On the Kernelization of Global Constraints
Clément Carbonnel, Emmanuel Hébrard

To cite this version:
Clément Carbonnel, Emmanuel Hébrard. On the Kernelization of Global Constraints. International Joint Conference on Artificial Intelligence (IJCAI 2017), Aug 2017, Melbourne, Australia. 7p., 2017. <hal-01670318>

HAL Id: hal-01670318
https://hal.laas.fr/hal-01670318
Submitted on 21 Dec 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
On the Kernelization of Global Constraints

Clément Carbonnel  
University of Oxford  
clement.carbonnel@cs.ox.ac.uk

Emmanuel Hebrard  
CNRS, LAAS-CNRS, Université de Toulouse  
hebrard@laas.fr

Abstract

Kernelization is a powerful concept from parameterized complexity theory that captures (a certain idea of) efficient polynomial-time preprocessing for hard decision problems. However, exploiting this technique in the context of constraint programming is challenging. Building on recent results for the VERTEX COVER constraint, we introduce novel “loss-less” kernelization variants that are tailored for constraint propagation. We showcase the theoretical interest of our ideas on two constraints, VERTEX COVER and EDGEDOMINATING SET.

1 Introduction

Global constraints are central to the success of constraint programming for solving real-world computational problems. Represented algorithmically (through a propagator) rather than extensionally, they can encapsulate powerful reasoning while being time and memory efficient. Because a global constraint can be just any NP subproblem, finding a support for a given variable-value pair is potentially NP-hard. For such constraints, which are quite common [Bessiere et al., 2004], enforcing arc consistency is costly, and relaxations of various types are often employed in stead of complete algorithms.

A study of the parameterized complexity of global constraints, and of their relevant parameters, can help to address this issue. For instance, propagating the NVALUE constraint is NP-hard in general [Bessiere et al., 2006], but a refined analysis using parameterized complexity has shown that this is FPT when the parameter is the number of holes in the domains [Bessiere et al., 2008]. This FPT algorithm has been subsequently improved via kernelization [Gaspers and Szeider, 2011], which is used as a preprocessing for probing: for every variable-value pair, the problem of deciding if a support exists is solved on the kernel.

In [Carbonnel and Hebrard, 2016], a novel family of kernels, called loss-less kernels, was introduced for subset minimization problems. Unlike general kernelization, loss-less kernels maintain complete information on all supports and thus define sound propagation rules. A powerful feature of these new kernels is the ability to propagate a constraint with a single probe of the kernel, or even without a probe, although yielding a weaker filtering in this case. Naturally, this comes at the price of a larger bound on the kernel size.

In this paper, we propose a definition of loss-less kernelization relevant to general constraints, hence less restrictive than the one in [Carbonnel and Hebrard, 2016]. Moreover, these definitions are carefully crafted to make sense from a theoretical and from a practical point of view. Next, we focus on the case of the VERTEX COVER constraint with parameter \( k \) equal to the cardinality of the minimum cover. Using a novel framework of \( z \)-rigid crown decompositions, we demonstrate the existence of a wide range of loss-less kernels whose size depend on the “slack”, i.e. the difference between the size of the smallest vertex cover and \( k \). Finally we review the kernel introduced in [Hagerup, 2012] for EDGE DOMINATING SET and show that it is in fact loss-less (although in a weaker sense than our kernels for VERTEX COVER).

2 Formal Background

A CSP instance is a triple \((X, D, C)\) where \( X \) is a finite set of variables, \( D \) is a finite set of values and \( C \) is a set of constraints. A constraint \( C \) is pair \((R_C, S_C)\) where the scope \( S_C \) is a subset of \( X \) and \( R_C \) is a Boolean predicate on \( S_C \). A solution to a CSP instance is a mapping \( \phi : X \rightarrow D \) such that for every constraint \( C \in \mathcal{C} \), the predicate \( R_C \) is true for the componentwise mapping of \( \phi \) to \( S_C \). A global constraint is a family of constraints induced by a predicate defined over different scope sizes. Constraint solvers represent the search space by associating a domain, that is, a set of values \( D(x) \) to every variable \( x \in X \). Moreover, they rely on so-called propagation algorithms, one for each (global) constraint to prune the search space by reducing the domains. Given a constraint \( C \) with \( x \in S_C \), a pair \((x, d)\) is consistent for \( C \) if it has a support, that is, a mapping \( \psi : S_C \rightarrow D \) such that \( \forall y \in S_C, \psi(y) \in D(y), \psi(x) = d \) and \( R_C \) is true on \( \psi(S_C) \). A constraint \( C \) is said arc consistent if every pair \((x, d)\) with \( x \in S_C \) and \( d \in D(x) \) is consistent for \( C \).

The complexity of a problem can be more finely characterized by considering parameters besides the size of the input. Parameterized complexity aims at understanding which parameters are relevant to explain the hardness of a problem. Given a problem \( \mathcal{P} \) and a parameter \( p \), \((\mathcal{P}, p)\) is in the FPT class if there exists an algorithm that can decide an instance \( x \) of \( \mathcal{P} \) in time \( f(p) |x|^{O(1)} \) where \( f \) is a computable function. A kernelization for a parameterized problem \((\mathcal{P}, p)\) is a...
polynomial-time computable function $K$ that maps each instance $(x, k)$ of $(P, p)$ to an instance $(x', k')$ of the same problem such that $(x, k)$ is a yes-instance if and only $(x', k')$ is, $|x'| \leq f(k)$, and $k' \leq g(k)$ for some computable functions $f, g$. There is intense research both on FPT algorithms and kernelization methods [Cygan et al., 2015].

3 Loss-less Kernels

Let $\mathbb{Q} = \mathbb{Q} \cup \{\infty, -\infty\}$ denote the set of rational numbers with infinity. A (n-ary) cost function over a finite domain $\mathcal{D}$ is a function $\pi : \mathcal{D}^n \to \mathbb{Q}$. Throughout the paper we consider NP problems that correspond to the decision version of minimizing an input cost function $\pi$ drawn from a fixed (but infinite) family $\Pi$ of cost functions.

**II-Minimization-Deci**

**Input:** A set of variables $X = \{x_1, \ldots, x_n\}$ with domains $D(x_1), \ldots, D(x_n) \subseteq \mathcal{D}$, $\pi \in \Pi$ over $\mathcal{D}$ of arity $n, k \in \mathbb{Q}$

**Question:** Does there exist an assignment $\phi : X \to \mathcal{D}$ such that $\forall x \in X, \phi(x) \in D(x)$ and $\pi(\phi(x_1), \ldots, \phi(x_n)) \leq k$?

Note that any problem corresponding to a Boolean predicate imposed upon a set of variables (that is, a constraint) can be written in that form by letting $\phi$ return $-\infty$ if the predicate is satisfied and $\infty$ otherwise. Assignments to $X$ that respect the domains and have finite cost are called feasible, and solutions are feasible assignments of cost less than $k$. For a given instance $I$ with variable set $X$, $\sigma(I)$ denotes the minimum cost of a feasible assignment to $X$, $U(I)$ denotes the set of all pairs $(x, d)$ with $x \in X$ and $d \in D(x)$, and $\Sigma(I)$ denotes the set of all pairs $(x, d) \in U(I)$ such that there exists a solution $\phi$ to $I$ with $\phi(x) = d$ (in the CSP world, this is the set of arc consistent variable-value pairs). Depending on the context, we may write $k_I$ instead of $k$ to stress that $k$ belongs to the instance $I$. We shall assume that the choice of encoding for $\pi$ is part of the definition of $\Pi$, but to preserve membership in NP we will assume that $\pi$ can be evaluated in polynomial time. For example, VALUED CSP is a problem of this kind where $\pi$ is a sum of smaller cost functions (constraints), each of which is represented using tables of costs. Given a fixed family $\Pi$ of cost functions, we can turn an instance $I$ of II-Minimization-Deci into a global constraint $C_{\pi}$ whose tuples are the solutions to $I$. The scope of that constraint is $(x_1, \ldots, x_n, K)$, where $K$ is a cost variable, and the predicate of that constraint is $\pi(x_1, \ldots, x_n) \leq K$. The purpose of this section is to identify new notions of kernelization for those cost-minimization problems that are relevant to the propagation of their associated constraint.

As classical kernels typically use rules to identify which values are relevant to the decision problem (i.e. finding a tuple-solution), we want to define kernels which capture the values that are relevant to propagation. In other words, if the parameter is $p$ we want a subproblem of size $f(p)$ such that the knowledge of propagated values in this subproblem allows complete propagation of all values in polynomial time. This is much more consistent with the spirit of kernelization, extends smoothly constraints with polynomial-time propaga-

tors and yields a propagation algorithm with running time $O(g(p) + |I|^{O(1)}p^{O(1)})$, instead of $O(|I|^{O(1)}g(p))$ for probing plus classical kernelization.

Now that we have identified the foundations of the desired notion of kernelization, we can refine it for cost minimization constraints using a recurrent phenomenon in CSP solving. Let us consider a cost minimization constraint $C_{\pi}$ with cost variable $K$ and underlying instance $I$. Let $\tilde{K}$ be the maximum value of $K$. If $\sigma(I)$ is much smaller than $\tilde{K}$ then $C_{\pi}$ is likely to contain many tuples, hence propagating this constraint is unlikely to filter out values. However, during search the domain of $K$ will be reduced until $\tilde{K} \leq \sigma(I) + z$ for a small constant $z$. The constraint will get tighter, hence more likely to entail propagation. Therefore, it makes sense to bolster the propagation methods when $z$ is small (e.g., we have a lower bound of $\sigma(I)$). From the kernelization perspective, this means that we should prioritize kernels whose size will provably decrease as $z$ gets closer to 0, or even kernels that only exist for small values of $z$. Thus, instead of large kernels that preserve all supports, we can focus on supports whose cost is at most $z$ from the optimum cost - at the small price of being useful only when the constraint becomes tight.

We will define a hierarchy of three different types of kernels for propagation, which we call $z$-loss-less and will be respectively labelled as direct, weak and strong. Direct $z$-loss-less kernels, which are the smallest and most general, follow closely the thought process of the above paragraphs.

**Definition 1** Let $z \in \mathbb{Q}$ and $\Pi$ be a fixed family of cost functions. A direct $z$-loss-less kernelization of II-Minimization-Deci parameterized by $p$ is a function that maps in polynomial time each parameterized instance $(I, p)$ of II-Minimization-Deci to a new parameterized instance $(I', p')$, such that

(i) There exist computable functions $f, g$ such that $|I'| \leq f(p)$ and $p' \leq g(p)$;

(ii) $I$ is a yes-instance if and only if $I'$ is;

(iii) There exists a polynomial-time algorithm $A$ which on input $(I, \sigma(I'))$ computes $\sigma(I)$;

(iv) There exists a polynomial-time algorithm $B$ which outputs $\Sigma(I)$ on input $(I, \sigma(I'), \Sigma(I'))$, if $\sigma(I) \geq k_I - z$.

Weak $z$-loss-less kernels improve on their “direct” counterpart by allowing polynomial-time propagation of non-kernel values even if only the optimum cost $\sigma(I')$ of the kernel $I'$ is known. This gives a nice flexibility: with $n$ variables and $d$ values in each domain, one can either be satisfied by propagating the up to $nd - f(p)$ non-kernel values at the cost of solving a single FPT optimization subproblem, or push further and decide to propagate the kernel values as well by solving a number of subproblems that depends only on the parameter. Note that in practice we often have $U(I') \subseteq U(I)$ although this is not required in the following definitions.

**Definition 2** Let $z \in \mathbb{Q}$ and $\Pi$ be a fixed family of cost functions. A weak $z$-loss-less kernelization of II-Minimization-Deci parameterized by $p$ is a function that maps in polynomial time each parameterized instance $(I, p)$ of II-Minimization-Deci to a new parameterized instance $(I', p')$, such that
(i)-(iv) It is a direct z-loss-less kernelization;

(v) On input \((I, \sigma(I))\) and whenever \(\sigma(I) \geq k_1 - z\), the algorithm \(B\) outputs \(\Sigma(I) \cap (\mathcal{U}(I) \setminus \mathcal{U}(I'))\).

In the next sections we present weak \(z\)-loss-less kernels for VERTEX COVER with an additional property that turns out to be highly relevant to propagation: in these kernels, the algorithm \(B\) is capable of performing partial propagation if a lower bound of \(\sigma(I)\) is given in input instead of \(\sigma(I)\). This means that invoking \(A\) becomes optional, and hence the kernel remains useful even if the FPT subproblem is not solved at all. More precisely, in our case the algorithm \(B\) converges towards complete propagation as the lower bound gets closer to \(\sigma(I)\). Of course, if \(B\) does nothing unless \(\sigma(I)\) is given as input it technically converges; but the key observation is that \(B\) cannot do that in general because it is a polynomial-time algorithm and checking if the input is indeed \(\sigma(I)\) is likely to be difficult. If an algorithm \(B\) with this property exists, we say that the kernelization is strong rather than weak.

**Definition 3** Let \(z \in \mathcal{O}\) and \(\Pi\) be a fixed family of cost functions. A strong \(z\)-loss-less kernelization of \(\Pi\)-MINIMIZATION-DECISION parameterized by \(p\) is a function that maps in polynomial time each parameterized instance \((I, p)\) of \(\Pi\)-MINIMIZATION-DECISION to a new parameterized instance \((I', p')\), such that

(i)-(iv) It is a direct \(z\)-loss-less kernelization;

(v) On input \((I, l)\) where \(l \in \mathcal{Q}, l \leq \sigma(I)\), the algorithm \(B\) outputs a set \(\Sigma(\mathcal{I}) \subseteq \mathcal{U}(\mathcal{I}) \setminus \mathcal{U}(\mathcal{I'})\) such that

- For every \(l_1, l_2\) with \(l_1 \leq l_2, \Sigma_{l_1} \subseteq \Sigma_{l_2};\)
- \(\sigma(I) < k_1 - z\) or \(\Sigma_{\sigma(I)} = \Sigma(I) \cap (\mathcal{U}(I) \setminus \mathcal{U}(I')).\)

There are numerous ways to generalize these definitions - for instance, turning \(z\) into a function of the parameter or considering kernels for higher-order consistencies that preserve supports for simultaneous assignments to variables. We leave these generalizations for future research.

**Related work.** Damaschke defined full kernels [Damaschke, 2006] for subset minimization problems which are particular cases of our framework. Such problems ask for a subset \(S\) of size \(\leq k\) of a universe \(U\) with some property \(P\) which is \(\triangleright\)-closed, i.e. \(Y \triangleright X\) has the property \(P\) whenever \(X\) has. A full kernel is a subset of \(U\) containing the union of all inclusion-minimal solutions. When computable in polynomial time, they are strong \(\omega\)-loss-less kernels where \(\sigma(I) = \sigma(I')\) and \(B\) does not remove any values unless the input lower bound is exactly \(k\), in which case it no longer finds supports for the value \(\text{true}\) in the domain of non-kernel variables. This is an example for which the convergence of \(B\) to complete propagation is the worst possible, since \(B\) does nothing until the lower bound is equal to \(k\).

**Enum-kernels** are kernels that preserve all necessary information to enumerate all solutions with FPT delay [Creignou et al., 2013]. In that aspect they are incomparable with our kernels which are required to preserve one support (i.e. solution) for each variable-value pair, and the existence of these supports must be decidable in polynomial time (not FPT) from the knowledge of kernel supports. However, since the core idea is relatively similar we can expect many direct loss-less kernels to be enum-kernels as well (and vice versa). Finally, in the very specific context of strong backdoors Samer and Szeider have introduced loss-free kernels [Samer and Szeider, 2008], which are essentially identical to full kernels.

### 4 Vertex Cover

We first introduce some elementary notions of graph theory and relevant notations. A graph is an ordered pair \(G = (V, E)\) where \(V\) is a set of vertices and \(E\) is a set of edges, that is, binary subsets of \(V\). The subgraph of \(G = (V, E)\) induced by a subset of vertices \(W\) is denoted \(G[W] = (W, 2^W \cap E)\).

Given a graph \(G = (V, E)\) and \(W, I \subseteq V, W \cap I = \emptyset\), we write \(G_{B}^{W, I}\) for the bipartite graph with bipartition \(W, I\) and edge set \(\{(u, v) \in E : u \in W \land v \in I\}\). An independent set is a set \(I \subseteq V\) such that no pair of vertices in \(I\) is in \(E\). A matching is a subset of pairwise disjoint edges. A vertex cover of \(G\) is a set \(S \subseteq V\) such that every edge \(e \in E\) is incident to at least one vertex in \(S\), i.e., \(S \cap e \neq \emptyset\).

The size of a matching is a straightforward and widely used lower bound to that of any vertex cover because covering the matching itself requires at least one vertex per edge. The VERTEX COVER problem asks, given a graph \(G\) and an integer \(k\), whether \(G\) has a vertex cover of size at most \(k\). It is \(NP\)-complete [Garey and Johnson, 1979] and commonly parameterized by the solution size \(k\). The problem is easily seen as FPT via a bounded search tree, and more involved techniques yield an exact FPT algorithm with running time \(O(1.2738^k + |V|k)\) [Chen et al., 2006].

#### 4.1 Preliminaries: Classical Kernelization

In this section we survey the known kernelization methods for VERTEX COVER, and in particular crown-based kernels which will serve as a baseline for our contributions. The simplest kernelization is Buss’s rule. The idea is straightforward. If a vertex has degree \(k + 1\), it must belong to every size-\(k\) vertex cover as otherwise we would have to include its whole neighbourhood; thus we can safely remove it from the graph and decrement \(k\) by 1. If this rule is no longer applicable, then every vertex can cover at most \(k\) edges, and therefore no size-\(k\) vertex cover exists unless the graph has at most \(k^2\) edges and \(k^2 + k\) non-isolated vertices. A more refined kernelization algorithm works using decompositions called crowns.

**Definition 4** Let \(G = (V, E)\) be a graph. A crown decomposition of \(G\) is a partition \((H, W, I)\) of \(V\) such that

(i) \(I\) is an independent set;

(ii) There is no edge between \(I\) and \(H\);

(iii) There is a matching \(M\) between \(W\) and \(I\) of size \(|W|\).
because one has to actually compute a “good” crown decomposition, which is not obviously easy. For this, two competing methods exist: one based on maximal matchings that leaves a residual instance $G[H]$ with $3k$ vertices [Abu-Khzam et al., 2007], and one that uses the relaxation of the LP formulation instead and yields a kernel of size $2k$ [Nemhauser and Trotter Jr, 1975]. The best known kernel for VERTEX COVER has size $2k - \varepsilon \log(k)$ by using a reduction to ALMOST-2-SAT on top of the LP method [Lampis, 2011], but the algorithmic cost becomes quickly prohibitive as $\varepsilon$ increases.

There is a fundamental difference between Buss’s rule and crown-based kernelization. Let us turn to the formulation as a cost function minimization problem, with one Boolean variable for each $v \in V$ that decides if $v$ should be in the cover. When Buss’s rule can be applied to a vertex $v$, it belongs to all vertex covers of size at most $k$. This means that the value $0$ in the domain of $v$ is inconsistent, and $1$ is consistent if and only if a solution exists. After removal of those, every other non-kernel vertex is isolated, and for these the value $1$ is consistent if and only if the minimum cost is strictly less than $k$. Overall, this is a strong $\infty$-loss-less kernelization. On the other hand, crown decompositions assert that the taking the vertices in $W$ in the cover is at least as good as taking those in $I$, but it may happen that size-$k$ (and even minimum-size) vertex covers contain vertices in $I$. In other words, we preserve at least one minimum-size cover but information on supports may be lost in the process. This idea is formalized more rigorously in the next proposition, which is definitely not surprising but included for completeness. A full proof based on a reduction from MINIMAL CSP can be found in [Carbonnel, 2016].

**Proposition 1** Unless $P = NP$, there is no polynomial-time algorithm that takes as input:
- A graph $G$ and a crown decomposition $(H, W, I)$ of $G$,
- The minimum size of a vertex cover of $G[H]$,
- The list of all vertices of $H$ that belong to all minimum-size vertex covers of $G[H]$, and
- The list of all vertices of $H$ that belong to no minimum-size vertex covers of $G[H]$
and decides if there exists a vertex in $W$ that belongs to all minimum-size vertex covers of $G$.

As a consequence crown decompositions are unsafe for computing loss-less kernels, even for direct ones and with $z = 0$. To address this issue, Chlebík and Chlebíková defined special crown decompositions for which every minimum-size vertex cover of $G[H]$ contains every vertex in $W$ and none in $I$ [Chlebík and Chlebíková, 2008]. These crowns are said to be strong. An optimal strong crown decomposition can be found in polynomial time using a variant of the LP method, where “optimal” means that $H$ does not have a strong crown decomposition and $|H| \leq 2k$ (thus matching the best known bound for general crown kernelization). When the upper bound $k$ is the minimum size of a vertex cover, this kernelization method gives simple rules to completely propagate non-kernel vertices: this is a strong 0-loss-less kernelization.

![Figure 1: A crown decomposition. When ignoring the dashed edges, there exists a minimum-size vertex cover of $G[W \cup I]$ that does not contain $W$ (circled vertices) hence the decomposition is not rigid. With the dashed edges, it becomes 1-rigid.](image)

### 4.2 $z$-Loss-Less Kernels for Vertex Cover

For a graph $G$, the VERTEX COVER constraint has one cost variable $k$ and one Boolean variable $x_v$ for each vertex $v$. Its predicate is $\pi(X) \leq k$, where $\pi$ returns $\infty$ if $S = \{v \mid x_v = true\}$ is not a vertex cover of $G$ and $|S|$ otherwise. On the scale of $z$-loss-less kernelization, Buss’s rule and strong crowns correspond to the two extreme cases $z = \infty$ and $z = 0$. Buss’s kernel is large (quadratic in $k$) but the propagation algorithm $B$ can be used at any time. In contrast, the strong crown kernel is almost as small as a kernel can be but does not propagate anything unless $k$ is exactly the minimum size of a vertex cover. In this section we show that a full hierarchy of intermediate $z$-loss-less kernels exist between these two, and their size is linear in both $z$ and $k$. To achieve this, we consider the following extension of strong crowns.

**Definition 5** Let $G = (V, E)$ be a graph and $z \in \mathbb{N}$. A $z$-rigid crown decomposition of $G$ is a crown decomposition $(H, W, I)$ of $G$ such that every vertex cover of the bipartite graph $(W, I)$ of size at most $|W| + z$ contains $W$.

This idea is illustrated in Figure 1. $z$-rigid crown decompositions define sound reduction rules for $z$-loss-less kernelizations: if the size $k_{opt}$ of a minimum vertex cover is such that $k_{opt} + z > k$, every vertex in $W$ belongs to all solutions and those in $I$ belong to a solution if and only if $k_{opt} < k$.

The rest of this section is devoted to prove the soundness of our $z$-loss-less kernelization algorithm (for arbitrary $z$) $z$Crown (Algorithm 1). We will make use of the following proposition. A crown decomposition is $\text{straight}$ if $|W| = |I|$; straight crown decompositions are not $z$-rigid for any $z$.

**Lemma 1** ([Abu-Khzam et al., 2007]) There exists a polynomial-time algorithm NT that computes a crown decomposition $\Sigma_N = (H, W, I)$ of any graph $G$ such that all crown decompositions of $G[H]$ are straight.

We now prove key properties of the algorithm $z$Crown. A crown decomposition $(H, W, I)$ is $\text{trivial}$ if $W = I = \emptyset$.

**Lemma 2** For every graph $G$ and integers $k, z \geq 0$, $z$Crown$(G, k, z) = (H, W, I)$ is a $z$-rigid crown and $G[H]$ has no nontrivial $z$-rigid crown decomposition.
Algorithm 1: \texttt{zCrown}(G = (V,E), k, z)

1. \( (H, W, I) \leftarrow \text{NT}(G) \);
2. \textbf{while} there exists a minimum vertex cover \( S \) of \( G^W,I \)
   such that \( W \nsubseteq S \) and \(|S| \leq |W| + z \) \do\n3. \( W \leftarrow W \cap S \);
4. \( I \leftarrow I \cap S \);
5. \textbf{return} \( (H, W, I) \);

Proof: First, we prove that \((H, W, I)\) being a crown is an invariant of the loop Line 2. \( S \) is a vertex cover of \( G^W,I \)
so no edge may exist between \( I \cap S \) and \( W \setminus S \), and because \( I \)
remains an independent set throughout the algorithm we need only show that there is a matching between \( W \setminus S \) and \( I \) of
size \(|W \cap S|\). For the sake of contradiction, let us assume that
no such matching exists. By König’s Theorem, it follows that
there exists a vertex cover \( S' \) of \( G^W,I \) such that \(|S'| < |W \cap S| \),
and \( S' \cap (I \cap S) \) is a vertex cover of \( G^W,I \) since \( I \cap S \)
controls all edges except those between \( W \setminus S \) and \( I \). However,
we have \(|S' \cap (I \cap S)| < |W \cap S| \cup (I \cap S)| = |S| \),
which contradicts the minimality of \( S \).

By Lemma 1, in the output of the algorithm \texttt{NT} at Line 1,
all crown decompositions of \( H \) are straight and thus every \( z \)-rigid crown decomposition of \( G \) must be a subcrown of
\((H, W, I)\). We will prove that the latter property is another
invariant of the loop Line 2. Suppose for the sake of contradiction
that there exists a \( z \)-rigid crown \( \Sigma = (H^z, W^z, I^z)\)
of \( G \) with \( W^z \nsubseteq (W \cap S) \). Then, by definition of \( \Sigma \),
\(|S \cap (W^z \cup I^z)| > |W^z| + z \). Furthermore, because \( \Sigma \) is a
crown there is no edge between \( W \setminus W^z \) and \( I^z \), and therefore
the matching \( M \) between \( W \) and \( I \) witnessing that \((H, W, I)\)
is a crown must match the vertices in \( W \setminus W^z \) with those in
\( I \setminus I^z \). Because \( S \) must hit each edge in the matching, it
follows that \(|S| \geq |S \cap (W^z \cup I^z)| + |W \setminus W^z| > |W| + z \), a
contradiction with the loop condition.

Finally, because the loop Line 2 exits only when \((H, W, I)\)
from its own task to bound the size of the residual graph
\( G[H] \) produced by \texttt{zCrown} by a function of \( z \) and \( k \).
To this purpose, we will make a small detour and consider decompositions that are not quite crowns. A partition \((H, W, I)\)
of the vertex set of a graph \( G \) is an almost-crown if \( I \) is an
independent set and \( N(I) = W \) (an almost-crown is simply
an alternative representation of independent sets). The
width \( \delta(\Sigma) \) of an almost-crown \( \Sigma = (H, W, I) \) is the size of a
maximum matching between \( I \) and \( W \). A crown is then an
almost-crown \( \Sigma \) with \( \delta(\Sigma) = |W| \).

Proposition 2 Let \( \Sigma = (H, W, I) \) be an almost-crown of a
depth \( G = (V,E) \) with no isolated vertices. If \(|I| > (z + 1)\delta(\Sigma) \),
then \( \Sigma \) contains a \( z \)-rigid subcrown.

Proof: We proceed by induction on the value of \( \delta(\Sigma) \). If
\( \delta(\Sigma) = 1 \), every vertex \( i \in I \) has the same (unique) neighbour
\( v \in W \). It follows that \( W = \{v\} \) and \( \Sigma \) is itself a \( z \)-rigid
subcrown. This is true for all almost-crowns \( \Sigma' \) with \( \delta(\Sigma') \leq k \).

5 Edge Dominating Set: \( \infty \)-Loss-Less Kernel

A subset of edges \( D \subseteq E \) of a graph \( G = (V,E) \) is an edge
dominating set (EDS) if every edge in \( E \) shares a vertex with at
least one edge in \( D \). It is independent (IEDS) if the edges in
\( D \) are pairwise disjoint. The EDGE DOMINATING SET problem
asks, given a graph \( G \) and an integer \( k \), whether there exists
an EDS of \( G \) of cardinality at most \( k \). This problem is
NP-complete and remains so even if \( G \) is bipartite with
maximum degree 3 [Yannakakis and Gavril, 1980]. The
correcting constraint has one Boolean variable \( x_e \) for each
edge \( e \) and a cost variable \( K \). Its predicate is \( \pi(X) \leq K \).
where $\pi$ returns $\infty$ if $D = \{e \mid x_e = \text{true}\}$ is not an EDS of $G$, and $|D|$ otherwise. The best known kernelization for this problem parameterized by $k$ is found in [Hagerup, 2012]. Due to space constraints we do not explain in detail Hagerup’s kernelization; instead we extract from his work a proposition that summarizes the main result that we will use.

**Proposition 3 ([Hagerup, 2012])** There exists a polynomial-time algorithm that computes from an input graph $G = (V, E)$ a subset $A' \subseteq V$ such that:

- Every vertex in $A'$ appears in all size-$k$ IEDS of $G$.
- The graph $G' = (V', E')$, obtained by removing from $G$ all vertices with only neighbours in $A'$ and then adding to each vertex $x \in A'$ a new degree-1 neighbour $d_x$, has at most $\max(6k, \frac{1}{2}k^2 + \frac{1}{2}k)$ vertices.

The contribution of this section is a proof that Hagerup’s algorithm is in fact a direct $\infty$-lossless kernelization. The next lemma asserts the equivalence with respect to propagation of the EDS and IEDS problems. Its proof is inspired by Harary’s work on the relationship between edge dominating sets and maximal matchings [Harary, 1969].

**Lemma 3** Let $e$ be an edge of a graph $G$ and $k \geq 0$. Then, $e$ belongs to no (resp. all) EDS of size at most $k$ of $G$ if and only if $e$ belongs to no (resp. all) IEDS of size at most $k$ of $G$.

**Proof:** We start with the “no” part of the claim. One implication is trivial: if $e$ belongs to no-size-$k$ EDS, then it belongs to no-size-$k$ IEDS. For the converse implication, let us assume that $e$ belongs to some size-$k$ EDS $D$. We will construct an IEDS at least as small as $D$ that contains $e$ as well.

First, we build from $D$ a minimal EDS of $G$ that contains $e$ as follows. First, we isolate $e$ from the other edges in $D$ by replacing each edge $e' = (x, y) \in D$ sharing an endpoint $x$ with $e$ by an arbitrary edge $e''$ incident to $y$ that does not belong to $D$. If no such edge $e''$ exists, we simply remove $e'$ from $D$. By construction, after this procedure $D$ remains an EDS of $G$ and its size has not increased. Then, we attempt to greedily remove from $D$ as many edges as possible while maintaining the property that $D$ is an EDS. Once a fixed point is reached, $D$ is a minimal EDS of $G$ that contains $e$.

Now, if $D$ is not independent, there exist two edges $e_1 = (u, v)$ and $e_2 = (v, w)$ sharing an endpoint $u$. Because $e$ does not share any endpoint with other edges in $D$, we can assume $e_1 \neq e$. Because $D$ is minimal, there exists at least one edge $e_1'$ incident to $u$ that is dominated only by $e_1$. Then, we remove $e_1$ from $D$ and add $e_1'$ instead. After this exchange $D$ is still an EDS and its size in unchanged, but strictly fewer pairs of edges in $D$ share an endpoint. We repeat the operation until $D$ is independent; by construction $e \in D$ and $|D| \leq k$.

For the “all” part of the claim, we follow a similar reasoning. Again, one direction is trivial: if $e$ belongs to all size-$k$ EDS it belongs to all size-$k$ IEDS as well. For the converse implication, let $D$ be a size-$k$ EDS that does not contain $e$. We will build an IEDS at least as small that does not contain $e$ either. We turn $D$ into a minimal EDS that does not contain $e$. This is more straightforward, as the greedy algorithm only remove edges. Then, if $D$ is not an IEDS, there exist two edges $e_1 = (u, v)$ and $e_2 = (v, w)$ sharing an endpoint. $D$ is minimal, so $e_1$ (resp. $e_2$) is incident to an edge $e_1'$ (resp. $e_2'$) that is only dominated by $e_1$ (resp. $e_2$). This implies that $e_1' \neq e_2'$ and at least one of them, say $e_1'$, is not $e$. We remove $e_1$ from $D$ and add $e_1'$ instead. We repeat the operation until $D$ is an IEDS; by construction $e \notin D$ and $|D| \leq k$. $\square$

**Theorem 2** Edge Dominating Set has a direct $\infty$-lossless kernelization leaving at most $\max(6k, \frac{1}{2}k^2 + \frac{1}{2}k)$ vertices in the kernel.

**Proof:** We show that the sets of all edges that belong to all/no size-$k$ EDS of $G$ can be deduced in polynomial time from the knowledge of such sets for $G'$, as defined in Proposition 3.

First, observe that no deleted edge $(u, v)$ with $v \in A'$ may belong to all size-$k$ EDS of $G$: otherwise by Lemma 3 it would belong to all size-$k$ IEDS of $G$ as well, and because every neighbour of $u$ is in the vertex set of all size-$k$ IEDS of $G$ we can replace $(u, v)$ by any other edge incident to $v$ in an IEDS to obtain an EDS of same size that does not contain $(u, v)$. Then, by another application of Lemma 3 there exists an IEDS of $G$ that does not contain $(u, v)$, a contradiction.

Now, we prove that $(u, v)$ with $v \in A'$ belongs to no-size-$k$ EDS of $G$ if and only if $(v, d_v)$ belongs to no-size-$k$ EDS of $G'$ (equivalently, one belongs to some EDS in $G$ if and only if the other does the same in $G'$). Suppose that a size-$k$ EDS of $G$ that contains $(u, v)$ exists. By Lemma 3, an IEDS $D$ of $G$ with the same property exists. We replace each edge $(x, y)$ in $D$ with exactly one endpoint $y$ in $A'$ with $(y, d_y)$. By the property of $A'$, after this procedure $D$ becomes a size-$k$ IEDS (and thus an EDS) of $G'$ that contains $(v, d_v)$. Conversely, given an EDS $D'$ of $G'$ we can obtain an EDS of $G$ of same size by replacing any edge $(x, y) \in D'$ by any edge of $G$ incident to $x$; in particular we can replace $(v, d_v)$ with $(u, v)$.

Incidentally, it follows from the above method of converting an EDS of $G'$ into an EDS of $G$ (and vice-versa) that every edge both in $G$ and $G'$ belongs to all (resp. no) EDS of $G$ if and only if it belongs to all (resp. no) EDS of $G'$. Combined with the bound on the size of $G'$ given by Proposition 3, it follows that $(G', k)$ is a direct $\infty$-lossless kernel of $(G, k)$.$\square$

### 6 Conclusion

Observing that standard kernelization does not quite meet the needs of constraint propagation problems, we introduced loss-less kernels as a mean to reduce the whole propagation process to a subproblem whose size is only a function of the parameter. We refined this idea for cost-minimization constraints and defined variants which perform increasingly better propagation as the constraint becomes tighter, i.e. the cost of every tuple is at most a small constant $z$ from the optimum.

We investigated this new perspective of kernelization on the case study of VERTEX COVER. We showed that crown kernelization techniques can be adapted to compute $z$-loss-less kernels with at most $(z + 2)k$ vertices for every fixed $z$, matching the results of [Chlebík and Chlebíková, 2008] for $z = 0$. Finally, we showed with the example of EDGE DOMINATING SET that the idea of loss-less kernelization applies to others problems as well. Our results demonstrate the theoretical value of loss-less kernels and complement well the practical study of [Carbonnel and Hebrard, 2016].
References


