the best cost (i.e., a "myopic" policy). The Greedoid Theory is

concerned solely with this particular greedy rule. In this section we characterize the optimal sequences of problems that are greedily optimized by their cost functions, and provide a necessary condition for a ranking function to generate such sequences.

The cost function is an optimizing greedy rule for a problem P if, and only if, for any subproblem P_l, any subsequence $\langle x_1, \ldots, x_j \rangle$ of E_l that has a maximal cost over all subsets of size j of E_l can be extended to a sequence of length $j + 1$ that has a maximal cost over all subsets of size $j + 1$ of E_l. Formally,

$$\forall \langle x_1, \ldots, x_i \rangle \text{ optimal over } \Phi_i \quad \exists x_{i+1} \in E_l - \{x_1, \ldots, x_i\} \tag{10}$$

such that

$$\langle x_1, \ldots, x_i, x_{i+1} \rangle \text{ is optimal on } \Phi_{i+1},$$

where Φ_i denotes all permutations of subsets of i elements in E_l. When condition (10) is satisfied, the greedy rule (9) generates an optimal sequence, σ, satisfying the following property: any subsequence $\langle x_1, \ldots, x_j \rangle$ of σ has a maximal cost over all subsets of size j of E_l. An optimal sequence that has this property is called a **dominant sequence**. Clearly, a problem which is optimized by the cost function has all its optimal sequences dominant. Also, if a problem has only dominant optimal sequences it can be greedily optimized by the cost function. Theorem 6 summarizes the above:

Theorem 6. *A necessary and sufficient condition for a problem to be greedily optimized by the cost function is that all its optimal sequences are dominant.*

Dominance is not a necessary condition for the optimality of a greedy rule, however. For example, the cost function

$$C(\sigma) = \sum_{i=1}^{n} u_i \sum_{j=1}^{i} p_j \tag{11}$$

which, as we already know, is greedily optimized by the ranking function $g(u, p) = p/u$, is not dominant. To see this, consider the three-element subproblem, in which each element i has the parameters (u_i, p_i):

$$x_1 = (1, 4), \quad x_2 = (0.5, 3), \quad x_3 = (5, 10). \tag{12}$$

The values assigned by the ranking function to the elements are: $g(x_1) = 4$, $g(x_2) = 6$, $g(x_3) = 2$, so that the optimal sequence is: $\langle x_2, x_1, x_3 \rangle$. Evaluating sequences of two elements we see that

$$C(\langle x_2, x_1 \rangle) = 8.5$$

while

$$C(\langle x_3, x_1 \rangle) = 64.$$

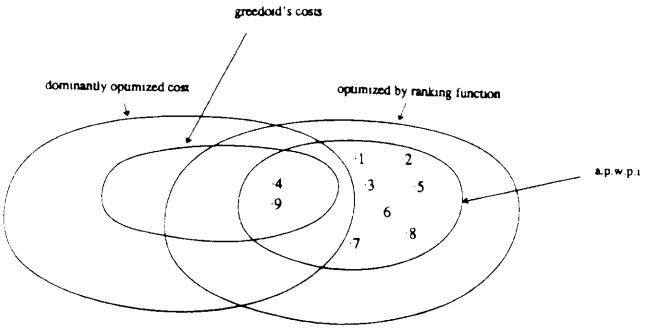

Figure 1. Relationships among classes of greedily optimized problems.

Obviously, the length-2 subsequence of the optimal sequence is not maximal, and thus the ranking function is not dominant.

We observe, similar to the discussion at the end of the previous section, that there is a simple way for recognizing cost functions which are not dominant. Since the cost function is defined for sequences of one element as well, to make pairs dominant, the cost function must satisfy

$$\text{if } C(\langle a, b \rangle) > C(\langle b, a \rangle) \quad \text{then } C(a) \geq C(b). \tag{13}$$

In particular, a **dominant ranking function**, f, has to agree with the partial order imposed by the cost of pairs and with the partial order imposed by the cost of single elements and, therefore, these two partial orders (assuming both are transitive), should coincide. If inconsistency is found, i.e., condition (13) is not satisfied, then the hypothesis that the problem has a dominant optimizing greedy rule can be rejected. (It still may be optimized by a non-dominant greedy rule as the last example has demonstrated.)

The relationships between the different classes of greedily optimized problems discussed thus far is presented in Figure 1.

5. Examples

In this section we illustrate the ideas presented above by verifying the properties of two known greedily optimized problems.

The first problem is **minimizing maximum job lateness on a single machine** (#6). Given n jobs, each associated with a deadline d_i and a processing time p_i, find an optimal sequencing that minimizes the maximum job lateness, $\max\{F_i - d_i\}$, where F_i is the flow time of job i defined by:

$$F_i = \sum_{j=1}^{i} p_j.$$

Jackson[5] proved that the problem is greedily optimized by the due-dates (he used the process suggested by Theorem 4).

Let $x_1 = (p_1, d_1)$ and $x_2 = (p_2, d_2)$ be a pair of jobs with their processing times and due-dates. Using the process suggested by Theorem 5, of identifying ranking functions based on costs of pairs, it can be shown that

$$C(\langle x_1, x_2\rangle) > C(\langle x_2, x_1\rangle) \leftrightarrow d_1 > d_2, \quad (14)$$

that is,

$$\max\{p_1 - d_1, p_1 + p_2 - d_2\} > \max\{p_2 - d_2, p_2 + p_1 - d_1\} \leftrightarrow d_1 > d_2. \quad (15)$$

Therefore, d_i constitutes indeed an optimizing ranking function. The cost associated with one element is

$$C(x_1) = p_1 - d_1. \quad (16)$$

This cost does not provide the same ordering as the due-dates and, therefore, the necessary condition for being greedily optimized by the cost function is violated (see (13)).

The second problem is **minimizing maximum flow-time in a two-machine flow-shop**. Let (A_i, B_i) be a pair associated with the ith job. A_i is the length of time the job must spend on the first machine of the shop and B_i is the time spent on the second machine. For each i, A_i must be completed before B_i can begin. Given the $2n$ values: $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_n$, find the ordering of these jobs on each of the two machines so that neither the precedence nor the occupancy constraints are violated and so that the maximum of the flow time F_i is made as small as possible.

It has been shown,[1] again, by following the process suggested by Theorem 4, that the maximum flow-time is minimized if job j precedes job $j + 1$ when

$$\min\{A_j, B_{j+1}\} < \min\{A_{j+1}, B_j\}. \quad (17)$$

Looking only at two-job problems, it is easily verified that the ordering dictated by (17) coincides with the order determined by costs on pairs. If (A_1, B_1) and (A_2, B_2) are two jobs then the cost function for the sequence $\langle x_1, x_2\rangle$ is:

$$C(\langle x_1, x_2\rangle) = A_1 + \max\{A_2, B_1\} + B_2. \quad (18)$$

It can be shown that

$$A_1 + \max\{A_2, B_1\} + B_2 < A_2 + \max\{A_1, B_2\} + B_1 \quad (19)$$

iff

$$\min\{A_1, B_2\} < \min\{A_2, B_1\}, \quad (20)$$

which is indeed the same as (17). From the transitivity property of the order induced by (20) we know that there exists a ranking function f on **individual** jobs that reflects this order. After some manipulation such a ranking function can indeed be formed. It can be shown that if

$$\min\{A_1, B_2\} < \min\{A_2, B_1\} \quad (21)$$

then

$$\frac{\text{sign}(A_1 - B_1)}{\min(A_1, B_1)} < \frac{\text{sign}(A_2 - B_2)}{\min(A_2, B_2)}. \quad (22)$$

Therefore the function

$$f(A_i, B_i) = \frac{\text{sign}(A_i - B_i)}{\min(A_i, B_i)} \quad (23)$$

is a uniform ranking function for the problem. (The processing times on both machines are assumed to be strictly positive, but a generalization to the case where either one is zero is straightforward.) The cost associated with one element only is $A_1 + B_1$ and it does not coincide with the ranking function (23). We can therefore conclude that this cost function will not necessarily yield an optimal solution since the necessary condition (13) is violated.

6. Summary and Conclusions

This paper provides necessary and sufficient conditions for a problem to be greedily optimized by a ranking function. The virtue of these conditions is that they are easy to test and thus may be useful in mechanizing the process of generating greedy strategies by computers.

The necessary condition (Theorem 5) can be translated into a simple procedure for rejecting the hypothesis that a problem is greedily optimized and otherwise, determines a partial order which all optimizing ranking function (given that there is one), must satisfy. The sufficient condition identifies a class of pairwise preferentially independent and acyclic (a.p.w.p.i.) problems that are optimized via a uniform ranking function, and show that several well known examples fall into this class. A procedure, which can be used to verify the p.w.p.i. property, and which can identify uniform ranking functions is provided. Although the p.w.p.i. property is quite restrictive, **all** the problems in the Appendix that are optimized by a ranking function possess this property (Figure 1).

Observing that the optimal solutions to many greedily optimized problems are **dominant**, and since human intuition usually suggests using the cost as the greedy rule, we also relate our study to this class. We stress that this is a limited class since many greedily optimized problems do not have dominant solutions. The greedoid theory provides a sufficient condition for dominance. A simple, but useful, necessary condition

for a problem to be dominantly optimized is given in (13).

Appendix: A Glossary of Greedily Optimized Ordering Problems

1. **The spanish treasure problem.**[11] An unknown number of chests of spanish treasure have been buried on a random basis at n sites. For each site i there is a known unconditional probability p_i that a chest was buried there, and an excavation cost q_i. Find a search strategy (i.e., a permutation of the integers from 1 to n) that minimizes the average cost of finding the first chest.

Greedy strategy: Select the sites in nondecreasing order of q_i/p_i.

2. **A sequential search problem.**[3] An object is hidden in one of n locations. Let p_i be the prior probability that the object is in location i ($\sum_{i=1}^{n} p_i = 1$), and α_i be the probability that the object is overlooked in location i, even though it is there. The cost of each search at location i is c_i. Find a search procedure that will minimize the expected total cost of finding the object.

Greedy strategy: Choose the locations in a nondecreasing order of the ratios

$$\frac{c_i}{\pi_i(k)},$$

where $\pi_i(k)$ is the probability that the object will be found for the first time during the kth search of location i, i.e.,

$$\pi_i(k) = p_i \alpha_i^{k-1}(1 - \alpha_i).$$

3. **Job sequencing with deadlines.** Given n jobs, for each job i a deadline d_i and a profit p_i, and assuming that each job takes one unit to process, find a maximum profit subset of jobs that can be completed by their deadlines.

Greedy strategy: Select the jobs in a nonincreasing order of profits as long as they can be completed by their deadlines.

4. **Job sequencing on a single processor.** Given n jobs with a processing time p_i for job i find a sequencing that will minimize the following performance measure:

$$\frac{\sum_{k=1}^{n} F_k^{\alpha}}{n}$$

where the jobs are indexed here by their positions in the sequence, $F_k = \sum_{j=1}^{k} p_j$, and $\alpha > 0$. In the special case where $\alpha = 1$, F_k is the *mean flow time* of job k, and in this case the problem is identical to problem 3 above, and is also a special case of problem 10 below (when all weights are equal).

Greedy strategy: Sequence the jobs in order of nondecreasing processing times:

$$p_1 \leq p_2 \leq \cdots \leq p_n$$

5. **Minimizing weighted average flow-time on a single processor.** Given n jobs with processing time p_i and weight u_i for job i find a sequencing that will minimize the weighted mean flow time:

$$\bar{F} = \sum_{k=1}^{n} u_k \sum_{j=1}^{k} p_j,$$

where the jobs are indexed by their positions in the sequence.

Greedy strategy: Sequence the jobs in a nondecreasing order of p_i/u_i.

6. **Minimizing maximum job lateness on a single machine.** Given n jobs, each associated with a deadline d_i and a processing time p_i, find a sequencing that minimized the maximum job lateness or the maximum job tardiness. The *lateness* L_i of job i is

$$L_i = F_i - d_i,$$

and its *tardiness* T_i is

$$T_i = \max(0, L_i),$$

where F_i is the flow time of job i.

Greedy strategy: Sequence the jobs in nondecreasing order of their due-dates.

7. **Maximizing minimum job lateness on a single machine.** Given n jobs, each with a deadline d_i and a processing time p_i, find a sequencing that maximizes the minimum job lateness or minimum job tardiness.

Greedy strategy: Sequence the jobs in a nonincreasing order of their slack times, i.e., so that

$$d_1 - p_1 \leq d_2 - p_2 \leq \cdots \leq d_n - p_n.$$

8. **Minimizing maximum flow-time in a two-machine flow-shop.** A set of n jobs is to be processed on two machines A and B such that, for each job, the work on machine A must be completed before the work on machine B can start. Given, for each job i, the processing times A_i and B_i on machines A and B, respectively, find an ordering of the jobs on each of the two machines so that neither the precedence nor the occupancy constraints (i.e., each machine can process only one job at a time) are violated, and so that the total time to complete all the jobs (maximum flow-time) is minimized.

Greedy strategy: Sequence the jobs so that for every j job j precedes job $j + 1$ if

$$\min(A_j, B_{j+1}) < \min(A_{j+1}, B_j).$$

9. **Optimal storage on tape.** Given a set of n programs such that program i has a length l_i, find an order in which they should be placed on a tape so that the average retrieval time is minimized. The average retrieval time is given by

$$\sum_{k=1}^{n} \sum_{j=1}^{k} l_j$$

Where l_j is the length of the jth program in the selected ordering.

Greedy strategy: Select the programs in a nonincreasing order of their lengths.

ACKNOWLEDGEMENTS

The authors wish to thank Judea Pearl for reading an earlier version of this paper. They wish also to thank the referees for their helpful comments and suggestions. This work was supported in part by the National Science Foundation under Grant #IRI-881552.

REFERENCES

[1] R.W. CONWAY, W.L. MAXWELL and L.W. MILLER, 1967. *Theory of Scheduling*, Addison-Wesley, Reading, MA.

[2] R. DECHTER and J. PEARL, 1987. *Network-Based Heuristics for Constraint-Satisfaction Problems*, **Artificial Intelligence 34: 1,** 1–38.

[3] A.H. DEGROOT, 1970. *Optimal Statistical Decisions*, McGraw-Hill, New York.

[4] E. HOROWITZ and S. SAHNI, 1978. *Fundamentals of Computer Algorithms*, Computer Science Press, Rockville, MD.

[5] J.R. JACKSON, 1955. *Scheduling a Production Line to Minimize Maximum Tardiness*, UCLA Technical Report, Management Sciences Research Project, Los Angeles, CA.

[6] B. KORTE and L. LOVASZ, 1984. *Greedoids—A Structural Framework for the Greedy Algorithm*, in W.R. Pulleyblank (ed.), **Progress in Combinatorial Optimization**, Academic Press, New York, pp. 221–243.

[7] D. KRANTZ, D. LUCE, S. SUPPES and A. TVERSKY, 1971. *Foundations of Measurement*, Vol. 1, Academic Press, New York.

[8] E.L. LAWLER, 1976. *Combinatorial Optimization*, Holt, Rinehart, & Winston, New York.

[9] N. NILSSON, 1980. *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA.

[10] J. PEARL, 1983. *On the Discovery and Generation of Certain Heuristics*, **AI Magazine, No. 22–23.**

[11] H.A. SIMON and J.B. KADANE, 1975. *Optimal Problem Solving Search: All or None Solutions*, **Artificial Intelligence 6,** 235–247.

[12] D.J.A. WELSH, 1976. *Matroid Theory*, Academic Press, London.