Conditional Preconditioning*

Sandor Szabo† Bogdan Zavalnij‡

Abstract

In the field of combinatorial optimization preconditioning methods prove to be of crucial importance. We present a generic way of modifying some of these methods so that they are more effective in reducing the problem size.

1 Problem Specification.

In this extended abstract we shall speak about kernelization or preconditioning graphs in respect of optimization or decision problems on graphs. Kernelization is in the center of nowadays research and being used in several different aspects. It is the base of Fixed Parameter Tractable algorithms, it is used in Branch-and-Reduce methods, or by reducing a huge problem to a smaller one makes an unsolvable problem solvable. Making preconditioning and kernelization more effective helps all of these.

In the present short paper, for the sake of simplicity we will use the k-clique decision problem as a particular method. That is if given a simple graph we need to decide if a clique of size k, that is k different nodes that are all connected, are present in the graph or not. Obviously this is just an example, and the method described here can be used for other problems as well.

We will speak about kernelization and preconditioning in respect of this specific problem. Also, although these terms are not the same and have some differences, we shall concentrate on those properties which are common, and so use the two terms interchangeably: a method which makes the problem more easy to solve.

2 Kernelization.

There is no exact definition of kernelization, but one can speak about transforming the graph G into G' such way, that the transformed graph has two specific properties:

1. The k-clique problem gives the same answer for G and G'. That is if a k-clique present in one, then it is present in the other. Also, if G does not contain a k-clique, then G' does not contain either.[1]
2. The problem is more easily solved for G' then for G. Usually one prescribes that the 'size' of the transformed graph to be smaller. That is G' has less nodes and/or edges then G.

There are different techniques that fall into this category, for example [1, 2, 3]. Again, we shall choose a few of them as examples but the results may be extended to other methods as well. We shall closely look at some methods that delete nodes or edges from the graph. From those we bring just two examples here, the inspection of the color degree and dominance.

Let v be a node of G and let H be the subgraph of G induced by the set $N(v)$. We call the number of colors of the nodes of H the color degree of the node v. Let $e = \{u, v\}$ be an edge of G and let H be the subgraph of G induced by the set $[N(u) \cap N(v)]$. The number of colors of the nodes of H is called the color degree of the edge $e = \{u, v\}$. Of course coloring by itself is an NP-hard problem, so we must either use a heuristic approach, or reuse the coloring of the graph if it derives from the problem itself (see [5]).

LEMMA 2.1. *A node whose color degree is less than $(k-1)$ can be deleted from G when we are looking for a k-clique in G. An edge whose color degree is less than $(k-2)$ can be deleted from G when we are looking for a k-clique in G. (We leave the two nodes of the edge intact.)*

Let u, v be non adjacent nodes of the graph G. If $N(u) \subseteq N(v)$, then we say that node v dominates node u. Let $e = \{u, v\}$, $f = \{x, y\}$ be edges of G. If the nodes $\{u, v, x, y\}$ are not all adjacent in G and $[N(u) \cap N(v)] \subseteq [N(x) \cap N(y)]$, then we say that edge f dominates edge e. The following result is proved in [4].

LEMMA 2.2. *If node v dominates node u, then u maybe be deleted from G when we are looking for a k-clique in G. If edge f dominates edge e, then e maybe deleted from G when we are looking for a k-clique in G. (We leave the two nodes of the edge intact.)*

We shall examine a special property of the presented kernelization methods. It is about the number of k-cliques before and after applying the method in question. In the case of color degree we always delete a node

*Supported by NKFIH Fund No. SNN-135643.

†University of Pecs, Hungary.

‡Renyi Institute of Mathematics, Hungary.

[1]For some kernelization methods the value of k also changes, like in the struction transformation, where we search for a k-clique in G, and k'-clique in G', $k' = k - 1$. [2]

(or edge) that cannot be part of any k-clique. So this preconditioning is not deleting any k-clique but leaves them intact. For the dominance this is quite different. It *can* delete a node or an edge from a k-clique, provided that at least one k-clique stays intact. That is the number of k-cliques may decrease after applying this preconditioning. Obviously, one can always decide for given preconditioning method in which category it falls.

3 Conditional edges.

Our main result we would like to present in this extended abstract is about using conditional edges. It is clear if a transformation deletes a node or an edge without changing the answer for the main question ("Is there a k-clique present?"), we can put that node or edge back as well without changing this answer. Putting back nodes does not seem to be interesting, but putting back edges may be indeed. In case of checking if v dominates u, it is desirable that v have as many neighbors as possible, so putting back some edges may help detecting a dominance. (In case of color degree it won't help though.) In this way we consider such edges as conditional. We put them back temporally, if in need, for carrying out a desired operation.

But a problem can arise. We would like to put back an edge, and use it for deleting a node or an edge. In the end we obviously want to take away that edge. But is it possible? Some problems may arise and we get into a vicious circle. As a very simplified example consider two edges, $e = \{u, v\}$, $f = \{x, y\}$. First, it turns out that edge f dominates edge e, so we delete e. Second, later in the process, we test if node u dominates node x, but in order to fulfill this property we need to put back edge e, and so we delete node x. That means deleting edges going out from x as well, among them the edge $\{x, y\}$. But if we take away $\{u, v\}$ after the process, we may loose all k-cliques! As we deleted edge e because if it was contained in a k-clique, there must be a k-clique containing f. We simply cannot delete both edges and ensure that at least one k-clique remains. Also note, that such wrong circles may be much longer in opposed to this simple example.

There can be different solutions for the proposed problem of mutual deletion. We present here the most simple one. The distinction is based on the special property of clique deletion we described earlier. If we consider only those edges for being put back that were deleted by a method which leaves all k-cliques intact we cannot go in a wrong circle. As putting back and taking away that edge later never touches any k-clique. This way it is impossible to loose all k-cliques.

As an addition we would like to point out, that it is also possible to check a pair of unconnected nodes $\{u, v\}$ for the possibility of connecting them in the original graph G. This is done by a simple procedure: we check the color degree of this if-would-be edge, and if it can be deleted, then we can connect these two nodes without changing the answer for the presence of a k-clique. We obviously do not want to put an extra edge into the graph, that would contradict our aim of kernelization, but we can consider such an edge as conditional in the later process when searching for dominance.

4 Implementation, red-black edges.

This process may seem quite complicated, but a simple data structure is sufficient for programming it. Namely, we consider the edges of the original graph G colored all black. Later, if we delete an edge by color degree, we do not delete it, but rather color that edge red. In contrast, if we delete an edge by dominance, we do delete it.

We must alter the algorithm for deciding dominance slightly. If we check if u dominates v, we use the neighborhood of u including red edges as well. In contrast for neighborhood of v we only use black edges.

The presented ideas are still to be considered as work in progress. Nevertheless, we implemented them, and checked on a few problems arising from our other works. In the case of one particular example this method enabled our program to delete 7% more nodes using node dominance, and 30% more edges using edge dominance. Not a crucial, but a helpful difference. In our future work we will try to include other techniques to use conditional edges.

References

[1] Akiba, Takuya and Iwata, Yoichi, *Branch-and-reduce exponential/fpt algorithms in practice: A case study of vertex cover*, Theoretical Computer Science, 609 (2016) pp. 211–225.

[2] Ebenegger, Ch., Hammer, P.L. and De Werra, D., *Pseudo-Boolean functions and stability of graphs*, North-Holland mathematics studies, 95 (1984) pp. 83–97.

[3] Gellner, A., Lamm, S., Schulz, Ch., Strash, D. and Zavalnij, B., *Boosting Data Reduction for the Maximum Weight Independent Set Problem Using Increasing Transformations*, In: Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX21) SIAM (2021) pp. 128–142.

[4] S. Szabó, *Parallel algorithms for finding cliques in a graph*, Journal of Physics: Conference Series, 268 (2011)

[5] Szabo S. and Zavalnij B., *Clique search in graphs of special class and job shop scheduling*, Journal of Combinatorial Optimization. (submitted)