Fair Problems and Diverse Solutions in Parameterized Complexity

Tomáš Masařík

In the theoretical research of algorithms and complexity theory, we sometimes still do not understand fundamental questions like the famous problem of P $\neq$ NP. This question asks whether we can solve some hard problems in polynomial-time measured in the input size (which we think of as efficient). Although we do not know how to prove it, it is commonly believed that we cannot solve those hard problems efficiently. In fact, we know only exponential-time algorithms for them. However, that is not the end of the story. There are plenty of ways how to attack those hard problems and perhaps solve them efficiently after all. Our approach focuses on finding an additional measure of the instances, called the parameter. We can then look for an efficient algorithm provided that the parameter is a small constant. There are two large classes of such good algorithms. The better one (FPT), where the speed of the algorithm may depend on the parameter arbitrarily, except the speed is dependent only polynomially on the input size. The worse one (XP), where the algorithm's running time may have the dependence on the parameter in the exponent of the polynomial. Check Table 1 for an intuition why we distinguish those two classes.

k/n	50 (1 day)	100 (3.2T yrs)	500 (10^{133} yrs)	1000 (10^{283} yrs)
5	133 ns | 26 ms	266 ns | 0.8 s	1.3 μs | 43 mins	2.7 μs | 1 day
10	4 μs | 94 days	8 μs| 264 yrs	40 μs | 2.5G yrs	80 μs | 2.6T yrs
25	0.1 s | 10^{24} yrs	0.2 s | 10^{32} yrs	1 s | 10^{49} yrs	2 s | 10^{57} yrs

Table 1: Comparisons of running times between naive exponential (2^n), XP (n^k) and FPT ($2^k n$) algorithms on a current computer (4 cores, 3GHz). Naive running times are in the first row in brackets, FPT | XP running times are within the inner-fields of the table. Shortcut yrs stands for years.

We will study a group of problems with a common theme; We are interested in graph problems where we do not aim to a single solution of the smallest size as is typical. Instead, we focus on two non-traditional approaches. In the first group, we want a solution of some "fair" quality (Fair problems). Roughly speaking, the solution should be somewhat evenly distributed. In the second group, we aim for a small set of relatively good solutions that are as different as possible (Diverse solutions). Our motivation is natural. In case of fair problems the goal is not to provide a solution of small size, but the one which does not discriminate any party. We can go even further. The potential user of the algorithm might not even know what property he is looking for. In that case, diverse solutions offer him a choice from a bunch of very different options. Imagine we are in a role of an architect studio. We decided to automatize the design of floor plans. Of course, we want to let the customers choose, but at the same time, they also do not want to be overwhelmed with too many too similar options.

To study the mentioned problems, we will use the parameterized complexity theory toolbox, which has been used to study them only very recently. Our task is to classify the parameterized complexity of those problems as much as possible, giving us better insight. We plan a systematic study under different structural parameterizations using various expressive powers of logic, which describe the problems. We will also study fair variants of fundamental problems on their own. For the problems that will turn out to be hard, we intend to derive parameterized approximation algorithms. Those do not provide an optimal solution but still give us some guarantee on its value. We will also explore alternative fair costs and diverse measures definitions. We will examine how those changes affect the speed of new and existing algorithms.