The guess approximation technique and its application to the Discrete Resource Sharing Scheduling Problem.

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

1 Introduction

We present in this work a technique for designing approximation schemes and its application to the discrete resource sharing problem ($dRSSP$). We use the (classical) oracle formalism, in which a reliable oracle provides some information (usually called "a guess") that helps to construct a good approximated solution. This oracle technique is sometimes called "partial enumeration" [ST07] or "output structuring" [SW00]. Generally, this oracle design technique consists of two steps: construct an algorithm that, given the oracle guess, provides the approximated solution, and then simply enumerates the set of all the possible guesses. The bigger the guess length (number of bits), the better the approximated solution. However the set of all possible guesses is also larger. The goal is to find an "efficient" guess, that is a small guess containing a lot of information. Thus, we are interested in finding a compact way to encode the guess. This (natural) idea has already been addressed [HS89] by mixing two techniques (structuring the input, and apply partial enumeration on this simplified input). We propose in this paper a slightly different solution by approximating the guess itself through a contraction function. We believe that this refinement may lead to non trivial guesses, and enable to use guesses which were too long before the contraction. Moreover, approximation schemes derived from this technique have a complexity depending on two parameters (the complexity indeed depends on the length of the guess, and the roughness of the guess approximation), which allows a more flexible tradeoff between quality of the approximation and computational complexity.

In the second section, we present the targeted problem ($dRSSP$) and some associated results (NP completeness, approximations algorithms and a worst case); in section three, we define the guess approximation technique and apply it to the $dRSSP$.

2 Targeted problem: $dRSSP$

The discrete Resource Sharing Scheduling Problem ($dRSSP$) consists in allocating a set of discrete resources to heuristics to solve given instances in minimum time. The idea in this problem is to find an efficient partition of resources to deploy the set of heuristics on the resources. This problem can formally be described as follows:

*This PhD is supported by DGA-CNRS.

†FirstName.LastName@imag.fr. Laboratoire d'Informatique de Grenoble, Team MOAIS, 51 av. Kuntzmann, 38330 Montbonnot Saint-Martin, France.

Definition 1 (discrete Resource Sharing Scheduling Problem ($dRSSP$))
Input: *A finite set of instances $I = \{I_1, \ldots, I_n\}$, a finite set of heuristics $H = \{h_1, \ldots, h_k\}$, a set of m identical resources, a cost $C(h_i, I_j, p) \in R^+$ for each $I_j \in I$, $h_i \in H$ and $p \in \{1, \ldots, m\}$*
Output: *Find a partition of resources $S = (S_1, \ldots, S_k), S_i \in \{0, \ldots, m\}, 0 < \sum_{i=1}^{k} S_i \leq m$ which minimizes $\sum_{j=1}^{n} \min_{1 \leq i \leq k} \{C(h_i, I_j, S_i) | S_i > 0\}$*

The individual instance cost ($\min_{1 \leq i \leq k} \{C(h_i, I_j, S_i)\}$) introduced by Sayag et al. [SFM06] and used here, stems from the fact that for each instance, all the different heuristics are executed with the defined share and then stop their execution when at least one heuristic finds a solution. We focus on the linear version, where the execution cost is anti-proportional to the number of resources used ($C(h_i, I_j, p) = \frac{C(h_i, I_j, m)m}{p}$). This problem has been studied in the continuous case in [SFM06]. It has been shown in [BDG$^+$09] that the discrete version has no polynomial approximation algorithm within a constant factor unless $P = NP$. Thus we tackle a restriction of this problem in which each heuristic must use at least one processor ($S_i \geq 1$). In [BDG$^+$09], several approximation schemes were provided for this restriction including a $\frac{k}{g+1}$ approximation in $O((km)^g kn)$, for any $g \in \{1, \ldots, k-1\}$. We complete this work by providing a tight worst case for this algorithm, and the proof of the NP completeness of this restriction of $dRSSP$, which was still opened in [BDG$^+$09]. In the next section, we introduce the guess approximation technique and we apply it to our problem, improving drastically the previous approximation schemes.

3 The guess approximation technique

We start by introducing the appropriate oracle formalism. The two steps process (provide a solution given a fixed guess, and enumeration) described previously can by formalized as follows. A ρ oracle approximation is an algorithm A that given an instance I, a question $q(I)$, and a oracle response $r^* \in R$ (such that a particular property $P(q(I), r^*)$ is true), delivers a feasible solution $S(r^*) \leq \rho\ S_{opt}$, where S_{opt} is the optimal value for the instance I. Then, the set R is enumerated to find such an r^*. The standard notion of guess corresponds here to the couple $(q(I), r^*)$. This formalism is close to the "outline scheme" introduced in [HS89], but it seems more general to use an arbitrary property P rather than a labelling function. The guess approximation technique consists in finding a contraction function $f : R \rightarrow f(R)$ and an algorithm A' delivering $S(f(r^*)) \leq \rho'\ S_{opt}$, knowing that $P(q(I), r^*)$ is true. Thus, it is sufficient to only enumerate the set $f(R)$, which should be smaller than R.

We now focus on an application of the guess approximation technique to $dRSSP$. Recall that a solution S for this problem is described by a k dimensional vector $S = (S_1, \ldots, S_k)$. The $\frac{k}{g+1}$ oracle approximation in [BDG$^+$09] is achieved using an algorithm denoted by A_0, an empty question $q(I)$, and a response $r^* = [(i_1^*, \ldots, i_g^*), (r_1^*, \ldots, r_g^*)]$ satisfying the following property $P(q(I), r)$: $\exists S^*$ such that $(i_1, \ldots, i_g)$ are the index of the most used heuristics, and $(S_{i_1}^*, \ldots, S_{i_g}^*) = (r_1, \ldots, r_g)$. We mean by "the most used heuristics" that $T(h_{i_1}) \geq .. \geq T(h_{i_g}) \geq T(h_i), \forall i \notin \{i_1, \ldots, i_g\}$, where $T(h_i)$ is the sum of the execution time of all the instances j solved by heuristic i. The length of the response is $|r^*| = g(log(k) + log(m))$.

We now look for an appropriate contraction function f. The oracle provides two types of information: index of heuristics (which verify particular properties) and number of processors.

The first type of information seems to be hard to approximate. Indeed, the information here is intrinsically binary, that is a given heuristic satisfies or not a given property. Thus, we are more interested in approximating the second part of the oracle information: the number of processors allocated to particular heuristics (in an optimal solution). Given any response $r = [(i_1, \ldots, i_g), (r_1, \ldots, r_g)] \in R$, we consider a mantissa-exponent representation for the r_i, with a fixed size $j_1 \in \{1, .., \lceil \log(m) \rceil\}$ of mantissa. Thus, we write $r_i = a_i 2^{e_i} + b_i$, where a_i is encoded on j_1 bits, $0 \leq e_i \leq \lceil \log(m) \rceil - j_1$, and $b_i \leq 2^{e_i} - 1$. Then, we define $f(r) = [(i_1, \ldots, i_g), (a_1, e_1), \ldots, (a_g, e_g)]$. Notice that $|f(r)| = \Sigma_{j=1}^{g}(|i_j| + |a_j| + |e_j|) \leq g(log(k) + j_1 + log(log(m))$. Using this contraction function and the same algorithm A_0 it is straightforward to derive a $\beta + \frac{k-g}{g+1}(1 - \frac{1}{2^{j_1-1}})$ approximation (with $\beta = 1 + \frac{1}{2^{j_1-1}}$) in $O((k2^{j_1} log(m))^g kn)$, for any $g \in \{1, \ldots, k-1\}$ and $j_1 \in \{1, \ldots, log(m)\}$.

The guess approximation technique enabled to make usable a guess that contained a lot of information. Moreover, the obtained approximation schemes can be adjusted according to two parameters g and j_1. The technique proposed in this work was applied to a specific scheduling problem ($dRSSP$), improving previous existing approximation schemes. Remark that these approximation schemes are even $PTAS$ when k (the number of heuristics) is fixed. We intend to use this technique for other problems, including multi-organization scheduling problem.

References

[BDG+09] M. Bougeret, P.F. Dutot, A. Goldman, Y. Ngoko, and D. Trystram. Combining multiple heuristics on discrete resources. In *11th Workshop on Advances in Parallel and Distributed Computational Models APDCM, (IPDPS)*, 2009.

[HS89] L. A. Hall and D. B. Shmoys. Approximation schemes for constrained scheduling problems. In *SFCS '89: Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 134–139, Washington, DC, USA, 1989.

[SFM06] T. Sayag, S. Fine, and Y. Mansour. Combining multiple heuristics. In Durand and Thomas, editors, *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science*, volume 3884 of *LNCS*, pages 242–253. Springer, 2006.

[ST07] H. Shachnai and T. Tamir. Polynomial time approximation schemes - a survey. *Handbook of Approximation Algorithms and Metaheuristics, Chapman & Hall*, 2007.

[SW00] P. Schuurman and G.J. Woeginger. Approximation schemes - a tutorial. In *Lectures on Scheduling*, 2000.