European Journal of Operational Research 199 (2009) 620–629

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Hybridizing exact methods and metaheuristics: A taxonomy

L. Jourdan *, M. Basseur, E.-G. Talbi

LIFL/INRIA/CNRS, Bat M3, Cité Scientifique, 59655 Villeneuve d'Ascq, France

ARTICLE INFO

Article history:
Received 22 September 2006
Accepted 10 July 2007
Available online 13 April 2008

Keywords:
Taxonomy
Combinatorial optimisation
Metaheuristics
Exact methods

ABSTRACT

The interest about hybrid optimization methods has grown for the last few years. Indeed, more and more papers about cooperation between heuristics and exact techniques are published. In this paper, we propose to extend an existing taxonomy for hybrid methods involving heuristic approaches in order to consider cooperative schemes between exact methods and metaheuristics. First, we propose some natural approaches for the different schemes of cooperation encountered, and we analyse, for each model, some examples taken from the literature. Then we recall and complement the proposed grammar and provide an annotated bibliography.

1. Introduction

NP-hard problems are difficult to solve and no polynomial time algorithm are known for solving them. Unfortunately, most combinatorial optimization problems, such as the Travelling Salesman, N-Queens, Bin Packing, 0/1 Knapsack, Graph Partitioning, are NP-hard. Two approaches can be considered to solve this kind of problems depending on their size.

For small instances, researchers usually use exact methods. Exact methods find the optimal solution and assess its optimality. There exist numerous exact methods such as the family of Branch and X (Branch and Bound algorithm [58], Branch and Cut algorithm [42], Branch and Price algorithm [12]), Linear Programming, Dynamic Programing, etc. A branch and X algorithm uses a divide and conquer strategy to partition the solution space into subproblems and then optimizes individually each subproblem. Exact methods are known to be time expensive, so they can not be applied to large NP-hard problems or difficult ones.

When instances become too large for exact methods, heuristics and in particular metaheuristics are often used. There are two main categories of metaheuristics: single solution algorithms and population based algorithms. The first category gathers local search (LS) [54], greedy heuristic (GH) [70], simulated annealing (SA) [50], tabu search (TS) [40], Iterated Local Search (ILS) [56] etc. The second category, which is more and more studied, regroups evolutionary algorithms such as genetic algorithms [44], evolution strategies [74], genetic programming [52], and also ant colonies (AC) [31], scatter search (SS) [39], immune systems [48] etc. However, in general, metaheuristics are not able to solve the problems to optimality and some convergence problems can be encountered.

During the last years, many works have been realized on cooperative (or hybrid) optimization approaches. In many cases, best results are obtained with this kind of approaches, especially on real-life problems. At the beginning, cooperations were mainly realized between several metaheuristics. But nowadays, more and more cooperation schemes between metaheuristics and exact approaches are proposed. These strategies usually give good results because they are able to exploit simultaneously the advantages of both types of methods. For example, it may allow to give quality guarantees to the identified solutions.

In this article, we propose to survey the different cooperation between these two types of method. The fact that more and more papers deal with this kind of approaches (see Fig. 1) clearly indicates that it is an important issue for the operational research community. So it seems interesting to classify these works. A state of the art of this type of cooperation has been proposed recently by Stützle and Dumitrescu [34]. They distinguish five classes of approaches for cooperation between exact and local search methods; they also provide an example for each type. The five classes proposed are:

- Use exact algorithms to explore large neighborhoods in local search algorithms.
- Perform several runs of a local search and exploit information in high quality solutions to define smaller problems that are amenable for solution with exact algorithms.
- Exploit bounds in constructive heuristics.
- Use information from relaxations of integer programing problems to guide local search or constructive algorithms.
- Use exact algorithms for specific procedures in hybrid metaheuristics.

* Corresponding author.
E-mail addresses: laetitia.jourdan@inria.fr (L. Jourdan), basseur@lifl.fr (M. Basseur), talbi@lifl.fr (E.-G. Talbi).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2007.07.035

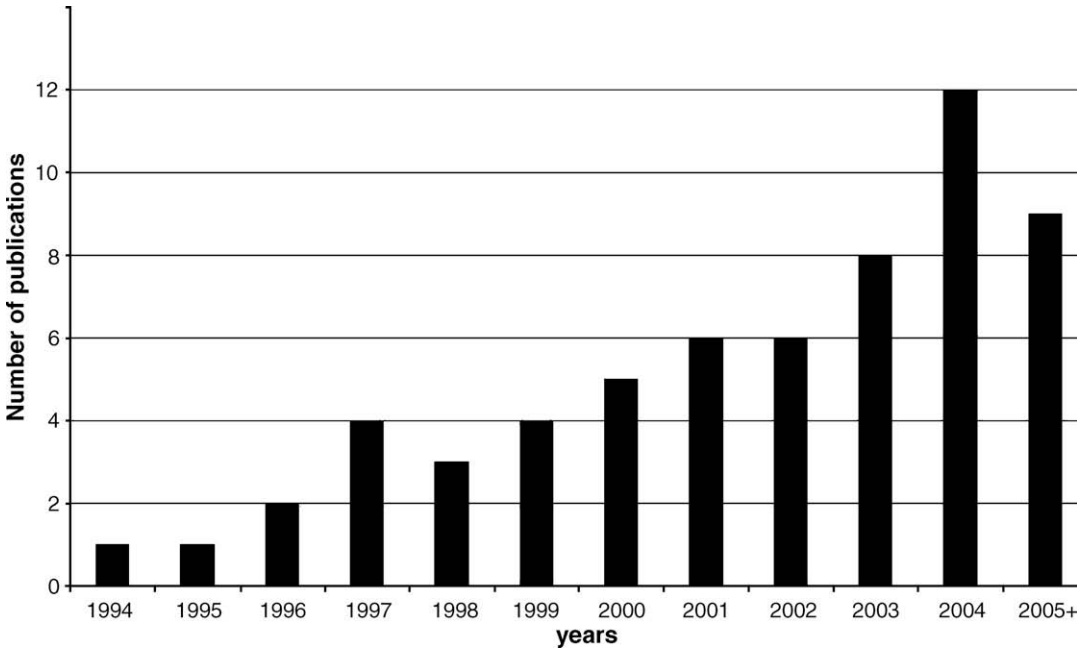

Fig. 1. The evolution of the publication activity on hybridizaton between exact methods and metaheuristics.

The survey proposed by Stützle and Dumitrescu presents several cooperative approaches to explain the different classes but would have more interest if it were generalized to every optimization methods. Their paper also excludes some combination such as preprocessing.

In [68], Puchinger and Raidl, propose a survey of the state-of-the-art approaches that combine exact methods and metaheuristics. Their survey provides a classification of methods thanks to different classes. The first one deals with collaborative combinations where no algorithm is contained in any other. This class is divided into subclasses:

- Sequential execution.
- Parallel and intertwined execution.

The second class regroups integrative combinations and is subdivided into two subclasses:

- Incorporating exact algorithms in metaheuristics.
- Incorporating metaheuristics in exact algorithms.

Puchinger and Raidl illustrate each subclass with examples issued from the literature.

In this article, we propose also to classify different articles issued from the literature but we will also propose a taxonomy of methods that combines exact and heuristic approaches. Important contributions of this article are the formal grammar proposed to classify the methods, integration of conceptual and hierarchical aspects. A separation between design and implementation is also taken into account. Our survey is far from providing an exhaustive list but it will constitute a good way for authors of new cooperation papers to classify their approach or for developers to find ideas on how to combine methods efficiently. In this article, cooperation and hybridization will be used in the same way. These terms will indicate algorithms which combine different optimization methods. The present paper uses the taxonomy proposed by Talbi [80] as we observe that it is valuable for cooperative methods between metaheuristics and exact approaches.

The remainder of the article is organized as follows. In Sections 2–4 the taxonomy used in [80] is recalled and illustrated with examples of cooperation between exact methods and metaheuristics, as in [80] the author only considers cooperation between metaheuristics. The taxonomy is divided into three general aspects: cooperation method design (Section 2), approach design (Section 3), and implementation issues (Section 4). In Section 5, the grammar for hybrid metaheuristics is extended, and an annotated review of different references is presented according to the taxonomy. Conclusions are drawn in Section 6.

2. Cooperation method design

Cooperation involves two main components: the design and the implementation. The former category concerns the cooperative algorithm itself, involving issues such as the functionality and the architecture. The implementation takes into account the hardware platform, programming model and the environment.

In this section, we will focus on the design of the cooperative mecanisms, i.e. how the methods will cooperate. For each type of classification, the derived classes are presented and some examples from the literature are described.

To facilitate the reading of our survey, the terms used in [80] are recalled. The design of metaheuristics can be classified in two types of design classification:

- Low-level/High-level
  - *Low-level*: The functional composition of a single optimization method. A given function of a metaheuristic is replaced by another method.
  - *High-level*: The different algorithms are self-contained.

- Relay/Teamwork
  - *Relay*: A set of methods is applied one after another, each using the output of the previous as its inputs, acting in a pipeline fashion.
  - Teamwork represents cooperative optimization models.

Four classes are derived from this hierarchical taxonomy (see Fig. 2).