Using oracles for the design of efficient approximation algorithms
Marin Bougeret, Pierre-Francois Dutot, Denis Trystram

To cite this version:
Marin Bougeret, Pierre-Francois Dutot, Denis Trystram. Using oracles for the design of efficient approximation algorithms. MAPSP: Models and Algorithms for Planning and Scheduling Problems, Jun 2011, Nymburk, Czech Republic. hal-00738513

HAL Id: hal-00738513
https://hal.science/hal-00738513
Submitted on 14 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Using oracles for the design of efficient approximation algorithms

Marin Bougeret (Speaker) * Pierre-Francois Dutot † Denis Trystram ‡

We are interested here in oracle techniques for the design of approximation algorithms. Following the classical definition, an oracle is a black box capable of answering correctly and instantaneously any question. Several classical PTAS design techniques can be expressed using oracle formalism (by allowing the algorithm to “guess” some values during the computation).

Our objective in this work is to point out the interest of oracle techniques, beyond the design of PTAS. Indeed, questions to the oracle (i.e. guessed values) leading to non-polynomial algorithms must also be considered, as the complexity may be exponential, but in a parameter that is supposed to be “small”. Moreover, we aim at showing how it is possible to “degenerate” questions asked to the oracle to derive fast implementations of these interactive algorithms. These ideas will be illustrated on the classical makespan minimization on uniform machines problem (Q||Cmax).

Context: oracle algorithms

Given an instance I of an optimization problem, an oracle algorithm Aor asks the oracle for a guess, in the form of a string r∗ ∈ R, that generally provides some information on the structure of an optimal solution. Then, the algorithm constructs a solution Aor(I, r∗) for the initial problem. From such an algorithm, it is possible to derive a “classical” algorithm A (without oracle), by either re-executing Aor(I, r) for any r ∈ R, or constructing separately r∗ (using another algorithm). Taking the example of scheduling problems, a very classical question r∗ is an “optimal configuration” of a well-chosen small subset of k tasks (among the n of the instance), where k is constant. Such information may allow arbitrarily good approximation ratios (like 1 + 1/k) at the price of subset enumeration, when simulating the oracle.

Hence, it is clear that there exist deep connections between oracle algorithms and techniques for designing approximation schemes. As shown in [1], an oracle formulation allows natural alternative definitions of several classical techniques (as those presented in [9]). Most of such techniques are based on information obtained by exhaustive enumeration or by binary search. Replacing them by oracle answers separates difficulties due to the information determination from the ones due to its utilization.

*marin.bougeret@ens-lyon.fr. Ecole Normale Superieure de Lyon, France
†pierre-francois.dutot@imag.fr. Grenoble University, France
‡trystram@imag.fr. Grenoble University and Institut Universitaire de France
Application on the classical $Q\|C_{\text{max}}$ problem

Let us consider the problem of minimizing the makespan when scheduling independent tasks on uniform machines as a case study. It is shortly denoted by $Q\|C_{\text{max}}$.

Several approximation algorithms have been proposed for this problem. The 2 ratio (achieved by the classical Longest Processing Time algorithm [4]) has been improved to $\frac{3}{2}$ in [5] (using the dual approximation technique), and to 1.382-approximation in [2]. Among all existing approximation schemes, the most relevant here are the following (we list below the time complexity to achieve a ratio of $(1 + \epsilon)$):

- $O(mn^{1.9} + 3)$ in [5]
- $O((\frac{1}{\epsilon}n^2)m^{m-1})$ (also applies to $R\|C_{\text{max}}$) in [6]
- $O((n + 1)^{\frac{m}{\epsilon}}\text{poly}(n,m))$ (also applies to $R\|C_{\text{max}}$) in [8]
- $O(n) + \left(\frac{\log(m)}{\epsilon}\right)O(m^2)$ (also applies to $R|c_{ij}|C_{\text{max}}$) in [3]
- $O(2^{O(1/\epsilon^2\log(1/\epsilon)^3)}\text{poly}(n,m))$ in [7]

We propose an oracle algorithm based on [5]. For any $a \in \mathbb{N}^*$, our algorithm guarantees an $1 + \frac{1}{a}$ ratio by asking some information on the “big” tasks scheduled on each machine (i.e. whose computation requires more than a fraction $\frac{1}{a}$ of the total computation time on this machine).

Firstly, notice that the classical guess (i.e. asking the index of the big tasks scheduled on each machine) would lead to an approximation scheme with the same complexity as the one in [8]. Thus, we show how to reduce the amount of information asked, and thus the size of $R_I$, for small values of $a$ (typically $a = 3$ or 4). We get for instance a $\frac{4}{3}$ (resp. $\frac{5}{4}$)-approximation by only asking the number of big tasks for each machine, leading to an algorithm in $O(2^m\text{poly}(n,m))$ (resp.$O(3^m\text{poly}(n,m))$). Thus, these approximation algorithms can be faster than the better approximation schemes applied for $\epsilon$ equal to $\frac{1}{3}$ (resp. $\frac{1}{4}$), as there is no constant hidden in the exponent.

Secondly, we discuss efficient implementations where the algorithms avoid asking some sub-parts of the question. The key idea is to check if some additional a priori unexpected conditions become true during the execution on the particular current instance, allowing then to make a local optimal decision (without oracle query).

The approach presented in this work leads to the following natural questions for $Q\|C_{\text{max}}$.

- Using only one bit of information for each machine, what information should be asked to obtain a ratio better than $\frac{4}{3}$?
- How to reduce the amount of information used for larger values of $a$?
References


