A Survey of FPT Algorithm Design Techniques with an Emphasis on Recent Advances and Connections to Practical Computing

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. The talk will survey the rich toolkit of FPT algorithm design methodologies that has developed over the last 20 years, including "older" techniques such as well-quasi-ordering, bounded treewidth, color-coding, reduction to a problem kernel, and bounded search trees — which have continued to deepen and advance — as well as some recently developed approaches such as win/win's, greedy localization, iterative compression, and crown decompositions.

Only in the last few years has it become clear that there are two distinct theoretical "games" being played in the effort to devise better and better FPT algorithms for various problems, that is, algorithms with a running time of $O(f(k)n^c)$, where c is a fixed constant, k is the parameter, and n is the total input size. The first of these games is the obvious one of improving the parameter cost function $f(k)$. For example, the first FPT algorithm for Vertex Cover had a running time of $2^k n$. After a series of efforts, the best currently known algorithm (due to Chandran and Grandoni, in a paper to be presented at IWPEC 2004), has a parameter cost function of $f(k) = (1.2745)^k k^4$. The second theoretical game being played in FPT algorithm design has to do with efficient kernelization. The naturality and universality of this game comes from the observation that a parameterized problem is FPT if and only if there is a *polynomial time* preprocessing (also called *data reduction* or *kernelization*) algorithm that transforms the input (x, k) into (x', k'), where $k' \leq k$ and $|x'| \leq g(k)$ for some kernel-bounding function $g(k)$, and where of course (x, k) is a yes-instance if and only if (x', k') is a yes-instance. In other words, modulo polynomial-time pre-processing, all of the difficulty, and even the size of the input that needs to be considered, is bounded by a function of the parameter. The natural theoretical game from this point of view about FPT is to improve the kernel-bounding function $g(k)$. For example, after strenous efforts, a kernel-bounding function of $g(k) = 335k$ has recently been achieved (by Alber et al.) for the Planar Dominating Set problem.

Some of the new techniques to be surveyed pertain to the $f(k)$ game, some address the kernelization game, and some are relevant to both of these goals in improving FPT algorithms. One of the striking things about the "positive" toolkit of FPT algorithm design is how unsettled the area seems to be, with

S. Albers and T. Radzik (Eds.): ESA 2004, LNCS 3221, pp. 1–2, 2004.

seemingly rather basic methodological insights still emerging in recent years with surprising frequency. In other words, despite 20 years of research on FPT algorithm design, the area still seems to be in its infancy.

Most of the work on FPT algorithm design to date has been theoretical, sometimes highly theoretical and seemingly completely impractical (such as in FPT results obtained by well-quasiordering methods). The last few years have seen two developments concerning the practical relevance of FPT algorithmics.

First of all, there has been a surge of implementation efforts that have helped to clarify the somewhat complex relationship between the theoretical games and practical algorithms. In many cases, adaptations of the structural and algorithmic insights of the theoretical FPT results are yielding the best available practical algorithms for problems such as Vertex Cover, regardless of whether the parameter is small or not! For example, polynomial-time pre-processing rules uncovered by the kernelization game will *always* be useful, and are sometimes of substantial commercial interest. Similarly, clever branching strategies that are often a key part of advances in the $f(k)$ game are of universal significance with respect to backtracking heuristics for hard problems. Some striking examples of this interplay of FPT theory and practical implementations will be described by Langston in his IWPEC 2004 invited address.

Secondly, a broad realization has developed that many implemented practical algorithms for hard problems, that people are quite happy with, are actually FPT algorithms, unanalyzed and not described as such, sometimes even with quite terrible $f(k)$'s (if viewed theoretically) and employing relatively unsophisticated pre-processing. So it does seem that there is plenty of scope for theoretical research on improved FPT algorithms to have considerable impact on practical computing.