ACCELERATED GREEDY ALGORITHMS FOR MAXIMIZING SUBMODULAR SET FUNCTIONS

Michel MINOUX
Ecole Nationale Supérieure de
Techniques Avancées - PARIS - France

Abstract :

Given a finite set E and a real valued function f on $\mathcal{P}(E)$ (the power set of E) the optimal subset problem (P) is to find $S \subseteq E$ maximizing f over $\mathcal{P}(E)$. Many combinatorial optimization problems can be formulated in these terms. Here, a family of approximate solution methods is studied : the greedy algorithms.

After having described the standard greedy algorithm (SG) it is shown that, under certain assumptions (namely : submodularity of f) the computational complexity of (SG) can often be significantly reduced, thus leading to an accelerated greedy algorithm (AG). This allows treatment of large scale combinatorial problems of the (P) type. The accelerated greedy algorithm is shown to be optimal (interms of computational complexity) over a wide class of algorithms, and the submodularity assumption is used to derive bounds on the difference between the greedy solution and the optimum solution.

I - INTRODUCTION

Among the great variety of existing methods for the solution of combinatorial optimization problems, the so-called "greedy algorithms" are conceptually the simplest and perhaps the most widely used in practise. One of the first algorithm of this type seems to be the famous KRUSKAL'S algorithm for the minimum spanning tree problem [4] . More generally, EDMONDS [5] has shown that these methods provide optimal solutions for the class of problems involving the search for a maximum (minimum) weight basis of a matroïd.

Most of real-world problems, of course, do not belong to this class, but it has long been noticed by many authors [1] [10] [11] that greedy algorithms often lead to very good approximate solutions. These observations had not been explained until a recent publication [15] where it is shown that the quality of greedy solutions are strongly related to a specific property of set functions called : submodularity.

Our purpose, here, is to carry on the study of combinatorial problems involving the maximization of submodular set functions. In particular we show that, when the submodularity property holds, the number of computations necessary to get a greedy solution can often be significaty reduced, and we describe an "accelerated" greedy algorithm which allows the treatment of large scale combinatorial problems of type (P).

Computational experiments illustrate the efficiency of the procedure. For example, optimum network problems on graphs involving about 180 nodes and 500 edges can be solved within a few minutes, using the accelerated greedy algorithms ; for the same result a standard greedy algorithms would require several hours of computations on the same computer (see [1]).

These experimental results are confirmed theoretically by proving that the accelerated greedy algorithms is optimal, in terms of computational complexity, among a wide class of algorithms (see section IV).

Finally, it is shown how submodularity can be used to derive bounds for the difference between the greedy solution and the optimum solution of a given problem. These bounds, being specific of each particular problem, are usually tighter than those in [15] .

II - THE OPTIMAL SUBSET PROBLEM : A GENERAL FORMULATION OF COMBINATORIAL OPTIMIZATION PROBLEMS.

Let $E = (e_1 , e_2 \dots e_n)$ be a finite set with n elements and $f : \mathcal{P}(E) \to \mathbb{R}$ a real function on $\mathcal{P}(E)$ the power set of E. Such a function will be called a set function.

The optimal subset problem (called problem (P)) is to determine a subset S^* of E such that :

$$f(S^*) = \max_{S \subset E} f(S)$$

It is easily shown that (P) is equivalent to the general mixed or pure integer program with bounded variables.

In view of this fact we shall restrict ourselves to problem (P). First, let us take a few examples :

Example 1 : minimum spanning tree [4]

$G = [X, U]$ is a (unoriented) connected graph with node set X and edge set U.

Associated with each edge $u \in U$ of G is a real number w(u) called the weight of u.

Let $S \subset U$ be a subset of edges, and $w(S) = \sum_{u \in S} w(u)$.

We want to determine a minimum spanning tree of G. If we take E as the set of edges U, and :

. $f(S) = -w(S)$ if $G_S = [X, S]$ is connected

. $f(S) = -\infty$ otherwise,

the problem is converted into (P).

Example 2 : center location problem : [8] [9] [10] .

This type of problem is frequently encountered in practise. In its simplest form, it may be stated as follows.

A set $I = \{1, 2, \dots M\}$ of M customers must be supplied from a certain (unknown) set of centers. A (finite) list of all possible locations for the centers : $E = \{e_1, e_2 \dots e_n\}$ is given.

For $j \in J$, $\alpha_j > 0$ is the cost of installation of a center at location e_j .

For every $i \in I$, and for $e_j \in E$, the cost $\gamma_{ij} > 0$ of assignment of customer i to location e_j is given.

If the subset $S \subset E$ of location is chosen, the total cost of the solution is the sum of :

. the cost of the centers : $\alpha(S) = \sum_{e_j \in S} \alpha_j$

. the cost $\gamma(S)$ of the assignment of customers to centers :

$$\gamma(S) = \sum_{i \in I} \operatorname*{Min}_{e_j \in S} \{\gamma_{ij}\}$$

The problem is then to find an optimal subset of centers $S \subset E$ such that the cost is minimal.

This problem is easily recognized as a problem (P) with function f defined by :

$$f(S) = -[\alpha(S) + \gamma(S)] \qquad \forall S \subset E$$

Example 3 : the optimum network problem [1] [7]

Let $G = [X, U]$ be a connected (unoriented) graph with node set X and edge set U.

With each edge $u \in U$ we associate two numbers :

$\delta(u)$ = fixed cost of link u

$\rho(u)$ = cost of link u per unit capacity.

$(\rho(u) \geq 0 \ \forall u)$.

For any $S \subset U$, the cost of subgraph : $G_S = [X, S]$ of G is the sum of two costs.

. A fixed cost $\delta(S) = \sum_{u \in S} \delta(u)$

. A variable cost $\rho(S) = \sum_{i \neq j} \lambda_{ij} \cdot l_{ij}(S)$

where $l_{ij}(S)$ is the length of the shortest chain between nodes i and j in G_S, with lengths $\rho(u)$ on the edges. ($l_{ij}(S) = +\infty$ if i and j are not connected in subgraph G_S). The λ_{ij} are given non negative point to point capacity requirements. The problem is to determine a subgraph $G_{S^*} = [X, S^*]$ for which the cost is minimum. Many problems of network engineering (computer networks, telephone networks, transportation networks ...) may be formulated in this way.

It reduces to problem (P) by taking E = U (the edge set of G) and by defining function f by :

$$f(S) = -\left[\delta(S) + \rho(S)\right] \qquad \forall\, S \subset U$$

The optimum network problem belongs to the class of the so-called : fixed cost linear programming problems [11] [12] [13] ; these may be similarly re-formulated as type (P) problems.

III - THE STANDARD GREEDY ALGORITHM

In this section we show how greedy algorithms can be used to provide exact or approximate solutions to problems (P).

Starting with solution $S = \emptyset$, for instance, a subset S of E is grown, little by little, by successively adding some elements of E which do not belong to the current S. At each step, that element e_i is added for which the increase in cost :

$f(S + \{e_i\}) - f(S)$ is maximum.

This may be summarized by the following procedure.

Standard greedy algorithm (SG)

```
(a) Take $S^\circ = \emptyset$ ; itération k = 0
(b) at step k, $S^k$ is the current solution of cost $f(S^k)$ and $|S^k| = k$
(c) for all $e_i \in E - S^k$ , compute :
    $\Delta^k(e_i) = f(S^k + \{e_i\}) - f(S^k)$.
(d) select $e_{io}$ such that $\Delta^k(e_{io}) = \max_{e_i \in E-S^k} \Delta^k(e_i)$ .
    if $\Delta^k(e_{io}) \leqslant 0$ Terminated = the current solution $S^k$ is (loccaly) optimal.
    otherwise :
(e) Set $S^{k+1} \leftarrow S^k + \{e_{io}\}$
    $k \leftarrow k + 1$
    and return to (b).
```

The solution determined by (SG) will be called a greedy solution of (P). Notice that, unless the minimum $\Delta^k(e_{io})$ is unique at each step k of the algorithm, the greedy solution is not necessarily unique.

For the minimum spanning tree problem, [4] (SG) is recognized as KRUSKAL II algorithm.Thus there exists a class of problems for which (SG) produces an optimum solution : these are one-matroid optimization problems (involving the search for a maximum or minimum weight basis of a matroid).

Though most combinatorial problems arising in practise do not belong to this class, it has long been recognized that greedy algorithms often lead to very good approximate solutions. This is the case for example 2 above (locations of centers [8][9][10][12]), example 3 (the optimum network problem [1][7]) and more generally, for some classes of fixed charge linear programming problems [11][12][13] (example 3). See [1] for some computational results.

The computational complexity of (SG) can easily be evaluated, in terms of the number of calculations of f.

At the first step of the algorithm, $S^\circ = \emptyset$ and the number of such calculations is : n . At the second step : n - 1 . At the kth step , n - k + 1 . Hence, the total number of calculation may be bounded above by =

$$n + (n - 1) + (n - 2) \dots + 1 = \frac{n\,(n+1)}{2}$$

And the computational complexity of (SG) is $\mathcal{O}(n^2)\ .\ \mathcal{O}(f)$ where $\mathcal{O}(f)$ is the complexity of one f(S) calculation.

For the center location problem (example 2) with N centers and M customers : $\mathcal{O}(f) = M.N$, hence (SG) is $\mathcal{O}(M\ N^3)$

For the optimum network problem (example 3) the calculation of f(S) for any given S involves the determination of all shorst paths in a N-node unoriented graph, thus : $\mathcal{O}(f) = \mathcal{O}(N^3)$. Then, if M is the number of edges, (SG) is : $\mathcal{O}(M^2\ N^3)$.

These examples show that, even though greedy algorithms are usually much faster than enumerative methods (like Branch and Bound), they may prove inefficient in solving large scale problems of type (P).

IV - AN ACCELERATED GREEDY ALGORITHM

In this section, we show how the efficiency of the standard greedy algorithm can be improved thus leading to an "accelerated greedy algorithm" (AG). When function f is submodular , we prove that the solutions given by (SG) and (AG) are identical.

A set function $f : \mathcal{P}(E) \to \mathbb{R}$ is said to be submodular if and only if, for every $S \subset E$, $T \subset E$:

$$f(S) + f(T) \geqslant f(S \cup T) + f(S \cap T). \qquad (1)$$

(see [20])

We now give another more useful characterization of submodular set functions in terms of the differences :

$$\delta(A, e_i) = f(A + \{e_i\}) - f(A).$$

Theorem 1 :

the two following properties are equivalent :

- function $f : \mathcal{P}(E) \to \mathbb{R}$ is submodular ;
- for every $A \subset E$, $B \subset A$; $e_i \notin A$:

$$\delta(A, e_i) \leqslant \delta(B, e_i) \qquad (2)$$

Proof : see [2] or [15] for instance.

For set functions, submodularity plays a role which is, in a sense, similar to that of concavity for ordinary functions.

In fact, relation (2) can be interpreted by saying that the benefit incurred by adding element e_i to a given solution A can only decrease if others elements are first added to A.

It can be shown (cf [2]) that submodularity holds for example 1 (minimum spanning tree) and example 2 (optimal location of centers) above, and for a number of other important combinatorial problems (see ref [15] for a list of such examples).

It is easily seen that it does not hold for general fixed charge linear programming problems. However for the optimum network problem it can be shown [1][2] that there exists a submodular function f' which closely approximates f.

The accelerated greedy algorithm (AG)

```
(a) Take $S^\circ = \emptyset$ as a starting solution
        step $k = 0$
(b) for every $e_i \in E$, compute :
        $\Delta(e_i) = f(\{e_i\}) - f(\emptyset)$
(c) At step k, let $S^k$ be the current solution, of cost $f(S^k)$
    Select $e_{io} \in E - S^k$ such that :
        $\Delta(e_{io}) = \underset{e_i \in E - S^k}{\text{Max}} \{\Delta(e_i)\}$
If $e_{io}$ has already been selected once at step k set : $\delta = \Delta(e_{io})$ and go to (e)
(d) compute :
        $\delta = f(S^k + \{e_{io}\}) - f(S^k)$
        and set : $\Delta(e_{io}) \leftarrow \delta$
        if $\delta < \underset{\substack{e_i \in E - S^k \\ e_i \neq e_{io}}}{\text{Max}} \{\Delta(e_i)\}$
        return to (c)
        otherwise :
(e) if $\delta \leq 0$ STOP : solution $S^k$ is (locally) optimal - Otherwise ( $\delta > 0$) :
(f) Set : $S^{k+1} \leftarrow S^k + \{e_{io}\}$
          $\Delta(e_{io}) = 0$
          $k \leftarrow k + 1$
          and return to (c).
```

Theorem 2 (convergence of (AG))

When function f is submodular, the accelerated greedy algorithm (AG) produces a greedy solution.

Proof : refer to [2].

Remark : if the greedy solution is unique, then the solutions produced by (SG) and (AG) are identical.

Applied to the optimum network problem (example 3) (AG) requires on the average only 2 or 3 calculations of f at each level. In this case the computational complexity : $\frac{n^2}{2}\,\mathcal{O}(f)$ reduces to about $3n.\,\mathcal{O}(f)$, which shows an improvement by a factor n/6 (on the average). It follows that the accelerated greedy algorithm can be used to solve large scale combinatorial problems of the (P) type, for which even standard greedy algorithms would prove inefficient. For instance, examples of optimum network problems on graphs with about 180 nodes and 500 edges have been solved within a few minutes by (AG) (*). (See [1][2] for computational results). On the same problems, (SG) would have been 50 to 100 times slower !

The above results provide an experimental confirmation of the efficiency of (AG) when applied to some real problems of the (P) type. Unfortunately, these cannot be demonstrated theoretically and it is easy to build worst-case examples for which (AG) requires the same number of computations as (SG).

However, what will be shown next is that these are, in a sense, the best possible results, by proving that, among a wide class of algorithms, (AG) is optimal in terms of number of calculations.

For any $S \subset E$, define a neighbourhood of S as a set of solutions $S' \subset E$ such that : $S' \supset S$ and $|S'| = |S| + 1$, and let's design by $\mathcal{L}$ the class of local optimization algorithms (LOCO algorithms as termed by EDMONDS [5]) :

(a) step $k = 0$, $S^\circ = \emptyset$ (starting solution)

(b) at step k , S^k is the current solution. Determine S^* such that :

$$f(S^*) = \operatorname*{Max}_{S \in \mathcal{N}(S^k)} f(S)$$

where $\mathcal{N}(S^k)$ is any neighbourhood of S^k.

(c) if $f(S^*) \leq f(S^k)$ STOP. Otherwise :

set : $S^{k+1} \leftarrow S^*$, $k \leftarrow k + 1$ and return to (b).

We now state :

Theorem 3 :

Algorithm (AG) is optimal in the class of LOCO algorithms ie : no other algorithm in $\mathcal{L}$ can provide a greedy solution to any submodular maximization problem with fewer computations of f.

<u>Proof</u> : Let (P1) be any submodular maximization problem, and suppose that (AG) requires p computations of f for getting a greedy solution. Consider an algorithm $(A') \in \mathcal{L}$ which computes a greedy solution of (P1) within a number $q < p$ of f computations and let's show that there exists at least one submodular maximization problem for which (A') doesn't provide a greedy solution. Since (A') performs fewer computation than (AG), there is at least one level at which the number of computations is lower ; let k be the first level at which this occurs. At this level, (A') will add an element e_{io} such that :

$\Delta^k (e_{io})$ is not the maximum of all $\Delta(e_i)$. Hence, there exists $e_{i1} \neq e_{io}$, with $\Delta(e_{i1}) > \Delta^k (e_{io})$.

Since the submodularity condition only requires that $\forall e_i$: $\Delta(e_i) \geqslant \Delta^{\ell}(e_i)$ at each step $t \geqslant \ell$ we can build a new problem (P'1) which consists in maximizing a submodular function f' such that :

(a) $\Delta'^{\ell} (e_i) = \Delta^{\ell}(e_i)$

$\forall \ell = 1 \ldots k , \quad \forall e_i \notin S^{\ell} + \{e_{i1}\}$

(b) $\forall \ell = 1 \ldots k$: $\Delta'^{\ell} (e_{i1}) = \Delta (e_{i1})$ at step ℓ of (AG) applied to problem (P1); this implies in particular :

$\Delta'^{k}(e_{i1}) > \Delta'^{k}(e_{io}) = \Delta^{k}(e_{io})$

(c) $f'(S^k + \{e_{io}\} + \{e_j\}) = -\infty \quad \forall e_j \notin S^k + \{e_{io}\}$

Applied to (P'1), algorithm (A') would stop at solution $S^k + \{e_{io}\}$, when at the same time (AG) would get a greedy solution $S^* \supset S^k + \{e_i\}$, hence such that :

$S^* \neq S^k + \{e_{io}\}$ and :

$$f'(S^*) \geqslant f'(S^k + \{e_{i1}\}) = f'(S^k) + \Delta'^{k}(e_{i1}) > f'(S^k) + \Delta'^{k}(e_{io}) = f'(S^k + \{e_{io}\})$$

This clearly demonstrates that no algorithm in $\mathcal{L}$ performing fewer calculations than (AG) can insure a greedy solution for any submodular maximization problem. $\square$

As a consequence of the above result, it is seen :

(a) that the number of f computations required by (AG) may be taken as a measure of the difficulty of submodular maximization problems ;

(b) the worst-case examples mentionned above (on which (AG) achieves a maximum number of calculations) must be actually recognized as problems of intrinsical difficulty.

Our last concern is to show how submodularity can be used to derive bounds on the difference between the optimal solution and the greedy solution of (P)

Suppose that f is submodular, and let $\bar{S}$ ($|\bar{S}| = p$) be the greedy solution produced by (SG) or (AG).

For i = 1 .... n, consider the quantites :

$\delta_i = \delta(\emptyset, e_i) = f(\{e_i\}) - f(\emptyset)$

and suppose that the elements e_i of E are considered in decreasing order :

$\delta_1 \geqslant \delta_2 \geqslant \dots\dots \geqslant \delta_n$

then :

<u>Theorem 4</u> :

1) $f(\emptyset) + \sum_{i=1}^{p} \delta_i$

is a upper bound on the value of an optimal solution of cardinality p.

2) $f(\emptyset) + \sum_{i/\delta_i \geqslant 0} \delta_i$

is a upper bound on the value of the true optimum of (P)

<u>Proof</u> : see [2] .

In [15] submodularity was used to derive general worst-case bounds between an optimal solution and a greedy solution. Theorem 3 above is however interesting in that it provides bounds which, being specific of each particular problem, are generally tighter than those given in [15].

It is clear, that for combinatorial problems of complex structure, it will be often impossible to prove, by theoretical considerations, that the submodularity assumption holds. However, submodularity may be tested, experimentally, by comparing the solution produced by (SG) and (AG), on medium size examples. For large scale problems, anyway, it is likely that only the accelerated greedy algorithm (AG) will be praticable.

REFERENCES

[1] MINOUX (M.) "Multiflots de coût minimum avec fonctions de coût concaves". Annales des télécommunications, 31, n° 3-4, (1976)

[2] MINOUX (M.) "Algorithmes gloutons et algorithmes gloutons accélérés pour la résolution des grands problèmes combinatoires". Bulletin de la Direction des Etudes et Recherches - EDF (France) Série C N° 1 (1977) pp.41-58

[3] LAURIERE (J.L.) "Un langage et un programme pour énoncer et résoudre des problèmes combinatoires" - Thèse, doc.ès sciences. Université PARIS VI - Mai 1976

[4] KRUSKAL (J.B.) "On the shortest spanning subtree of a graph and the travelling salesman problem" - Proc. Am. Math. Soc. 2 (1956) pp. 48-50

[5] EDMONDS (J.) "Matroïds and the greedy algorithm" - Mathematical programming 1, (1971), pp. 127-136.

[6] GONDRAN (M.) "L'algorithme glouton dans les algèbres de chemins" Bulletin Dir. Et. Rech. EDF Série C N° 1 (1975) pp.25-32

[7] BILLHEIMER (J.W.) GRAY (P.) "Network design with fixed and variable cost elements".
Transp. Science 7, n° 1 (1973) pp. 49-74

[8] LEGROS (J.F.) MINOUX (M.) OUSSET (A.) "Local networks optimization" Proc. ISSLS Conference London (May 1976)

[9] COOPER (L.) "The transportation location problem" Ops. Res. 20, n° 1 (1972) pp. 94-108

[10] KUENNE (R.E.) SOLAND (R.M.) "Exact and approximate solutions to the multisource weber problem". Mathematical programming 3 (1972) pp.193-209.

[11] STEINBERG (D.I.) "The fixed charge problem" - Nav. Res. Log. Quart. 17 (1970) pp. 217-236.

[12] BALINSKY (M.L.) "Fixed Cost Transportation Problems" - Nav. Res. Log. Quart. 8 (1961) pp. 41-54

[13] MALEK-ZAVAREI (M.), FRISCH (I.T.) "On the fixed cost flow problem" Intern.Journal Control 16, n° 5, (1972), pp. 897-902

[14] EDMONDS (J.) "Submodular functions, matroïds, and certain polyhedra" in : Combinatorial structures and their applications, R. Guy ed. pp.69-87 Gordon and Breach 1971.

[15] FISCHER (M.L.) NEMHAUSER (G.L.) WOLSEY (L.A.) "An analysis of approximations for maximizing submodular set functions" IX Internat. Symp. on Mathematical Programming BUDAPEST Hungary (1976).

[16] SAVAGE (S.L.) "Some theoretical implications of local optimization" Mathematical Programming 10 (1976) pp. 354-366.