Empirically Adjusted Greedy Algorithms (EAGH): A new approach to solving combinatorial optimisation problems

Albert Corominas

IOC-DT-P-2005-22
Octrubre 2005

Empirically Adjusted Greedy Algorithms (EAGH): A new approach to solving combinatorial optimisation problems[1]

Albert Corominas

Abstract

A greedy heuristic to solve a given combinatorial optimisation problem can be seen as an element of an infinite set of heuristics, H, which is defined by a function that depends on several parameters. We propose a procedure for determining the best element of H for a set of instances of the combinatorial optimisation problem. The procedure consists essentially in applying a direct non-linear optimization algorithm to a function of the parameters that characterise H.

Keywords: greedy algorithms, combinatorial optimisation, non-linear optimisation.

1. Introduction

Despite the spectacular advances that have been made in the last few years in the field of exact procedures for solving combinatorial optimisation problems, particularly those based on mixed integer linear programming (Bixby, 2002), heuristics still have a very important role to play.

Greedy heuristics are one of the most popular types of heuristics. Nevertheless, the theoretical basis of these heuristics is generally very weak, which has led to a wide range of greedy algorithms for solving a single problem. This raises the question of which of the various greedy algorithms is best and how to determine the answer to it.

The most common way of comparing a finite number of heuristics consists in applying them to a large number of instances generated by a specific procedure (e.g., using the uniform distribution to generating the processing times in the flow-shop problem or in assembly line balancing problems). Of course, the conclusions reached cannot be extended without further ado to other types of instances (Watson et al., 1999); then, one can consider, as the author does, that the aforesaid question (i.e., which is the best greedy algorithm for a problem) is an ill-posed one, since it only make sense referred to a well-specified set of instances of the problem and not , instead, referred to the set of all possible instances. Indeed, comparing heuristics is a non-closed topic that still deserves attention from the scientific community (Barr et al., 1995, Hooker, 1995 and Rardin and Uzsoy, 2001).

In this paper, a procedure for selecting greedy heuristic algorithms is proposed. Given:

(i) a combinatorial optimisation problem that has an objective function, *f* ;

[1] This research has been supported by projects DPI2004-05797 and DPI2004-03472 (Ministerio de Educación y Ciencia, Spain, and FEDER).

(ii) an infinite set of heuristics, *H,* defined by a function, *h,* which depends on several parameters;

(iii) a training set of instances extracted from a given population

an optimum element of *H*, for the training set (and hopefully for the population which this set is extracted from) is sought.

The sum of the values of objective function *f* that correspond to the solutions obtained using a heuristic belonging to *H* for the instances of the training set is, therefore, a function, φ, of the parameters that *h* depends on. This leads to an optimisation problem with real variables (the parameters), to minimise the aforementioned function, φ.

The rest of the paper is divided up as follows: Section 2 includes a brief discussion of greedy heuristics; Section 3 describes our proposal for a procedure for selecting heuristics, which we refer to as empirically adjusted greedy algorithm (EAGH), and Section 4 gives an overview of possible future research.

2. Greedy heuristics

Many of the heuristic algorithms put forward in the literature are of the greedy type. Greedy algorithms are, moreover, an essential component of GRASP (Greedy Randomized Adaptive Search Procedure) algorithms (Feo and Resende, 1995). Notwithstanding this, it is not easy to find a satisfactory definition of a greedy heuristic.

The entry under “Greedy Algorithm” (Gass and Harris, eds., 1996) reads as follows: “A heuristic algorithm that at every step selects the best choice available at that step without regard to future consequences. A greedy method never rescinds its choices or decisions made earlier. A greedy method is usually applied to an optimisation problem for which the method attempts to determine an optimal solution (least cost, maximum value), with no guarantee that the optimal solution will be found. Kruskal and Prim’s minimum spanning tree algorithms are greedy methods that do produce an optimal solution.” As one can see, the definition can be interpreted in slightly different ways, since greedy algorithms are said to be heuristic (that is, not exact), but both of the examples provided are in fact exact algorithms; consequently, it remains unclear whether the algorithms that should be classed as greedy are all the algorithms that have certain traits or only, among them, those that are heuristic. Moreover, the advisability of including in the definition “the best choice available at that step without regard to future consequences” is debatable, because, as will be discussed below, it restricts unnecessarily the scope of the concept of greedy algorithm.

Similarly, according to Silver (2004), “a special constructive approach is the so-called greedy method, where, at each step, the next element of the solution is chosen so as to give the best immediate benefit (highest profit contribution or lowest cost). The greedy approach is very similar to a sequential myopic perspective [...].”

An Internet search provides one with dozens of more or less formal definitions of a greedy algorithm that are very similar to those cited.

At each iteration of a greedy algorithm an irreversible decision is taken. This decision is based on the values of an *indicator* that is associated with each of the possible decisions. This indicator is calculated using the data of the instance to be solved and, if the algorithm is adaptive, takes into account the consequences of the decisions taken at previous iterations.

In our opinion, what characterises a greedy algorithm is that it arrives at a solution by means of a finite number of iterations and that in each of these iterations an irreversible decision is taken. The decision may consist in incorporating an element into a set or in eliminating it (the set may be the vertices, arcs or edges of a graph or an object in the Knapsack Problem, etc.). However, it may also consist in modifying the solution to a problem to make it fulfil a constraint that it did not previously fulfil (e.g., Clarke & Wright heuristic for the Vehicle Routing Problem: Clarke and Wright, 1964).

Instead, a heuristic need not be myopic or shortsighted to be greedy, although many greedy heuristics are. There are at least three (related) ways of avoiding shortsightedness in greedy algorithms:

i) Taking into account the opportunity costs associated to a decision.

For instance, let us consider the problem of the maximum weight internally stable set in an undirected graph. Let us define a greedy heuristic, GH1, that consists in ordering the vertices of the graph from the highest to the lowest weight, and, following the order established, incorporating the vertices into the set that are compatible with those that have already been incorporated. Alternatively, we might define a heuristic, GH2, into which the vertex incorporated at each iteration k, is, among the compatible with those already incorporated, that which maximises the value of the quotient between its weight and the sum of the weights of the compatible vertices to which it is joined by an edge. Certainly, GH1 does not consider the future consequences of each decision. However, GH2 does, because it penalises the vertices for which the denominator (that is, the weights of the vertices that can no longer be part of the set once the chosen vertex has been incorporated) is a high value, which is a way of taking the future consequences of the decision into account.

ii) Probing the repercussion of one decision (to be taken in one iteration) in subsequent iterations.

For instance, the well-known nearest-neighbour heuristic for the TSP is myopic, but that which consists in evaluating the cost of taking k steps, in all the possible ways, from the last vertex reached and going to the first vertex of the best of these partial itineraries is also a greedy heuristic. Several authors have used the expression “greedy look-ahead” to refer to the greedy algorithms that take into account, to a greater or lesser extent, the consequences of a decision for decisions in subsequent iterations.

iii) Including in the indicator a bound (or, more generally, an evaluation) of the cost of the decisions to be taken to determine the solution.

Sometimes, there is a clear case for choosing the indicator on which the decision that is taken at each iteration is based; at other times, there are various indicators that may be worthy of consideration or that seem reasonable (e.g. in assembly line balancing problems: the duration of the task, the number of subsequent tasks, etc.). Nevertheless, there is a risk that a greedy heuristic will provide not only bad solutions, but even the worst (Bang-Jensen et al., 2004). When there are various possible indicators, infinites may be generated using a weighting of the elementary indicators, although the choice of weights is not trivial (see, for instance, Altinel & Öncan, 2005). Indeed, these weights are sometimes assigned implicitly and without full justification (e.g. in the permutation flow-shop problem, when the process times of two fictitious machines based on the process times of real machines are calculated in order to reduce a problem with $m > 2$ machines to a problem of 2 machines, as in the algorithms proposed in Companys, 1966, Campbell et al., 1970 and Dannenbring, 1977).

Generally, therefore, an indicator for a greedy heuristic may be a function of the data and of several parameters. Thus, for many combinatorial optimisation problems, there are potentially an infinite number of greedy heuristics. The sum (or any other function) of the values of the objective function corresponding to the solutions obtained by applying each one of these heuristics to a given set of instances depends on the specific heuristic applied and, hence, is a function of the aforementioned parameters.

With the help of an example, we will go on to formalise these ideas and define them in detail.

Let us assume that we face a combinatorial optimisation problem P (such as the one-dimensional Knapsack Problem or KP):

$$\begin{aligned} & max\, f(X) \\ & s.t. \\ & X \in E \end{aligned}$$

where E is the finite set of feasible solutions.

In a greedy algorithm, a decision is chosen at each iteration k from those belonging to a set of feasible decisions, D_k (in the case of the KP, the decision may consist in determining which object should be placed in the knapsack of those that have not yet been added or rejected and for which there is still room in the knapsack). The decision, i_k^*, is such that $i_k^* = \arg\max_{i \in D_k} h\left(d_i^k\right)$, where h is the heuristic function (although different heuristic functions might be applied at each of the iterations, this possibility is not considered here) and d_i^k are the attributes that correspond to decision i at iteration k (in the KP, the attributes corresponding to object i are its weight, w_i, its value, v_i, and the heuristic function, e.g. $h(w_i,v_i)=\frac{v_i}{w_i}$). In the example of the KP, the values of the attributes are independent of k; however, as stated above, in adaptive algorithms the information on which the decision is based takes previous decisions into account.

The heuristic function can be made to depend on a set of n parameters, Π. Then, the decision corresponding to iteration k is that which fulfils

$$i_k^* = \underset{i\in D_k}{\arg\max}\, h\left(d_i^k,\Pi\right)$$

(such as $h(w_i,v_i,\alpha,\alpha',\beta,\beta',\gamma,\gamma')=w_i^{\alpha}\cdot v_i^{\alpha'}+\beta\cdot w_i^{\gamma}+\beta'\cdot v_i^{\gamma'}$ in the KP). In this way, an infinite set of heuristics, H, is defined for problem P.

3. Empirically Adjusted Greedy Algorithm (EAGH)

Choosing the best heuristic from those belonging to set H, for a given set of instances I, may be approached as the optimisation of a function φ of the parameters of $\wp$. For all the instances of I, the φ function may be the sum of the values of objective function f that correspond to the solutions that are obtained when the heuristic h defined by the values of the parameters is applied. There are also other possibilities for defining φ, such as weighting the values of function f with weights associated with the instances.

Let us consider a set of instances, I, of problem P and one of its subsets, $T\subset I$ (where $|T|=N$), which we will call the training set. Let $X_{i,\bar{\Pi}}$ be the solution of the instance $i\in T$ when the greedy algorithm defined by the heuristic function $h\left(d_i^k,\bar{\Pi}\right)$ is applied. The following function can then be defined:

$$\varphi(\Pi)=\sum_{i\in T} f\left(X_{i,\Pi}\right)$$

In general, this function is not expected to have any special, recognisable properties. Let us assume only that, by applying the heuristic to the training set of instances, T, we can calculate the value of the function for each of the parameters' set of values. Consequently, to look for good values for the parameters (i.e., values that provide good values for the φ function) only a direct algorithm may be used (that is, an algorithm that only uses the values of the function), such as the one devised by Nelder and Mead (N&M), also named the flexible polyhedron algorithm (Nelder and Mead, 1965, Box et al., 1969 and Corominas et al., 1997).

The idea of optimising a function of several parameters to determine the best element in a set of heuristics is the inspiration for the CALIBRA program (Adenso-Díaz and Laguna, 2005), which is used to determine the values for the parameters of a metaheuristic. In the case of the metaheuristic, however, the parameters are explicit and it is a well-known fact that finding appropriate values for these parameters is a difficult task. In the case of greedy heuristics, the parameters are implicit; indeed, the parameters are the result of a certain way of “seeing” the greedy heuristic, that is, as an element in an infinite set (of course, a given heuristic may be part of a multitude of sets).

Schematically, the procedure EAGH consists in the following:

(i) Generating a set of training instances, T .

(ii) Determining the best values for the parameters using a direct algorithm (optimising $\varphi(\Pi)$).

(iii) Validating the result by applying the heuristic to a new set of instances from the same population, $I \setminus T$.

The algorithm chosen for step (ii) is Nelder and Mead's. Other algorithms might be used, but N&M has been widely applied, with good results, since it was published, and recent literature (Anjos et al., 2004 and Chelouah and Siarry, 2005) indicates that it is still one of the instruments used to solve optimisation problems with non-differentiable objective functions.

The N&M algorithm is based on $n+1$ points, in the n-dimensional space of the parameters, that must form a simplex (also called hypertetrahedron), which is preferably regular; at each iteration, one or more points are generated in this space and the value of the function (φ) at each of these points is calculated. In EAGH, this calculation involves applying the heuristic defined by the coordinates of the point (i.e., the values of the parameters belonging to set Π) to all the instances of the training set. The algorithm leads to a point that is probably close to a local optimum; given that the properties of the function are not known, one may generally assume that it is multimodal, so an approximation to a global optimum cannot be guaranteed. It is known that this difficulty is common to all global optimisation problems.

One or two vertices of the initial simplex in the N&M algorithm may correspond to the greedy heuristics that are already known, from among those belonging to set H, which turn out to be the most appropriate for the training set (to make three or more points correspond to known heuristics could lead to an inappropriate initial simplex, since, as stated above, it should ideally be regular). In this way, one ensures that the quality of the result provided by the EAGH, for the training set T , is not lower than that provided by the best heuristic of those that are known, because in the N&M algorithm the last solution obtained cannot be worse than the best of the initial ones.

This approach has been applied to a sequencing problem, with very good results that were better than those obtained with greedy heuristics based on elementary indicators (Corominas et al., 2005).

The EAGH procedure can of course be applied to a single instance (this is the specific case $N=1$, which we will refer to as EAGH-1, distinguishing it from the case $N>1$ which we will call, from now on, EAGH-N). Obviously, in EAGH-1, the distinction between functions f and φ ceases to be relevant. EAGH-1 must be classed as a specific heuristic algorithm (in fact, a local optimisation procedure in the space of heuristics, H) that cannot be worse that that corresponding to the best vertex of the initial simplex in the N&M algorithm.

So, given N instances of a problem, that constitute the training set T, one can apply EAGH-N to determine a set of values for the parameters and then use them to define a heuristic that can be applied to new instances of I. The alternative of applying EAGH-1 to each instance may provide better solutions overall (although this is a conjecture to be explored), but excludes the possibility of extrapolating to new instances.

4. Future research

Future research will involve applying EAGH to various combinatorial optimisation problems and determining the most appropriate functions *h* and the best values for the parameters for instances drawn from specific populations.

Acknowledgment

To Dr. Rafael Pastor for his careful reading of the first drafts of this paper and for his useful comments and suggestions.

References

Adenso-Díaz, B. and Laguna, M. (2005) Fine-tuning of algorithms using fractional experimental designs and local search. Pending publication in *Operations Research*.

Altinel, I. K., Öncan, T. (2005) A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. *Journal of the Operational Research Society* 56, 954-961.

Anjos, M.F., Cheng, R.C.H. and Currie, C.S.M. (2004) Maximizing revenue in the airline industry under one-way pricing. *Journal of the Operational Research Society* 55, 535-541.

Bang-Jensen, J., Gutin, G. and Yeo, A. (2004) When the greedy algorithm fails. *Discrete optimization* 1, 121-127.

Barr, R., Golden, B., Kelly, J., Resende, M. and Stewart Jr, W. (1995) Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics* 1, 9-32.

Bixby, R.E. (2002). Solving real-world linear programs: a decade and more of progress. *Operations Research* 50 (1) 3-15.

Box, M.J., Davies, D. and Swann, W.H. (1969) Non-linear optimization techniques. Oliver & Boyd.

Campbell, H. G., Dudek, R. A., Smith, M. L. (1970) A heuristic algorithm for the n-job, m-machine sequencing problem. *Management Science*, 16, B630-637.

Chelouah, R. and Siarry, P. (2005) A hybrid method combining continuous tabu search and Nelder-Mead simplex algorithms for the global minimization of multiminima functions. *European Journal of Operational Research* 161, 636-654.

Clarke, G. and Wright, J. (1964) Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research, 12, 568-581.*

Companys, R. (1966) Métodos heurísticos en la resolución del problema del taller mecánico, Estudios empresariales, vol. 5, nº 66/2, 3-14.

Corominas, A., Companys, R., Coves, A.-M., Ferrer, J. and Roselló, X. (1997) *Mètodes quantitatius d'organització industrial: problemes no lineals.* Edicions UPC.

Corominas, A., Pastor, R. and Sánchez, A. (2005) Heurística, resultado de calibración de heurísticas, para la determinación de secuencias en una máquina multiproducto sujeta a fallos y con costes cuadráticos. CIO2005, Ninth Conference on Organizational Engineering, Gijón.

Dannenbring, D. G. (1977) An evaluation of flow-shop sequencing heuristics. *Management Science*, 23, 1174-1182.

Feo, T.A. and Resende, M.G.C. (1995) Greedy randomized adaptive search procedures. *Journal of Global Optimization* 6, 109-133.

Gass, S.I. and Harris, C.M., eds. *Encyclopaedia of Operations Research and Management Science*. Kluwer, 1996.

Hooker, J.N. (1995) Testing heuristics: we have it all wrong. *Journal of Heuristics* 1, 33-42.

Nelder, J.A. and Mead, R. (1965) A simplex method for function minimization. *The Computer Journal* 7, 308-313.

Rardin, R.L. and Uzsoy, R. (2001) Experimental evaluation of heuristic optimization algorithms: a tutorial. *Journal of Heuristics* 7, 261-304.

Silver, E.A. (2004). An overview of heuristic solution methods. *Journal of the Operational Research Society* 55 (9), 936-956.

Watson, J.P., Barbulescu, L., Howe, A.E. and Whitley, L.D. (1999) Algorithm performance and problem structure for flow-shop scheduling. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), Orlando, FL, USA.*