Parameterized Tractability of Single Machine Scheduling with Rejection

By Yuan Wei and Maxim Shelopugin

Parameterized Complexity Definition

- Most interesting problems in Complexity Theory are in NP-hard
- NP - hardness of the problem does not tell much about its solvability
- We can view NP-hard problems from a different perspective
- Once realized that the problem is dependent on some parameter, reasoning about its complexity is more convenient

Parameterized Problem

- Problem P solution to which is dependent on a fixed parameter k
- Such k might be an output of a function, which depends on the problem's innate description, but once fixed - is constant
- Allows for pseudo-polynomial solutions for many problems
- A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$

Parameterized Complexity Class FPT

Fixed Parameter Tractable (FPT) class:

There exists an algorithm that solves a decision version of the parameterized problem

$(x, k) \in L?$ in time: $O(f(k)n^c)$

Where f(k) is computable, c is a constant, and n is a size of the problem

This class separates the dependency between the size and parameter

Parameterized Complexity Class XP

XP class:

A problem P is in class XP, if any instance p of P can be solved in $n^{f(k)}$

Again, f(k) is computable, and n is a size of the problem

XP puts the parameter as an exponent, hence the name

FPT is preferable to XP, since it is asymptotically faster

Both parameter and problem description contribute to the complexity

Boolean Circuit

- A directed acyclic graph contains n input gates, any intermediate gates, and a single output gate
- Each gate can be large or small
- Weft of a boolean circuit (t): the maximum number of large gates on any path from an input gate to the output gate

Weighted Circuit SAT:

- Given a boolean circuit, decide if there is an assignment of inputs (0 or 1) to make the output gate be 1, and it has exact k number of 1 inputs.

Parameterized Complexity Class W[]

W hierarchy

W[t] class: a problem P is in W[t] class if there is a constant d and a parameterized reduction from P to the weighted circuit SAT problem with weft = t and depth = d

W[0]-hard: a problem P is in W[0] class, in which weft = t = 0 (FPT is W[0]-hard)

W[1]-hard: a problem P is in W[1] class, in which weft = t = 1

W[t>=1] is not Fixed Parameter Tractable

Reduction in Parameterized Problem

Parameterized reduction from problem P2 to P1: there is an algorithm that maps an instance I2 of P2 with parameter k2 to an instance I1 of P1 with parameter k1, where I1 is a yes-instance of P1 iff I2 is a yes-instance of P2, and such algorithm runs in

$$f(k2)|I2|^c$$

time, and

$$k1 \leq f(k2)$$

A Single Machine Scheduling Problem With Rejection

A set of n jobs: $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ and each job J_j $(j = 1, 2, \ldots, n)$ is associated with two parameters: processing time p_j and rejection cost e_j

A solution has two steps: (i) partition the set $\mathcal{J}$ into two subsets: the set of accepted jobs A and the set of rejected jobs $\overline{A}$. (ii) sequence the set of accepted jobs A on the single machine.

C_j is the completion time of a job j (Completion time C_j = Processing time p_j + Waiting time)

Goal: find a solution that minimizes $F_1 = \sum_{J_j \in A} C_j$ subject to $F_2 = \sum_{J_j \in \overline{A}} e_j \leq R$

where R is the predefined total rejection cost

W[1] Hardness wrt |A| - amount of accepted jobs

- Reduction from kSUM problem
  - kSUM - find a subset of k integers such that their sum is equal to some target T
- kSUM is proved to be W[1] Hard wrt. k
- Parameterized reduction from kSUM where k = |A|
- Very complex proof, induction for both sides
  - If the instance is true for k-SUM, it is true for Scheduling with Rejection
  - If the instance is true for Scheduling with Rejection, it is true for k-SUM

W[1] Hardness wrt |A| - amount of accepted jobs

Construct an instance for the decision version of the scheduling problem wrt parameter |A| with the given instance of kSUM problem (partition and map the scheduling problem wrt parameter |A| related to the kSUM problem)

Prove if having a yes-instance (k numbers summation equal to the goal) for the kSUM problem, iff there exists a yes-instance (a feasible schedule) for the constructed scheduling problem wrt parameter |A| (this is proved by induction, some sub-steps are proved by contradiction)

FPT wrt v_p - different processing times

- Scheduling Problem with rejection is FPT wrt different processing times
- Provide Mixed Integer Convex Programming formulation to solve the problem
- Mixed Integer Convex Programming (MICP) is a mathematical optimization to minimize the convex function over the convex domain, with some variables constrained to be integers while others be non-integers
- MICP formulation is FPT wrt number of integer variables
- Formulate each constraint as a Convex Function

FPT wrt v_p - different processing times

Construct convex functions which represent the scheduling problem wrt parameter v_p

Separate Jobs into k sets with equal processing times, sort sets

Sort each set with ascending order of rejection cost

Let x_i be accepted and y_i rejected jobs in such sets

$$\Sigma_{i=1}^{k} \Sigma_{j=1}^{y_i} e_{i,j} \leq R$$

$$\Sigma_{J_{ij} \in A} C_{ij} = \frac{1}{2} (\Sigma_{i=1}^{k} p_i x_i + \Sigma_{i=1}^{k} g_i^2 (\frac{1}{p_i} - \frac{1}{p_{i+1}}))$$

By Knop and Koutecky (2018) - sum of completion times is convex

FPT wrt p_{max} - maximum processing time

Separate into sets based on processing time

Sort sets, sort elements in sets in ascending order of rejection cost

Define $F_j(C, l)$ - min total rejection cost, with l accepted and C total time

$$F_j(C, l) = min[F_{j-1}(C, l) + e_j, F_{j-1}(C - lp_j, l - 1)]$$

A state $F_j(C, l)$ is reachable by rejection from state *j-1* with addition of rejection cost

Or from state *j-1* by accepting and updating previous value

Runs in $O(n^4 p_{max})$

Parameterized Complexity Conclusion

- Is a tool to find efficient algorithms for NP-hard problems in practice
- Has the complexity classes to identify how hard the parameterized problem is
- Can use some general methods to classify parameterized problems
- Not all parameterization provides an efficient solution

Questions ?