European Journal of Operational Research 226 (2013) 1–8

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Invited Review

Multi-start methods for combinatorial optimization [☆]

Rafael Martí [a,*], Mauricio G.C. Resende [b], Celso C. Ribeiro [c]

[a] Departamento de Estadística e Investigación Operativa, Facultad de Matematicas, Universidad de Valencia, Dr. Moliner 50, 46100 Burjassot, Valencia, Spain
[b] Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932, USA
[c] Department of Computer Science, Universidade Federal Fluminense, Rua Passo da Pátria, 156, Niterói, RJ 24210-240, Brazil

A R T I C L E   I N F O

Article history:
Received 12 March 2012
Accepted 10 October 2012
Available online 1 November 2012

Keywords:
Metaheuristics
Multi-start methods
Adaptive memory programming
GRASP

A B S T R A C T

Multi-start methods strategically sample the solution space of an optimization problem. The most successful of these methods have two phases that are alternated for a certain number of global iterations. The first phase generates a solution and the second seeks to improve the outcome. Each global iteration produces a solution that is typically a local optimum, and the best overall solution is the output of the algorithm. The interaction between the two phases creates a balance between search diversification (structural variation) and search intensification (improvement), to yield an effective means for generating high-quality solutions. This survey briefly sketches historical developments that have motivated the field, and then focuses on modern contributions that define the current state-of-the-art. We consider two categories of multi-start methods: memory-based and memoryless procedures. The former are based on identifying and recording specific types of information (attributes) to exploit in future constructions. The latter are based on order statistics of sampling and generate unconnected solutions. An interplay between the features of these two categories provides an inviting area for future exploration.

1. Introduction

The methods that provide the origins of what we now call *multi-start* procedures consist primarily of repeated applications of constructive methods. The best solution produced by these repeated applications is then normally selected for implementation. Early proposals can be found in the domains of heuristic scheduling (Muth and Thompson, 1963; Crowston et al., 1963), the traveling salesman problem (Held and Karp, 1970; Lawler et al., 1985), and knapsack problems with single and multiple constraints (Senju and Toyoda, 1968; Wyman, 1973; Kochenberger et al., 1974). It would be possible to go back even farther in time and identify various methods used in statistics and calculus as instances of utilizing repeated constructions to produce a preferred candidate, although such methods were not used to address problems in the realm of optimization as we view it today.

More recently, Glover (1977) makes several connections to multi-start search by means of a framework in which multi-start search includes local search to improve the starting solutions. Within this framework, procedures are given for generating starting values for variables and for generating values perturbed from other starting points. By varying the rules for the perturbation, these strategies include customary local search approaches for producing re-starts. A series of extensions of this framework are given in Glover (1986, 1989, 2000), addressing controlled randomization, learning strategies, induced decomposition, and adaptive memory processes (as introduced in tabu search). Emphasis is placed on the interaction between intensification and diversification as a means for creating a more effective search process. Several parts of the discussion bear on the area of multi-start methods. Controlled randomization classically takes two forms. The first is the well-known *random restart* approach, which injects a randomizing element into the generation of an initial starting point to which a heuristic is subsequently applied. The second classical version of this approach is the *random shakeup* procedure which, instead of restarting, periodically generates a randomized series of moves that leads the heuristic from its customary path into a region it would not otherwise reach.

Early multi-start methods from the optimization setting can be interpreted as using a binary representation of decision variables, starting from a null solution and selecting variables to set to 1, thus identifying assignments of jobs to machines, or edges to tours, or items to compose a knapsack, and so forth. This construction process continued until obtaining a complete or maximally feasible construction, at which point all remaining variables were implicitly assigned values of 0. We adopt this perspective of assigning values to zero-one variables as a basis for describing constructive processes within multi-start methods in general, allowing for the added provision of considering associated *destructive* processes

[☆] AT&T Labs Research Technical Report.
[*] Corresponding author.
E-mail addresses: rafael.marti@uv.es (R. Martí), mgcr@research.att.com (M.G.C. Resende), celso@ic.uff.br (C.C. Ribeiro).

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2012.10.012

that instead operate by successively assigning values of 0 to selected variables (where these variables would normally be assigned values of 1 in constructive processes). Different coding schemes can be used to encompass a vast array of problems within this framework, even in cases where a binary formulation that casts a problem as a mathematical program would be inappropriate or counterproductive (e.g., due to the complexity of describing the problem objective or constraints within a zero-one formulation). Later in this section, we make use of one such coding scheme. Aggregation and disaggregation methods can also be expressed within this framework by defining zero-one variables within hierarchies, but we will not focus on such approaches here.

The first multi-start methods were typically based on implementing the re-starting step by randomly varying the choice of variables to receive a unit value, or at the other extreme by simply going through a pre-defined list of choice rules and applying a currently selected rule to build the current construction. In problem contexts where it was possible to modify a completed construction by moves that did not hopelessly destroy feasibility, the approach that is now commonly given the name of *local search* or *neighborhood search* was sometimes applied in conjunction with the constructive processes in an effort to improve the solutions generated. More recently, this marriage of constructive and local search procedures has become the customary way to apply multi-start methods, such as in the GRASP heuristics which we examine later in this survey. The motivation of enhancing constructed solutions is additionally joined by the motivation of using the varied re-constructions as a means of diversifying the solutions that launch the local search. In short, multi-start methods from the modern perspective embody a blend of intensification and diversification, and it is generally acknowledged that the nature of this blend is a primary determinant of the effectiveness of the overall method.

In this survey we will chiefly focus on the features that have been found to characterize some of the best multi-start methods, rather than attempting to work our way through the back alleys of all the various methods that have been proposed over time. Among the substantial range of ways for classifying multi-start methods, we elect to employ a classification that divides multi-start methods into two main groups, consisting of memory-based versus memoryless procedures. A useful outcome of this classification is that it permits us to conveniently differentiate certain innovations that have provided important advances, according to our focus on the multi-start methods that currently rank among the leading algorithms of this genre.

Multi-start procedures were originally conceived as a way to exploit a local or neighborhood search procedure, by simply applying it from multiple random initial solutions. Modern multi-start methods usually incorporate a powerful form of diversification in the generation of solutions to help overcome local optimality. Without this diversification, such methods can become confined to a small region of the solution space, making it difficult, if not impossible, to find a global optimum.

The explicit use of memory structures constitutes the core of a large number of intelligent solvers, including tabu search (Glover, 1989), scatter search (Laguna and Martí, 2003), and path-relinking (Ribeiro and Resende, 2012). These methods, generically referred to as *adaptive memory programming*, exploit a set of strategic memory designs. On the other hand, we can also find successful metaheuristics, such as simulated annealing (Kirkpatrick et al., 1983), noising methods (Charon and Hudry, 2002), and GRASP (Feo and Resende, 1995), with no memory structure in their original designs. To focus our attention on memory-based and memoryless multi-start methods, we target adaptive memory programming and GRASP heuristics.

The re-start mechanism of multi-start methods can be superimposed on many different search methods. Once a new solution has been generated, a variety of options can be used to improve it, ranging from a simple greedy routine to a complex metaheuristic-based heuristic. An open question in order to design a good search procedure is whether it is better to implement a simple improving method that allows a great number of global iterations or, alternatively, to apply a complex routine that significantly improves a few generated solutions. A simple procedure depends heavily on the initial solution but a more elaborate method takes much more running time and therefore can only be applied a few times, thus reducing the sampling of the solution space.

The remainder of this paper is organized as follows. In Section 2 we introduce notation for combinatorial optimization and provide pseudo-codes for solution construction procedures and multi-start algorithms. Adaptive memory programming methods, which focus on exploiting a set of strategic memory designs, are addressed in Section 3. Section 4 reviews greedy randomized and GRASP multi-start methods and makes a connection between these methods and the path-relinking strategy for search intensification. Concluding remarks are drawn in Section 5.

2. Combinatorial optimization

We consider in this survey a combinatorial optimization problem defined by a finite ground set $E = \{1, \ldots, n\}$, a set of feasible solutions $F \subseteq 2^E$, and an objective function $f : 2^E \rightarrow \mathbb{R}$. In its minimization version, we search an optimal solution $S^* \in F$ such that $f(S^*) \leqslant f(S)$, $\forall S \in F$. The ground set E, the cost function f, and the set of feasible solutions F are defined for each specific problem. For instance, in the case of the traveling salesman problem, the ground set E is that of all edges connecting the cities to be visited, F is formed by all edge subsets that determine a Hamiltonian cycle, and $f(S)$ is the sum of the costs of all edges in S. Another example is the maximum clique problem, where the ground set E is the set of all vertices of the graph, F is the set of all subsets of E for which all vertices are mutually adjacent, and $f(S)$ is the cardinality of the clique $S \in F$.

We consider next a simple algorithm to construct a feasible solution $S \in F \subseteq 2^E$ for a large class of combinatorial optimization problems. Infeasible solutions are those that contain certain subsets of 2^E and we restrict ourselves to problems for which we can easily identify infeasibilities by detecting the presence of any of these subsets in the solution under construction. This setup encompasses many combinatorial optimization problems, including problems such as set covering, maximum clique, quadratic assignment, and traveling salesman. For example, in the case of the traveling salesman problem, a subset of the edges corresponding to a subtour indicates infeasibility. For the maximum clique problem, any set of vertices whose elements are not mutually adjacent indicates infeasibility.

Algorithm 1. Pseudo-code for a solution construction procedure.

```
procedure ConstructSolution
  Initialize solution: $S \leftarrow \emptyset$;
  Initialize candidate set:
      $\mathcal{C} \leftarrow \{s \in E \setminus S \mid S \cup \{s\}$ is not infeasible$\}$;
  while $\mathcal{C} \neq \emptyset$ do
      Select $s \in \mathcal{C}$;
      Add $s$ to solution: $S \leftarrow S \cup \{s\}$;
      Update candidate set:
          $\mathcal{C} \leftarrow \{s \in E \setminus S \mid S \cup \{s\}$ is not infeasible$\}$;
  end
  return $S$;
```