PROPERTIES OF GREEDILY OPTIMIZED ORDERING PROBLEMS

Rina Dechter
Avi Dechter

June 1986
CSD-860048

PROPERTIES OF GREEDILY OPTIMIZED ORDERING PROBLEMS*

Rina Dechter and Avi Dechter**

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

Abstract -- The greedy method is a well-known approach for problem solving directed mainly at the solution of optimization problems. The ability to characterize greedily optimized problems (i.e., that can be solved optimally by a greedy algorithm) is important for the mechanical discovery of hueristics and for the understanding of human problem solving. This paper discusses the properties of certain classes of greedily optimized ordering problems which are not covered under currently available theories (e.g., Matroids and Greedoids).

I. INTRODUCTION AND MOTIVATION

The greedy method is a well-known approach for problem solving directed mainly at the solution of optimization problems. Greedy algorithms use an irrevocable search control regime that uses local knowledge to construct a global solution in a "hill climbing" process [7]. Usually, they involve some real-valued function defined on the states of the search space. The greedy control strategy selects the next state so as to achieve the largest possible increase in the value of this function.

Our interest in greedy algorithms is twofold. First, **greedily optimized** problems (i.e., that can be solved optimally by a greedy algorithm) represent a class of relatively easy problems. Since many heuristics used in the solution of hard problems are related to simplified models of the problem domain [9], the ability to characterize easy problems is important, particularly if the process of discovering hueristics is to be mechanized. Second, greedy schemes are probably the closest to explaining human problem-solving strategies because they require only a minimum amount of memory space and because they often produce adequate 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.)

Three examples of greedily optimized problems, along with their respective greedy strategies are given below:

1. **Minimum (Maximum) Spanning Tree.**
Given a graph $G = (V,E)$ where V is the set of vertices and E is the set of edges, and given a set of values associated with the edges, find a spanning tree with minimum sum of values.

   Greedy strategy: select edges in nondecreasing order of their values as long as they do not create a cycle.

2. **Job sequencing 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 $\frac{p_i}{u_i}$.

3. **Optimal merge patterns.**
Merge n files in pairwise manner so that the total merging cost will be minimized. Each file has weight w_i. The cost of merging files i and k into a new file j is:

$$w_j = w_i + w_k .$$

   Greedy strategy: Merge the two files which have the lowest cost, then add the resultant file to the list and repeat (Huffman procedure).

All three problems involve the task of selecting in some order, from a given set of elements (edges of a graph, jobs, files), a subset (not necessarily proper) that satisfy some property, so as to maximize (or minimize) the value of a cost function defined on all possible solutions. These problems demonstrate a useful classification of greedily optimized problems. The first is an example of a **selection problem**, where the cost function is not dependent on the order of the elements in the subset of elements which constitutes a solution. Other examples of greedily optimized selection problems are the continuous knapsack problem and the problem of storing programs on a limited amount of tape. The second is an **ordering problem**, where the value of the cost function is dependent on the order of the elements as well as on their identity. Other examples in this class are scheduling sequential search

*This work was supported in part by the National Science Foundation, Grant #DCR 85-01234

**R. Dechter is also with the Artificial Intelligence Center, Hughes Aircraft Company. A. Dechter is also with the California State University, Northridge, CA

[10] and minimizing maximum flow-time in a two machine flow-shop [1]. The third problem is an instance of a tree construction problem. These problems require the generation of a tree which induces a partial order on the set of elements.

The solution of selection and tree-construction problems by greedy algorithms has been studied extensively. Specifically, selection problems are covered by the work of Edmonds and Gale [6] on the relationships between matroid theory and greedy algorithm. Tree-construction problems are summarized by the work of Parker [8] which characterizes the set of cost functions defined on trees that are minimized using the Huffman algorithm (which is a greedy procedure).

No comparable body of knowledge exists for characterizing greedily optimized ordering problems. A generalization of the Matroid structure, called Greedoids [4], represents an attempt in this direction. However, many greedily optimized ordering problems (including the job sequencing problem mentioned above) cannot be patterned after either matroid or greedoid theories. In this paper we present a theory intended to fill this gap by characterizing cost functions that permit the optimal solution of ordering problems by a greedy algorithm.

The remainder of the paper is organized as follows. In section 2 we introduce the notion of a Problem Scheme, used to describe classes of ordering problems, and state the necessary and sufficient conditions for a problem scheme to be greedily optimized. Section 3 discusses a special class of greedily optimized problems, those which have uniform ranking functions. In section 4 we focus on a particular type of greedily optimized problems having a dominant solution and provide some necessary and sufficient conditions for problems to be in this class. The various conditions and their possible uses are illustrated in section 5 with the aid of two examples. A summary and conclusions are given in Section 6. The theorems in this paper are given without proofs. The proofs can be found in [2].

II. PRELIMINARIES: DEFINITIONS AND NOMENCLATURE

Most, if not all, ordering problems can be described in terms of a **Problem Scheme** P defined as a triplet (E,PAR,C) where:

1. E is a set of elements.
2. PAR is a set of parameters, which are real valued functions defined over the elements of E. Parameters could be single-argument functions, in which case they are denoted by $\alpha(i)$, $\beta(i)$, etc., where i indexes the elements in E. They could also be multiple-argument functions, defined over all sequences of the same number of elements and denoted by $\gamma(i,j)$, $\delta(i,j,k)$, etc.
3. C is a real valued cost function. It associates a cost with any sequence of subsets of elements in E, and is dependent only on the the parameters of the elements and on their order. The cost function is written as $C = C(\sigma)$, where σ denotes a sequence of elements in E, and σ_i is the element in position i in σ.

A problem scheme describes an entire class, or family, of problems that are very similar in character. An instance of a problem scheme P, denoted by P_I is specified by a subset of elements E_I of E along with the values of their parameters. Let n be the cardinality of E_I. The problem associated with every instance P_I of P is to find a sequence $(e_1,\cdots,e_n)$ of all the elements of E_I such that $C(e_1,\ldots,e_n)$ is maximal (or minimal) over all permutations of the elements.

For example, the problem of sequencing jobs on a single processor so as to minimize a weighted average of the flow times can be formulated in terms of a scheme (E,PAR,C) where E is a set of jobs, PAR contain two parameters, p and u, that associate with each job a processing time and an importance weight, respectively, and C is a cost function defined on any sequence σ of n elements as follows:

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

where u_i and p_i are the parameters of the i^{th} element in the sequence σ.

Definition: A **greedy rule** for P is a sequence of functions $f = \{f_j\}$

$$f_j : \Phi_j \rightarrow R \tag{2}$$

where Φ_j are all sequences of j elements.

A greedy procedure for solving any instance P_I of P using $f = \{f_i\}$ is defined as follows (maximization is assumed here and hereafter):

Greedy(P,f):

1. choose $e \in E$ such that $f_1(e) = \max\{f_1(e') \mid e' \in E\}$
2. after choosing $e_1,e_2,\ldots,e_{i-1}$ choose $y \in E - \{e_1,\ldots,e_{i-1}\}$ such that $f_i(e_1,\ldots,e_{i-1},y) = \max\{f_i(e_1,\ldots,e_{i-1},x) \mid x \in E - \{e_1,\ldots,e_{i-1}\}\}$

Definition: P is **greedily optimized** (by f) if for every problem instance P_I having n elements, the sequence $(e_1,\ldots,e_n)$ generated by Greedy(P,f) has a maximum cost over all permutations of the elements.

In the balance of this paper we restrict our attention to problems with single-argument parameters and (unless specified otherwise) to greedy rules that satisfy

$$f_i(e_1,\ldots,e_i) = f_i(e_i) = f(e_i)\,. \tag{3}$$

In that case the greedy rule is defined over the set of parameters associated with each element, i.e.,

$$f(e_i) = f(\alpha_i,\beta_i,\ldots)\,. \tag{4}$$

We will refer to such rules as **ranking functions**. Possible ranking functions for the job sequencing problem discussed above are:

$$f(u_i,p_i) = u_i \tag{5}$$

and

$$f(u_i,p_i) = p_i u_i\,. \tag{6}$$

A ranking function induces a weak order among the elements of E and the greedy procedure simply chooses elements in a nonincreasing order of f.

Definition: A ranking function is **optimizing** for some problem scheme P, if for every problem instance, P_I, it generates an optimal order. A problem scheme is said to be **greedily optimized** if it permits an optimizing ranking function.

We now give a necessary and sufficient condition for a problem scheme to be greedily optimized. The condition is stated for problems having a one-to-one cost function (i.e., no two sequences have the same cost). It should be slightly modified to hold for any cost function but this involves some details that we choose to avoid for the sake of simplicity.

Theorem 1:
A necessary and sufficient condition for a problem scheme P, having a one-to-one cost function, to be greedily optimized is that for any two elements a and b (characterized by their assigned parameters) and for all problem instances of P in which they both participate, either a precedes b in all optimal sequences or b precedes a in all optimal sequences.

□

Consider, for example, a problem scheme defined by a set of four elements $\{1,2,3,4\}$ and some cost function. Suppose that the optimal sequence for the instance defined by the set $\{1,2,3\}$ is (3,2,1), and that the optimal sequence for the instance defined by the set $\{1,2,4\}$ is (4,1,2). Then, this schema 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 the optimal solution of the problem, 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.

III. 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 $P=(E,PAR,C)$ and let f be a ranking function (not necessarily one-to-one) defined over the set of parameters of each element in E. A ranking function f is called **uniform** relative to P, if for every problem instance and for every sequence σ of elements in that problem instance,

$$C(\sigma) \geq C(\sigma^i) \text{ if } f(\sigma_i) > f(\sigma_{i+1}) \tag{7}$$

and

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

for all i, where σ^i is the sequence resulting from the exchange of the i^{th} and $i+1^{th}$ elements in σ.

Theorem 2:
A problem scheme P that has a uniform ranking function f, is greedily optimized by it.

□

We next address the question of the properties the cost function should have to guarantee the existence of a uniform ranking function. Let Σ_{ab} be the set of all the sequences, in all the instances of problem scheme 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

$$C(\sigma) \geq C(\sigma^a) \quad \forall\, \sigma \in \Sigma_{ab}$$

with strict inequality for at least one sequence, or

$$C(\sigma) \leq C(\sigma^a) \quad \forall\, \sigma \in \Sigma_{ab}\,, \tag{8}$$

with strict inequality for at least one sequence, or

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

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 on b (resp. b on a), and denote it by $a >_{p.w.} b$ (resp. $b >_{p.w.} a$). In the third case we say that C is indifferent between a and b and use the notation $a \sim_{p.w.} b$.

definition: A pairwise preferentially independent cost function C is said to be acyclic if the relation $\geq_{p.w.}$ is transitive, i.e.,

$$\text{if } a \geq_{p.w.} b \text{ and } b \geq_{p.w.} c \text{ then } a \geq_{p.w.} c\,. \tag{9}$$

This last property (i.e., that the relation "C prefers a on b" satisfies transitivity) is required to assure a weak order [5] and does not follow automatically from p.w.p.i. The following example shows that a cost function can be pairwise preferentially independent but not acyclic. Consider a problem instance 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 instance, C is pairwise preferentially independent where 3 is preferred to 2 and 2 is preferred to 1. However, 1 is preferred to 3 thus violating transitivity.

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

□

In the remainder of this paper we use the term p.w.p.i to include both properties. The next theorem suggests a possible process for identifying a p.w.p.i. cost function and for discovering its optimizing ranking function.

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

$$C(\sigma) - C(\sigma^i) = K \cdot (d(\sigma_i) - d(\sigma_{i+1})) \tag{10}$$

for all i, where K is a nonnegative function defined on σ and d is a function defined on the parameters associated with each element, then d is an optimizing ranking function.

□

The process Theorem 4 suggests is: perform symbolic manipulation on the cost function and try to express the difference between the cost of an arbitrary sequence and that of the sequence which results from exchanging the i^{th} and $i+1^{th}$ elements. If the expression satisfies condition (10) then an optimizing ranking function 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^i be a sequence resulting from the exchange of the i^{th} and $i+1^{th}$ elements. We get that

$$C(\sigma) - C(\sigma^i) = (u_{i+1}p_i - u_i p_{i+1}) = u_{i+1}u_i\,(\frac{p_i}{u_i} - \frac{p_{i+1}}{u_{i+1}})\,. \tag{11}$$

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

The gap between the sufficient condition for a problem scheme to be greedily optimized, namely, that the cost function be p.w.p.i., and the necessary condition as given in Theorem 1, is not as wide as it may seem. For instance, in problem instances of two elements the two conditions coincide, because there are only two possible sequences, one of which has maximum cost. This observation leads to another way for discovering optimizing ranking functions for any greedily optimized problem (not necessarily with a p.w.p.i. cost function).

Theorem 5:
If P is any greedily optimized problem scheme then a ranking function f is optimizing if and only if it agrees with the ordering dictated by the cost function on pairs of elements, that is, for every two elements a and b, if $C(a,b) > C(b,a)$ then $f(a) > f(b)$.

□

The proof of Theorem 5 relies on our understanding that the ranking function is optimal for all instances of a problem scheme and in particular for instances of two elements. Practically all optimizing ranking functions for problems known to be greedily optimized satisfy this property.

Theorem 5 suggests that an optimal solution to a greedily optimized ordering problem is obtained by simply sorting the elements in the order dictated by applying the cost function to pairs of elements. When a problem is not known a' priori to be greedily optimized, this method could be used to either generate candidate ranking functions or to reject the hypothesis that the problem is greedily optimized (if, for instance, the pair-wise preference turns out to be non-transitive). When the objective is to find an explicit optimizing ranking function f then, by Theorem 5, candidate functions must satisfy $C(a,b)>C(b,a)\rightarrow f(a)>f(b)$ for any two elements a and b of E.

The above observations imply that cost functions defined on pairs of elements can be used as building blocks for generating cost functions that are greedily optimized. A cost function C_2, defined on pairs of elements with k parameters each, is said to be **transitive** if for every $x,y,z \in R^k$

$$C_2(x,y) \geq C_2(y,x) \text{ and } C_2(y,z) \geq C_2(z,y) \rightarrow C_2(x,z) \geq C_2(z,x) \quad (12)$$

Theorem 6:
If a cost function C_2 defined on pairs is transitive, then a problem scheme $P=(E,PAR,C)$, where C is given by:

$$C(e_1,e_2,\ldots,e_n) = \sum_{j=1}^{n}\sum_{k=1}^{j} C_2(e_k,e_j) \quad (13)$$

is greedily optimized.

□

IV. DOMINANT RANKING FUNCTIONS

A common greedy rule is the cost function itself, i.e.,

$$f_i(e_1,\ldots,e_i) = C(e_1,\ldots,e_i)\,. \quad (14)$$

This means that at each step the algorithm chooses that element which, if it was the last, would yield best cost (i.e., Myopic policy). The Greedoid Theory [4] is concerned solely with this greedy rule.

Greedy rule (14) is optimal for some problem scheme P if any instance P_I of P has the following property: any subsequence $(e_1,\ldots,e_j)$ of E_I that has a maximal cost over all subsets of size j of E_I can be extended to a sequence of length $j+1$ that has a maximal cost over all subsets of size $j+1$ of E_I. Formally,

$$\forall (e_1,\ldots,e_i) \text{ optimal over } \Phi_i \;\exists\; e_{i+1} \in E_I - \{e_1,\ldots,e_i\} \quad (15)$$

$$\text{such that } (e_1,\ldots,e_i,e_{i+1}) \text{ is optimal on } \Phi_{i+1}\,.$$

When this condition is satisfied, the greedy rule (14) generates an optimal sequence, σ, satisfying the following property: any subsequence $(e_1,\ldots,e_j)$ of σ has a maximal cost over all subsets of size j of E_I. An optimal sequence that has this property is called a **dominant sequence**. A greedy rule that generates dominant sequences for every problem instance is said to be **dominant**. A problem scheme (or its cost function) that has a dominant greedy rule is said to be **dominantly optimized**. It is clear then, that a problem scheme that satisfies condition (15) is dominantly optimized and its dominant sequences can be obtained using the greedy rule (14).

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

$$C(\sigma) = \sum_{i=1}^{n} u_i \sum_{j=1}^{i} p_j \quad (16)$$

which, as we already know, is greedily optimized by the ranking function $f(u,p) = \frac{p}{u}$, is not dominant. To see this, consider the three-element problem instance, in which each element i is defined by its parameters (u_i, p_i):

$$e_1 = (1,4),\ e_2 = (0.5,3)\ e_3 = (5,10) \quad (17)$$

The values assigned by the ranking function to the elements are: $f(e_1) = 4$, $f(e_2) = 6$, $f(e_3) = 2$, so that the optimal sequence is: (e_2,e_1,e_3). Evaluating sequences of two elements we see that

$$C(e_2,e_1) = 8.5 \quad (18)$$

while

$$C(e_3,e_1) = 105 \quad (19)$$

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

We now identify necessary and sufficient conditions for a cost function to be dominantly optimized and discuss the relationships between these conditions and those obtained for non-dominant greedily optimized problems. We are interested in identifying sufficient conditions which are stronger than (14) and may be easier to verify. Again, we focus on greedy rules which are ranking functions.

Since a dominant ranking function is optimizing, it is determined by the order imposed by the cost function on pairs of elements and should satisfy the condition of Theorem 5, and since the cost function is defined for sequences of one element, we must have $f(e) = C(e)$ and:

$$\text{if } C(a,b) > C(b,a) \text{ then } C(a) > C(b)\,. \quad (20)$$

To test this property, one should check the behavior of $C(e)$ as a ranking function: if inconsistency is found in the order induced by $C(e)$ and the order induced by the cost on pairs

(given that both orders are well defined) then the hypothesis that the problem has a dominant optimizing ranking function can be rejected. (It still may have a non-dominant optimizing ranking function as in (17).)

The necessary conditions for a dominant cost function requires the following definitions. Let Σ^{t}_{ab} denote all sequences for which element a immediately precedes b when b is the last element in the sequence.

Definition: A cost function is **tail pairwise preferentially independent** (t.p.w.p.i.) if either

$$C(\sigma) \geq C(\sigma^{a}) \ \forall \ \sigma \in \Sigma^{t}_{ab} \tag{21}$$

with strict inequality for at least one sequence, or

$$C(\sigma) \leq C(\sigma^{a}) \ \forall \ \sigma \in \Sigma^{t}_{ab}$$

with strict inequality for at least one sequence, or

$$C(\sigma) = C(\sigma^{a}) \ \forall \ \sigma \in \Sigma^{t}_{ab}$$

In the first two cases we say that C prefers a on b (resp. b on a) t.p.w., and denote it by $a >_{t.p.w.} b$ (resp. $b >_{t.p.w.} a$). In the third case we say that C is indifferent between a and b t.p.w and use the notation $a \sim_{t.p.w.} b$. If the relation is acyclic then it induces a weak order on the elements of E.

Definition: A cost function is **order preserving** if

$$\forall\, i \ \text{if } C(e_1,\ldots,e_i) \geq C(e_1',\ldots,e_i') \text{ and } C(e_{i+1}) \geq C(e_{i+1}') \tag{22}$$
$$\text{then } C(e_1,\ldots,e_i,e_{i+1}) \geq C(e_1',\ldots,e_i',e_{i+1}').$$

Theorem 7:
Let C be a cost function in $P = (E, PAR, C)$ which is both order preserving and t.p.w.p.i. If $C(a) \geq C(b)$ implies that $C(a,b) \geq C(b,a)$, then C is greedily optimized by a dominant ranking function.

□

It is easy to show that a cost function which is t.p.w.p.i. and order preserving is also p.w.p.i. Therefore, a cost function satisfying the conditions of Theorem 7 is p.w.p.i. and has a dominant optimizing ranking function. For example, the function

$$\sum_{i=1}^{n} p_1 p_2 \cdots p_i \tag{23}$$

is both p.w.p.i. and order preserving. The ranking function $f(p) = p$ results in a dominant optimal solution. For C(e) to be an optimizing ranking function, it is important to verify that the order induced by $C(e)$ agrees with that induced by the p.w.p.i. property (i.e. the last condition of Theorem 7). For instance, the cost function

$$\sum_{i=1}^{n} p_i{}^{i} \tag{24}$$

is both p.w.p.i. and order preserving. However, the ordering implied by $C(p)$ results in a decreasing sequence while the (optimal) ordering implied by exchanging adjacent elements is nondecreasing in p. Indeed, this cost function is greedily optimized with $f(p) = -p$, but is not dominantly optimized.

Up to now all the "nice" greedily optimized cost function properties we described required the cost function to be p.w.p.i. (or t.p.w.p.i.). Since this property is not a necessary condition it is natural to look for cost functions that are greedily optimized but not p.w.p.i. We show that the p.w.p.i. property may indeed be replaced by another strong property of cost functions.

Definition: A cost function is **strong order preserving** if $\forall i$ and $\forall\, x, y \in E$

$$C(e_1,\ldots,e_i) > C(e'_1,\ldots,e'_i) \rightarrow \tag{25}$$
$$C(e_1,\ldots,e_i,x) > C(e'_1,\ldots,e'_i,y)$$

For example, A lexicographical order among sequence of integers is strong order preserving. The cost function

$$C(e_1,\ldots,e_n) = \sum_{i=1}^{n} |e_i - e_{i+1}| \, 10^{n-i} \tag{26}$$

for $e_i \in [0,10)$, is strong order preserving but not p.w.p.i.

Theorem 8:
If a cost function is strong order preserving then it is dominantly optimized.

□

V. EXAMPLES

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

The first problem is **Sequencing Jobs according to due-dates.** Given n jobs, each associated with a deadline d_i and a processing time p_i, find an optimal sequencing that minimized the maximum job lateness defined by:

$$\max\{F_i - d_i\}$$

Where F_i is the flow time of job i defined by:

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

Jackson [3] had shown that the problem is p.w.p.i. and suggested the due-dates as the ranking function.

Let $e_1 = (p_1, d_1)$ and $e_2 = (p_2, d_2)$ 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

$$C(e_1,e_2) > C(e_2,e_1) \rightarrow d_1 > d_2, \tag{27}$$

that is,

$$\max\{p_1 - d_1, p_1 + p_2 - d_2\} > \max\{p_2 - d_2, p_2 + p_1 - d_1\} \tag{28}$$
$$\rightarrow d_1 > d_2.$$

The cost of one element is

$$C(e_1) = p_1 - d_1. \tag{29}$$

This cost does not provide the same ordering as the due-dates and, therefore, the cost function is greedily optimized but not dominant.

The second problem is **Minimizing maximum Flow-time in a two-machine flow-shop.** Let (A_i, B_i) be a pair associated with each job. A_i is the work to perform on the first machine of the shop and B_i is the work to be performed 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] that the maximum flow-time is minimized if job j precedes job $j+1$ when

$$\min\{A_j, B_{j+1}\} < \min\{A_{j+1}, B_j\} . \quad (30)$$

Looking only on two-job problems, it is easily verified that the ordering dictated by (30) 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 (e_1,e_2) is:

$$C(e_1,e_2) = A_1 + \max\{A_2, B_1\} + B_2 . \quad (31)$$

It can be shown that

$$A_1 + \max\{A_2, B_1\} + B_2 < A_2 + \max\{A_1, B_2\} + B_1 \quad (32)$$

iff

$$\min\{A_1, B_2\} < \min\{A_2, B_1\} \quad (33)$$

This criterion is the one known in the literature. From the transitivity property of the order induced by (33) we know that there exist a ranking function f that induces an individual order. After some manipulation such a ranking function can indeed be formed. It can be shown that if:

$$\min\{A_1, B_2\} < \min\{A_2, B_1\} \quad (34)$$

then

$$\frac{sign(A_1 - B_1)}{\min(A_1, B_1)} < \frac{sign(A_2 - B_2)}{\min(A_2, B_2)} . \quad (35)$$

Therefore the function

$$f(A_i, B_i) = \frac{sign(A_i - B_i)}{\min(A_i, B_i)} \quad (36)$$

is a uniform ranking function for the problem. The cost for one element only is $A_1 + B_1$ and it does not coincide with the ranking function (36). We can therefore conclude that this cost function is not dominant.

VI. SUMMARY AND CONCLUSIONS

This paper discusses the conditions for optimization ordering problems to be greedily optimized and the relationships between these conditions and the corresponding optimizing greedy rules. Within the greedily optimized problems, we identify a class of pairwise preferentially independent and acyclic (p.w.p.i) problems that are optimized via a uniform ranking function, and show that several well known examples fall into this class. We provide a procedure which, in certain cases, can be used to verify the p.w.p.i property and to identify uniform ranking functions.

Observing that the optimal solutions to many greedily optimized selection and tree-construction problems are dominant, we study this property in connection with ordering problems. This property is of particular interest because (as human intuition usually dictates) it makes the cost function itself an optimizing greedy rule. We show that p.w.p.i problems do not necessarily have dominant solutions, and that dominantly optimized problems are not necessarily p.w.p.i. We also give sufficient conditions for a problem to be both p.w.p.i and dominantly optimized. The relationship between the different classes is presented in Figure 1.

ACKNOWLEDGEMENT

We would like to thank Judea Pearl for contributing some of the ideas developed in this paper.

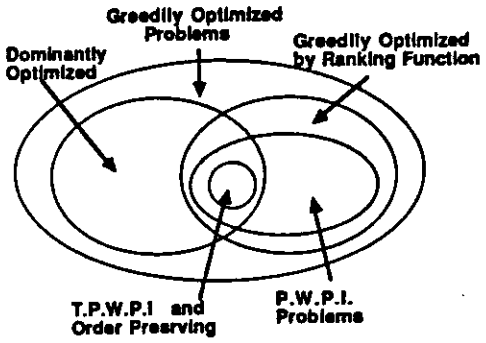

Figure 1 - Relationships Among Classes of Greedily Optimized Problems

REFERENCES

[1] Conway, R.W., W.L. Maxwell, and L.W. Miller, *Theory of scheduling*, Reading, Massachusetts: Addison-wesley Publishing company, 1967.

[2] Dechter, R., "Properties of greedily optimized problems," UCLA, Summer quartely , Los Angeles, Cal, Tech. Rep. Computer Science department, 1985.

[3] Jackson, J. R., "Scheduling a production line to minimize maximum tardiness," UCLA, Los Angeles, Cal, Tech. Rep. Management Sciences Research Project, 1955.

[4] Korte, B. and L. Lovasz, "Mathematical structures underlying greedy algorithms.," in *Proceedings Fundamentals of computation theory*, Szeged, Hungary: 1981, pp. 205-210. Springer Verlang's lecture Notes in Computer-Science #117.

[5] Krantz, D., D. Luce, S. Suppes, and A. Tversky, *Foundations of Measurement, Vol 1.*, New-York and London: Academic Press, 1971.

[6] Lawler, E.L., *Combinatorial optimization, Networks and Matroids*: Holt, Rinehart, and Winston, 1976.

[7] Nilsson, N., *Principals of Artificial Intelligence*, Palo Alto, California: Tioga, 1980.

[8] Parker, D.S., "Conditions for optimality of the Huffman algorithm.," *SIAM Journal of Computing*, Vol. 9, No. 3, 1980.

[9] Pearl, J., "On the discovery and generation of certain heuristics," *AI Magazine*, No. 22-23, 1983.

[10] Simon, H. A. and J. B. Kadane, "Optimal problem solving search: all or none solutions.," *Artificaial Intelligence*, Vol. 6, 1975, pp. 235-247.

Dechter, R., & Dechter, A., "Properties of Greedily Optimized Ordering Problems," UCLA Cognitive Systems Laboratory *Technical Report CSD-860048 (R-57); Proceedings, 1986 Canadian AI Conference*, Montreal, May 1986.