The Lost Continent of Polynomial Time: Preprocessing and Kernelization

Michael R. Fellows

School of Electrical Engineering and Computer Science
University of Newcastle, University Drive, Callaghan NSW 2308, Australia
mfellows@cs.newcastle.edu.au

Abstract. One of the main objectives of the talk is to survey the history of the practical algorithmic strategy of *preprocessing* (also called *data-reduction* and *kernelization*) since the beginnings of computer science, and to overview what theoretical computer science has been able to say about it.

Parameterized complexity affords the subject of preprocessing (kernelization) a central place *via* the (trivial) lemma that states:

Lemma. *A parameterized problem Π is* fixed-parameter tractable *if and only if there is a transformation τ from Π to itself, that takes an instance (x, k) to an instance (x', k') where:*

(1) (x', k') is a yes-instance if and only if (x, k) is a yes-instance,
(2) $|x'| \leq g(k)$ for some function g associated to τ,
(3) $k' \leq k$,
(4) τ runs in polynomial time, that is, in time polynomial in $|(x, k)|$.

In the situation described by the lemma, we say that Π can be *kernelized* to instances of size $g(k)$. The lemma is built around a transformation τ that is many:1. This can be generalized to a notion of P-time Turing kernelization. If the parameterized problem Π is solvable in time $O(f(k)n^c)$, then the lemma provides only a P-time kernelization bound of $g(k) = f(k)$. Hence, membership in FPT generally insures only an exponential kernelization.

Many parameterized problems admit P-time kernelization bounds $g(k)$ where g is a polynomial, or even linear function of k. Sometimes, the bounds are stated in terms of other instance measures than total size, for example, a Vertex Cover instance (G, k) can be kernelized to an instance (G', k') where G' has at most $2k$ vertices. Another avenue for generalization is therefore to consider kernelization as a P-time transformation that bounds one parameter in terms of another (the overall input size, the number of vertices or edges, the treewidth, etc.).

Pre-processing is a humble strategy for coping with hard problems, almost universally employed. It has become clear, however, that far from being trivial and uninteresting, that pre-processing has unexpected practical power for real-world input distributions, and is mathematically a much deeper subject than has generally been understood. It is almost impossible to talk about pre-processing

H.L. Bodlaender and M.A. Langston (Eds.): IWPEC 2006, LNCS 4169, pp. 276–277, 2006.

in the classical complexity framework in any sensible and interesting way, and the historical relative neglect of this vital subject by theoretical computer science may be related to this fact.

Here is the difficulty. If my problem Π is NP-hard, then probably there is no P-time algorithm to *solve* the problem, that is, to completely dispose of the input. If you suggested that perhaps I should settle for a P-time algorithm that instead of completely disposing of the input, at least simplifies it by getting rid of, or reducing away, the easy parts — then this would seem a highly compelling suggestion. But how can this be formalized? The obvious first shot is to ask for a P-time algorithm that reduces the input I to an input I' where $|I'| < |I|$ in a way that loses no essential information (i.e., trades the original input for smaller input, which can be called *data reduction*). The difficulty with this "obvious" formalization of the compelling suggestion is that if you had such a P-time data reduction algorithm, then by repeatedly applying it, you could dispose of the entire input in polynomial time, and this is impossible, since Π is NP-hard. Thus, in the classical framework, an effort to formulate a mathematically interesting program to explore polynomial-time preprocessing immediately crashes.

In the parameterized complexity framework, however, such a program can be formulated in an absolutely interesting and productive way. The effectiveness of P-time preprocessing is measured against the structure represented by the parameter. You might reasonably call the subject of FPT kernelization the *Lost Continent of Polynomial Time* (a *lost continent* being something that is large and interesting, that "should have been" explored long ago, and that was somehow overlooked).

In the last few years, a number of researchers have made important, pioneering investigations of subclasses of FPT that have stronger definitional claims on capturing *practical* fixed-parameter tractability. The effectiveness of kernelization offers another approach to such exploration of the internal structure of FPT, for example, the subclasses

$$lin(k) \subseteq poly(k) \subseteq FPT$$

of parameterized problems that admit problem kernels of size linear (or polynomial) in k. The talk will survey some of the important open problems that attend this perspective on the structure of FPT.

The challenges of finding effective P-time kernelization algorithms for parameterized problems in FPT seems to offer a rich combinatorial landscape for novel strategies with strong payoffs for practical computing. Beyond such concrete challenges, there also seem to be opportunities for developing systematic methodologies. Essentially, the issue seems to be the development of *P-time extremal structure theory* relating a source parameter to a target parameter (which might be, as in the lemma, the overall instance size), modulo polynomial-time processing.

The talk will survey some of the intriguing concrete open problems and new approaches in this area.