Envelope Quadratizations: A method that can quadratize with a negative number of auxiliary variables

Nike Dattani[1,*] and Andreas Soteriou[2,†]

[1]*University of Waterloo, Department of Electrical and Computer Engineering, Canada*
[2]*Surrey University, Department of Electrical and Electronic Engineering, UK*

We present a quadratization method that we have found to have advantages over all previously known quadratization methods for every case tested so far, in terms of requiring fewer auxiliary variables, less connectivity between the variables, and/or smaller coefficients. The quadratizations presented in this work, are also obtainable with orders of magnitude less CPU time than current state-of-the-art methods. The idea is that many *very* simple quadratic functions might *almost* reproduce a degree-k function perfectly, and each of these *almost* perfect quadratic functions can compensate for each others imperfections, hence leading to a perfect quadratization when all such quadratic functions are considered collectively. It is the *envelope* of multiple quadratic functions which reproduces the original degree-k function, so we call our method *envelope quadratization*. We are able to quadratize positive degree-k monomials with quadratic functions that have *much smaller* coefficients than in [Boros *et al.* (2018) "Compact quadratizations of symmetric pseudo-Boolean functions: sub-linear bounds on the number of auxiliary variables"], while also having *far* less connectivity between the variables of the quadratics, and requiring fewer auxiliary variables: The state-of-the-art was $\lceil \log_2 k/2 \rceil$ auxiliary variables and we in effect able to achieve as low as $\lceil \log_2 k/k \rceil$. We will also show examples in which the envelope of quadratic functions, each with *fewer* total variables than the original degree-k function, exactly reproduces the degree-k function, meaning that we effectively have a 'negative' number of auxiliary variables. We will also show an application of our method to the types of degree-k functions arising in a computer vision algorithm for image deblurring.

I. INTRODUCTION

Any real-valued function of binary variables can be turned into a quadratic one that maintains all the desired properties of the original degree-k function [1]. We call this a 'quadratization' of the degree-k function. The first quadratization technique of which we are aware, is Rosenberg's substitution method from 1975 [2], which for quadratizing a single positive monomial of degree-k, would need $k-2$ auxiliary variables (meaning that the quadratic function obtained after the quadratization, would involve $k-2$ more auxiliary variables than the original degree-k monomial). This was reduced by roughly one half in 2011 by Ishikawa, who introduced a quadratization for positive degree-k monomials, which needs only $\frac{k-1}{2}$ auxiliary variables (rounded down if k is even) [3]. In 2018 several new quadratizations were introduced for positive degree-k monomials: one required only $k/4$ auxiliary variables, one required only $\log_2 k$ (both of these were presented in [4]), and while it might be surprising that an improvement can be made over $\log_2 k$, a subsequent 2018 paper finally showed a quadratization which required only $\log_2(k/2)$ [5] (all three of these numbers are rounded *up* to the nearest integer). While it was astonishing that any positive monomial can be quadratized with only $\log_2(k/2)$ auxiliary variables, the resulting quadratic function would have very large coefficients and all possible quadratic terms over all original and auxiliary variables; and since large coefficients and a large number of quadratic terms, are two of the factors which most severely increase the cost when optimizing a quadratic discrete function, one may hesitate to use such quadratizations in practice.

Quadratization is also possible without the addition of *any* auxiliary variables [1, 6–9], and while these techniques should always be attempted, they in many cases might only reproduce the ground state of the degree-k function rather than the whole spectrum [1], and in many cases they might also be very expensive. One of these zero-auxiliary-variable methods that does reproduce the full spectrum, the split-reduc method [8], can be considered a special case of our *envelope quadratization* method.

In this work we introduce the concept of envelope quadratizations, and we will compare their efficacy to previously known methods. For the purpose of comparing the efficacy of different quadratization methods, throughout this paper we will assume that the final η-variable quadratic discrete optimization problems being solved are still rather difficult, with the cost of solving them being roughly $c2^\eta$ for some constant c that is very close to 1, and $c=1$ representing the cost of solving the quadratic problem by testing all possible inputs. We will introduce our concept of an *added cost factor* that arises from quadratizing a function (whether with envelope quadratizations or any of the previously known quadratization methods) to capture much of what we desire when comparing the efficacy of quadratization methods under the assumption that the cost of solving the problem after quadratization is still close to 2^η.

Envelope quadratizations turn out to be capable of 'perfectly' quadratizing functions with a smaller *added cost factor* than any other known method (by 'perfectly' we mean, reproducing the whole spectrum of possible input/output pairs for the degree-k function, since there exist methods such as [1, 6, 7] that might be more

efficient when the goal is only to reproduce the ground state).

Even when the goal is only to preserve the ground state, envelope quadratizations are often the most efficient way to quadratize a function; and in other cases they should likely always be considered in combination with other methods in order to achieve the most efficient quadratization possible. We have not yet found a single case in which the envelope of more than one quadratic function does not introduce a *strictly* smaller *added cost factor* than *all* of the other known methods for perfect quadratization.

II. A QUICK EXAMPLE

Let us review the idea behind most known quadratizations. Consider the equality:

$$b_1 b_2 b_3 = \min_{b_a} \left(b_1 b_a + b_2 b_3 - 2 b_2 b_a - 2 b_3 b_a + 3 b_a \right), \quad (1)$$

where all b_i are either 0 or 1 and the subscript a in b_a indicates that b_a is an 'auxiliary' variable because it does not appear in the original cubic function on the left side. For clarity we explicitly show why Eq. 1 is true by constructing the truth tables for the left and right sides, starting with the left side:

b_1	b_2	b_3	$b_1 b_2 b_3$
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

For the right side we have:

b_1	b_2	b_3	b_a	$b_1 b_a + b_2 b_3 - 2 b_2 b_a - 2 b_3 b_a + 3 b_a$	$\min_{b_a}$
0	0	0	**0**	0	**0**
0	0	0	**1**	3	
0	0	1	**0**	0	**0**
0	0	1	**1**	1	
0	1	0	**0**	0	**0**
0	1	0	**1**	1	
0	1	1	**0**	1	**0**
0	1	1	**1**	0	
1	0	0	**0**	0	**0**
1	0	0	**1**	4	
1	0	1	**0**	0	**0**
1	0	1	**1**	2	
1	1	0	**0**	0	**0**
1	1	0	**1**	2	
1	1	1	**0**	1	**1**
1	1	1	**1**	1	

It is *not* possible to construct a single quadratic function without any auxiliary variables, that reproduces the spectrum as seen in the tables above. However, minimizing over the quadratic function $b_1 b_2$ and the linear function b_3, perfectly yields the full spectrum of $b_1 b_2 b_3$ without introducing any auxiliary variables:

b_1	b_2	b_3	$b_1 b_2$	b_3	$\min(b_1 b_2, b_3)$
0	0	0	0	0	**0**
0	0	1	0	1	**0**
0	1	0	0	0	**0**
0	1	1	0	1	**0**
1	0	0	0	0	**0**
1	0	1	0	1	**0**
1	1	0	1	0	**0**
1	1	1	1	1	**1**

We call the set $\{b_1 b_2, b_3\}$ an *envelope quadratization* for $b_1 b_2 b_3$. At each point (b_1, b_2, b_3), the minimum (tangent) over all functions is equal to the degree-k function $b_1 b_2 b_3$.

III. GENERALIZATIONS OF THE QUICK EXAMPLE

A. Any product of variables

The simple example from the previous section, of a quadratic envelope reproducing $b_1 b_2 b_3$, can be generalized:

$$b_1 b_2 b_3 \dots b_k = \min \left(b_1 b_2 \dots b_{k_1}, b_{k_1+1} b_{k_1+2} \dots b_{k_2}, \right. \quad (2)$$
$$\left. b_{k_2+1} b_{k_2+2} \dots b_{k_3}, \dots, b_{k_p+1} b_{k_p+2} \dots b_k \right),$$

where $k_1, k_2, \dots, k_p$ can be chosen arbitrarily between 1 and k, and if $k_1 = 2$ and all other $k_i = k_{i-1} + 2$ then it is an envelope quadratization. A specific case of Eq. 2 is:

$$b_1 b_2 b_3 \dots b_k = \min \left(b_1, b_2, b_3, \dots, b_k \right), \quad (3)$$

which is actually an envelope *linearization* (and therefore also an envelope quadratization since all linear functions are examples of quadratic functions with the coefficient of the quadratic term equal to zero).

B. Any product of polynomials

Even more generally than what we presented in the preceding section, we have:

$$f_1 f_2 \dots f_\kappa = \min\left(f_1, f_2, \dots, f_\kappa\right), \text{ if} \quad (4)$$
$$f_i\left(b_{k_i}, b_{k_i+1}, \dots, b_{k_{i+1}-1}\right) \geq 0, \quad (5)$$

which means that any product of polynomials is equivalent to the right-hand side of Eq. 4 if each polynomial $f_i \geq 0$ for any input. Eq. 2, and therefore all envelope reductions presented so far, are special cases of Eq. 4 in which all f_i are monomials. If all f_i are quadratic then we have a quadratization.

Eq. 4 can be further generalized to any product of polynomials whether or not they satisfy $f_i \geq 0$, for example:

$$f_1 f_2 \dots f_\kappa = \min\left(f_1 f_2 \dots f_{\kappa-1} \max f_\kappa, \right. \quad (6)$$
$$\left. f_\kappa - \min f_\kappa + f_1 f_2 \dots f_{\kappa-1}\right), \text{ if}$$

$$f_{i<\kappa}\left(b_{k_i}, b_{k_i+1}, \dots, b_{k_{i+1}-1}\right) \geq 0, \min f_\kappa < 0, \quad (7)$$

which means that if one factor (call it f_κ) has a minimum value less than 0, and all other factors satisfy $f_{i<\kappa} \geq 0$ for any input, Eq. 6 is satisfied. Again, when all f_i are quadratic, we have a quadratization, and in Section VII B we will use Eq. 6 to quadratize a trinomial appearing in a computer vision application.

IV. USAGE

A. Quadratizing part of a function using envelope quadratizations

If we wished to minimize the function of binary variables:

$$3b_1b_2b_3 - 2b_1b_2 - 2b_1b_3 + 4b_2b_4 - 3b_4 - 3b_2b_5 \quad (8)$$

using the wealth of techniques we have available for minimizing *quadratic* functions of binary variables (such as the classical algorithm 'QPBO' and extensions of it [10], or quantum annealing using thousands of qubits [11] connected by graphs as complicated as Pegasus [12, 13]), we can quadratize the cubic term with the envelope quadratization $b_1b_2b_3 = \min(b_1b_2, b_3)$, and we get two objective functions to consider:

$$3b_1b_2 - 2b_1b_2 - 2b_1b_3 + 4b_2b_4 - 3b_4 - 2b_2b_5 \quad (9)$$
$$3b_3 - 2b_1b_2 - 2b_1b_3 + 4b_2b_4 - 3b_4 - 2b_2b_5. \quad (10)$$

Running the optimization procedure (for example QPBO or quantum annealing) on both functions would (if it succeeds in finding the true ground states) result in the following minima:

$$\text{Eq.9: } -5 \text{ at } b_1,b_2,b_3,b_4,b_5 = 10110 \text{ \& } 10111 \quad (11)$$
$$\text{Eq.10: } -4 \text{ at } b_1,b_2,b_3,b_4,b_5 = 11001. \quad (12)$$

The function with the lowest minima has a minimum value of -5 at both 10110 and 10111, so these are the global minima of the original degree-k function (this can be verified by simply optimizing the original quartic function, and indeed in Appendix A we have evaluated the quartic functions for all possible inputs, but for more complicated functions it is not possible to evaluate the degree-k function for all possible inputs, and the algorithms to optimize a super-quadratic function can be much more costly than optimizing several quadratic functions, or one quadratic function with auxiliary variables, or even several quadratic functions that each contain auxiliary variables).

We note that it is not only the global minima (or ground states), that are reproduced by our envelope quadratization. The full spectrum is actually reproduced, meaning that for each of the $2^5 = 32$ combinations for: $(b_1, b_2, b_3, b_4, b_5)$, we have $f = \min_{f_1,f_2}(f_1, f_2)$, where f is the function in Eq. 8 and f_1, f_2 are the functions in Eqs. 9-10 respectively. The 32-row table showing that $f = \min_{f_1,f_2}(f_1, f_2)$ for all 32 possible configurations $(b_1, b_2, b_3, b_4, b_5)$ is given in Appendix A.

B. Quadratizing multiple parts of a function using envelope quadratizations

To show that the method of envelope quadratizations can be applied to multiple parts of a degree-k objective function, consider the example:

$$2b_1b_2b_3b_4 - 3b_2b_4 + 3b_1b_4b_6 - 2b_3b_6 - 3b_1b_5 + b_5 + b_6 - b_3. \quad (13)$$

Here we quadratize $b_1b_2b_3b_4$ with $\{b_1b_2, b_3b_4\}$ and $b_1b_4b_6$ with $\{b_1b_4, b_6\}$. This gives us four possible objective functions to consider:

$$2b_1b_2 - 3b_2b_4 + 3b_1b_4 - 2b_3b_6 - 3b_1b_5 + b_5 + b_6 - b_3 \quad (14)$$
$$2b_1b_2 - 3b_2b_4 + 3b_6 - 2b_3b_6 - 3b_1b_5 + b_5 + b_6 - b_3 \quad (15)$$
$$2b_3b_4 - 3b_2b_4 + 3b_1b_4 - 2b_3b_6 - 3b_1b_5 + b_5 + b_6 - b_3 \quad (16)$$
$$2b_3b_4 - 3b_2b_4 + 3b_6 - 2b_3b_6 - 3b_1b_5 + b_5 + b_6 - b_3 \quad (17)$$

Running the optimization procedure on all four functions would (if it succeeds in finding the true ground states) result in the following minima:

Eq.14 : -5 at $b_1, b_2, b_3, b_4, b_5, b_6 = 011101$ (18)

Eq.15 : -4 at $b_1, b_2, b_3, b_4, b_5, b_6 = 011100$ & 111110 (19)

Eq.16 : -4 at $b_1, b_2, b_3, b_4, b_5, b_6 = 101011$ & 111011 (20)

Eq.17 : -5 at $b_1, b_2, b_3, b_4, b_5, b_6 = 111011$. (21)

The functions which have the lowest minima have minimum values of -5 at 011101 and 111011 respectively, so these are the global minima of the original degree-k function, and again this can be verified by simply optimizing the original quartic function or evaluating it for all possible inputs (which we have done in Appendix B).

Again we note that it is not only the global minima (or ground states), that are reproduced by our envelope quadratization. The full spectrum is actually reproduced, meaning that for each of the $2^6 = 64$ combinations for: $(b_1, b_2, b_3, b_4, b_5, b_6)$, we have $f = \min_{f_1, f_2, f_3, f_4} (f_1, f_2, f_3, f_4)$, where f is the function in Eq. 13 and f_1, f_2, f_3, f_4 are the functions in Eqs. 14-17 respectively. The 64-row table showing that $f = \min_{f_1, f_2, f_3, f_4} (f_1, f_2, f_3, f_4)$ for all 64 possible configurations $(b_1, b_2, b_3, b_4, b_5, b_6)$ is given in Appendix B.

V. WORST-CASE COST ANALYSIS

Unless there is some property of the function making it easier, optimizing a function of n binary variables by brute force requires 2^n queries, so if m binary variables are added in the quadratization process, the cost becomes $2^n 2^m$ (meaning the cost is increased by a factor of 2^m). While it is true that better optimization algorithms exist than doing brute force queries (hence the reason for quadratizing the original function at the expense of a possible 2^m increase in cost!), all general-purpose methods for optimizing such functions that are at least quadratic, still have a cost of $\mathcal{O}(2^\eta)$ for η total binary variables (this includes quantum annealing, where the exponential scaling up to ~1000 variables can be seen in the figures of [14] and [15]), so claiming that a factor of 2^m increase in cost is not unfair for analyzing costs of quadratizations that add m auxiliary variables (especially in the absence of any better measure for the added cost to the optimization in the situation described) might not be strongly disputed, but there is still the issue of the constant hidden under the $\mathcal{O}$. If the cost of minimizing a quadratic function of η binary variables is $c2^\eta$ for some positive constant c, and 2^η being the cost of evaluating the function for all 2^η possible inputs, and if the function is a comparatively difficult one to optimize (two examples of ways it can be difficult is if it has very many quadratic terms, or coefficients spanning several orders of magnitude) the constant c can be very close to 1. We are going to assume throughout this paper that our final quadratic functions are difficult to optimize, meaning that c is very close to 1.

Having established our notion of how much adding m auxiliary variables increases the cost of optimizing a quadratic function of binary variables, we will also mention that having to do r optimizations instead of 1, can be considered to increase the cost of the overall process by r times the cost of doing one optimization.

So if a quadratization splits the original optimization problem into r cases (or 'runs' of the optimization process) that each need to be optimized in order to find the global optimum, and each of these runs involves at most m auxiliary variables, then the following is an upper bound on the *added cost factor* C due to quadratizing:

$$C = r2^m. \tag{22}$$

In practice, many runs usually require fewer auxiliary variables than others (or no auxiliary variables at all), so in this sense Eq. 22 over-estimates the cost for multi-run quadratizations (envelope quadratizations) where $r > 1$; but since we are introducing a new method and comparing it to the state-of-the-art existing methods, it is okay for our notion of added cost factor to be overly harsh to the method we are introducing. Likewise, we discuss below the fact that envelope quadratizations may result in the final quadratic problem having *fewer* total variables than the original degree-k function, meaning that m is effectively negative, which would be very favorable in terms of evaluating C. However, to show that envelope quadratizations have advantages over all previously known quadratization methods according to Eq. 22, we never even need to make use of the fact that they can be considered to involve a negative value of m, even though a negative value of m will always contribute favorably to the added cost factor C. For envelope quadratizations where m is effectively negative, we simply treat m to be zero when calculating C, which is less favorable to our method when comparing it to other methods, but we are content with this because despite this, and the fact mentioned earlier in this paragraph which is also less favorable to our method, we still do not find any cases where our envelope quadratizations have an added cost factor greater than the previous state-of-the-art quadratization method!

We also mentioned that better algorithms exist for optimizing a quadratic function of binary variables than evaluating the function for all possible inputs at 2^η cost, so the 2^m factor in Eq. 22 might unfairly penalize quadratizations involving a large number m of auxiliary variables: This is why all analysis in this paper assumes that the final quadratic function is hard enough to optimize that 2^m is a fair approximation of

the true added cost of adding m auxiliary variables.

Finally we note that the split-reduc method of [8] is an example of an envelope quadratization because it can be written in the form $f = \min(f_1, f_2 \ldots, f_r)$ where the r quadratic functions f_i are the cases resulting from the split-reduc method and f is the original degree-k function. It was not written in this way when the split-reduc method was first introduced in [8]. All functions f_i coming from the split-reduc method have fewer total variables than f, which means that the m in Eq. 22 would be effectively negative (as is often also the case in the envelope quadratizations shown in the examples below, but it is much more extreme in the special case of a split-reduc envelope quadratization). This means that the fact that we are approximating the true cost of $\mathcal{O}(2^\eta) = c2^\eta$ to be exactly equal to 2^η, will give an unfair advantage to split-reduc when using Eq. 22 to compare quadratization methods. In this paper we show that envelope quadratizations have advantages over all other quadratization methods without ever mentioning split-reduc in the analysis, so this unfair advantage to split-reduc does not indicate any problem with using Eq. 22 for the purpose of this paper. However it is interesting to consider how the split-reduc envelope quadratization compares to other envelope quadratizations such as the other ones introduced in this paper. In many cases (all of them?), the envelope quadratization example presented in this work, and the best possible application of split-reduc, both have the same value of C according to Eq. 22, but split-reduc always has a much larger value of r and a much lower value of m. We note that we have made the approximation that $2^\eta \approx c2^\eta$ (and since $2^\eta \equiv 2^n 2^m$, this means the 2^m part of Eq. 22 is the part that is not exact), but the dependence of r in Eq. 22 is much more accurate: This means that Eq. 22 penalizes quadratization methods quite fairly in terms of r and *when the two methods being compared have the same value for* C, the value of c can be important in assessing which method leads to a quadratization that is easier to solve. The specific value for c will depend on the properties of the quadratic function being optimized (such as the number of quadratic terms and the range of coefficients), which can be very different for different quadratization methods. A thorough comparison between split-reduc and other types of envelope quadratizations such as the ones presented in this work, will need to await further work, but what we can say at present is that split-reduc may be at a disadvantage since it tends to require larger values of r in lieu of values of m that are more negative, so in $C = r2^m$ it is penalized fairly for the large value of r and may have an unfair advantage from the factor 2^m, so when two quadratization methods have the same value for C, split-reduc seems to be slightly less favorable.

VI. CLAIMS OF ADVANTAGE OVER PREVIOUSLY KNOWN METHODS

A. Positive degree-k monomial

Envelope quadratizations are able to *strictly* outperform the previous state-of-the-art quadratization method for quadratizing a single positive degree-k monomial, in terms of the *added cost factor* defined in Section V. In 2018, Boros *et al.* published a new method that requires only $\log_2(k/2)$ auxiliary variables (rounded up) [4, 5], and in the handbook of quadratizations [1] this method was called **PTR-BCR-4**.

For the positive degree-k monomial we compare the added cost of envelope quadratizations to the added cost of PTR-BCR-4. PTR-BCR-4 adds $m = \lceil \log_2(k/2) \rceil$ auxiliary variables, which means that the added cost C of optimization is in the interval $[k/2, k-1]$ with $C = k/2$ if k is a power of 2 and $C = k-1$ if k is 1 more than the next power of 2, and in between otherwise (see Appendix C).

Envelope quadratizations are for all k, capable of providing quadratizations for the positive degree-k monomial with an added cost factor of $C = r = \lceil k/2 \rceil$ (using Eq. 2 we can factor the monomial into $k/2$ quadratic monomials if k is even, and one extra linear monomial if k is odd). If k is even, $C = r = k/2$ for the described envelope quadratization, which PTR-BCR-4 matches *only if k is a power of 2*. If k is odd, then $C = r = \lceil k/2 \rceil = \frac{k+1}{2}$ for the envelope quadratization, and PTR-BCR-4 can match this *only if k is one less than a power of 2* (see Appendix C). In these two unique cases, in which our added cost factor for the envelope quadratization and PTR-BCR-4 are the same, we note that with the described envelope quadratization the quadratic functions obtained are just monomials with no change to the coefficient α of the original degree-k monomial being quadratized; whereas with PTR-BCR-4 the quadratic function obtained has all possible quadratic terms for $k + \lceil \log_2 k/2 \rceil$ variables, and the coefficients range (in the very best possible case) from $-\frac{\alpha k}{4}$ to $\frac{\alpha k^2}{8}$ (see the PTR-BCR-4 page of [1]). The enormous number of quadratic terms and the enormous range of coefficients in the quadratic function generated by PTR-BCR-4 make envelope quadratizations seem like a superior choice even in these cases where the added cost factor estimated by Eq. 22 is the same.

For all other values of k (anything not a power of 2 or one less than a power of 2, which constitutes the vast majority of the possible values of k), the added cost factor C is strictly smaller than for PTR-BCR-4 by up to a factor of $k - 1/\frac{k+1}{2} = \frac{2(k-1)}{(k+1)}$ (which is realized when k is the odd number that is one more than a power of 2). As k gets large, the numerator $(k-1)$ and denominator $(k+1)$ become more and more similar, and the envelope quadratizations tend towards adding a cost

factor of roughly half of what PTR-BCR-4 adds. We can think of envelope quadratizations as adding a cost factor that is equivalent to effectively giving, instead of $r > 1$ quadratic functions with $m = 0$ auxiliary variables, only $\tilde{r} = 1$ effective quadratic function with:

$$\lceil \log_2 k/4 \rceil < \tilde{m} \leq \lfloor \log_2 k/2 \rfloor \tag{23}$$

auxiliary variables. For example, when $k = 7$, which is 1 less than a power of 2, we have $(r, m) = (4, 0)$ and therefore $C = 4$, but $(\tilde{r}, \tilde{m}) = (1, \lfloor \log_2 7/2 \rfloor)$ also leads to $C = 4$; and when $k = 9$ which is 1 more than a power of 2, we have $(r, m) = (5, 0)$ leading to an added cost factor of $C = 5$ which is only slightly larger than $(r, m) = (1, \lceil \log_2 9/4 \rceil)$ for which C would be 4. The strict inequality $\lceil \log_2 k/4 \rceil < \tilde{m}$ can get closer and closer to being an equality as k gets large (for $k = 33$ we have $C = 2^{\tilde{m}} = 17$ and $2^{\lceil \log_2 k/4 \rceil} = 16$.

B. General functions of binary variables with real-valued coefficients

We have shown that when quadratizing a positive degree-k monomial, envelope quadratizations are now the state-of-the-art in terms of our *added cost factor* and the number of quadratic terms and sizes of coefficients in the quadratic functions. What about when quadratizing not only a single monomial but multiple terms in a function? PTR-BCR-4 is no longer necessarily the best quadratization. For example, quadratizing $b_1b_2b_3b_4b_5 + b_2b_3b_4b_5b_6$ would require 4 auxiliary variables (2 for each term), but the pairwise covers method of [1, 16] would only need two auxiliary variables: b_{a_1} being a substitution for $b_2b_3b_4b_5$ and b_{a_2} and being an auxiliary variable to quadratize $b_2b_3b_4b_5$. In this case we can make a 'trivial' argument that envelope quadratizations are at least as good as the previous state-of-the-art method, because the previous state-of-the-art method is just an example of an envelope quadratization in which $r = 1$. However, we have also found that for every case we have examined so far, we are also able to obtain an envelope quadratization with $r > 1$ that is strictly better in terms of added cost factor or number of quadratic terms than the previous state-of-the-art quadratization. There are some cases in which the added cost factor of Eq. 22 can be matched by previously known methods, but in those cases envelope quadratizations still have fewer quadratic terms and smaller coefficients (for example, see Section VII A below).

VII. MORE EXAMPLES AND POTENTIAL APPLICATIONS

A. Positive degree-8 monomial

It has been known for some time that no positive monomial with degree greater than 4 can be quadratized by a single function with fewer than 2 auxiliary variables (for example, see the last line of Pg. 182 in [17]). We therefore found it quite fascinating that we could find an envelope quadratization for the positive degree-8 monomial which involves at most 1 auxiliary variable in the two pieces:

$$b_1b_2b_3b_4b_5b_6b_7b_8 : \tag{24}$$
$$\longrightarrow 3b_a + b_1b_2 + b_1b_3 + b_1b_4 + b_2b_3 + b_2b_4 \tag{25}$$
$$+ b_3b_4 - 2b_a(b_1 + b_2 + b_3 + b_4) \tag{26}$$
$$\longrightarrow 3b_a + b_5b_6 + b_5b_7 + b_5b_8 + b_6b_7 + b_6b_8 \tag{27}$$
$$+ b_7b_8 - 2b_a(b_5 + b_6 + b_7 + b_8). \tag{28}$$

Here the added cost of a factor of $2 \times 2^1 = 4$ is still the same as it would be with PTR-BCR-4 (which would need only 2 auxiliary variables and therefore an added cost of a factor of $1 \times 2^2 = 4$). However the quadratic function arising from PTR-BCR-4 contains all possible quadratic terms for the 8 original variables and the 2 auxiliary variables, which would result in a *gadget graph* [13] of K_{10}, whereas the above envelope quadratization would for both quadratic functions involve the much simpler gadget graph of K_5. Furthermore, the quadratic terms resulting from the PTR-BCR-4 quadratization would have coefficients ranging from -2 to 8, and in the above envelope quadratization, they range from -2 to 1, which means the coefficient range is more than 3 times smaller in the envelope quadratization.

B. Application to a computer vision algorithm

The image deblurring algorithm of [3, 18] involves degree-k terms with $k = d^2$ with $d \geq 2$ being the length in pixels of the square-shaped mask). In all applications we know so far, $d = 2$ and $k = 4$ (though larger masks can in theory be used, resulting in an optimization problem with terms of degree at least 9). Each pixel is represented by a binary variable which during the deblurring of the image, will attain a value of 1 if it changes and 0 if it does not.

For each 2×2 array of pixels that has no contact with an edge of the overall image, 9 degree-4 terms and 41 degree-3 terms appear in the initial function to be optimized. After applying the ELC procedure of [6], on the same image (of a stone statue) used in that paper, the 50 super-quadratic terms associated with the 2×2 patch of the pixels 169, 170, 329, 330, reduce to just the trinomial: $b_1b_2b_3b_4 + b_2b_3b_4 - b_3b_4b_5$ (our input and

output files are provided in this paper's Supplementary Material). In [6] (published in 2014), these three super-quadratic terms were then quadratized with one auxiliary variable for each super-quadratic term (three auxiliary variables). Since 2017, new methods were introduced which could quadratize the same three terms with only two auxiliary variables: In 2017 the "pairwise covers" technique was introduced in [16] which would only require one auxiliary variable to cover $b_2b_3b_4$ and one to cover b_3b_4, and in 2019 it was proven in [19] that all 4-variable functions can be quadratized with only one auxiliary variable, so the first two terms can be quadratized with the same auxiliary variable, then only one auxiliary variable would be needed for the remaining cubic term.

Need to try ELCs and Deduc-Reduc and Grubner basis and mention Split-Reduc before saying that we have SOTA.

However it is easy to construct an envelope quadratization that effectively has the cost of adding only one auxiliary variable:

$$\begin{aligned} & b_1b_2b_3b_4 + b_2b_3b_4 - b_3b_4b_5 && (29)\\ = \; & b_3b_4\,(b_1b_2 + b_2 - b_5) && (30)\\ = \; & \min\left(2b_3b_4, b_1b_2 + b_2 - b_5 + 1 - b_3b_4\right). && (31)\end{aligned}$$

In Eq. 30 we have simply factored out b_3b_4 since it was in all three terms, and in Eq. 31 we have applied Eq. 6 which results in only 2 runs and 0 auxiliary variables.

C. Envelope quadratization with a reduction in added cost factor of 16 compared to the previous state-of-the-art

We discovered the concept of envelope quadratizations on 3 September 2019, so the number of functions for which we have searched for envelope quadratizations so far has been limited by the amount of time elapsed since then. However the best reduction in added cost factor (in the spirit of Eq. 22) we have found *so far* with envelope quadratizations compared to the previous state-of-the-art, was for the function:

$$b_1b_2b_3b_4b_5b_6b_7b_8 + b_2b_3b_4b_5b_6b_7b_8b_9 + b_3b_4b_5b_6b_7b_8b_9b_{10}. \quad (32)$$

This can be quadratized with $m = 6$ auxiliary variables by quadratizing each of the three terms by PTR-BCR-4. We have not been able to find a way to quadratize this function with $C < 64$ using techniques known to us before writing this paper. Below we show that with envelope quadratizations we can reduce the added cost factor down to $C = 4$ which is a factor of 16 times smaller than 64. We start by factoring Eq. 32 into the product of a quadratic function and a degree-6 function, then we apply Eq. 4:

$$\begin{aligned} & (b_1b_2 + b_2b_9 + b_9b_{10})b_3b_4b_5b_6b_7b_8 && (33)\\ = \; & \min\left(b_1b_2 + b_2b_9 + b_9b_{10}, 3b_3b_4, 3b_5b_6, 3b_7b_8\right). && (34)\end{aligned}$$

Since Eq. 34 involves 0 auxiliary variables and only 4 runs, we have an added cost factor of $C = 4$.

VIII. DISCUSSION

A. Speed of finding multi-run envelope quadratizations versus single-run quadratizations

Perhaps the only published algorithm for finding quadratizations of arbitrary functions, is the method which involves enumerating all vertices and rays of a convex polyhedron, as mentioned on Pg. 152 in section 3.8 of [17]; and Pg. 153 in the same section estimates that it would take at least 3 full months of CPU time to do this for a quadratization into a quadratic function with 7 total variables (for example 6 original variables and 1 auxiliary variable). This is in contrast to the ~1 hour of CPU time it took them to do this for 6 total variables (for example 5 original variables and 1 auxiliary variable). Therefore, finding quadratizations with 1-auxiliary variable for a 7-variable function (8 total variables) could possibly take several years, and finding quadratizations with 1-auxiliary variable for an 8-variable function could possibly take several centuries. However Section VII A shows a 1-auxiliary-variable envelope quadratization of an 8-variable function, which was simply done by applying formulas, which does not take any CPU time. Compared to the envelope quadratization generated instantly, the centuries of estimated CPU time for vertex enumeration would aim to find a quadratization twice as efficient (by the standard of Eq. 22, because r would be 1 instead of 2 and m would be 1 in both cases), but we already know that the centuries of estimated CPU time would not find any $(r, m) = (1, 1)$ perfect quadratization because it is already known that positive monomials of degree greater than 4 cannot be perfectly quadratized with $(r, m) = (1, 1)$. Nevertheless, the message we wish to convey is that if an $(r, m) = (1, 1)$ perfect quadratization were to exist for some 8-variable function, it could take many centuries to find it using the only other known algorithm for finding new perfect quadratizations of arbitrary functions, and we were able to find instantly an envelope quadratization with an equivalent added cost factor, and based on comparisons of all envelope quadratizations found so far, to previous state-of-the-art quadratizations, it is very likely that the coefficients and number of quadratic terms in the envelope quadratization would be signif-

icantly lower than whatever were to be found in these centuries of CPU time.

B. The quadratic functions in an envelope quadratization can (and usually do) have a 'negative' number of auxiliary variables compared to the original degree-k function

Observe that in every(?) envelope quadratization example we have given, the many/most of? (if not all) quadratic functions have fewer variables than the original degree-k function. We found this to be very astonishing, because perfect quadratizations are usually thought to require the introduction of new variables (and a lot of new variables!). In the case of Eq. 32, we have a 10-variable, degree-8 function of three terms, that is transformed into 4 quadratic problems that each have *at most* 4 variables (and in 3 of the 4 cases, only have 2 variables). In fact solving each of the 4 quadratic problems by brute force (trying all possible combinations of 0 and 1 for the input variables and finding the configuration that gives the lowest output), surprisingly would amount to fewer such brute force attempts (by about 1000) than needed for the original 10-variable problem, which we find interesting from a 'query complexity' or 'computational complexity' perspective, though the original function has a lot of symmetry and in the envelope quadratization case there is an additional (but very easy) minimization to be done over the 4 quadratic functions.

Even if an envelope quadratization aims only to quadratize some fraction of the complete set of super-quadratic terms in a degree-k function, and the variables that do not appear in one of the quadratic functions (such as b_1, b_2 and b_5 that do not appear in the function $2b_3b_4$ appearing in Eq. 31) do appear in other terms of the overall degree-k function, the number of auxiliary variables can still be considered to be effectively negative in some cases. When multiple options are available for the construction of the envelope quadratization, it may be preferred to choose an option which leads to a reduction in the total number of variables in the final optimization problem, if such an option exists among the multiple options available.

C. It is not yet the end of our journey

We originally wanted to find 'heuristic quadratizations', which would be quadratizations that do not necessarily need to reproduce a full spectrum, or even a ground state (optimum) of the original function, but would come very close (for example by finding the next lowest-energy state above the global minimum). This is because often we do not even need the *exact* ground state of a function, and all we seek is a good approximation to it, for example the true ground state is not being sought in the above-mentioned image de-blurring algorithm: it is an algorithm in which the closer we get to the optimum solution, the better the de-blurring will be, but eventually the difference between one de-blurring and a de-blurring closer to the optimal solution will hardly be noticeable, and eventually we may even reach the limitations of the de-blurring algorithm itself, meaning that reaching a lower-energy solution will no longer even guarantee a better de-blurring. When we are not seeking the absolute ground state of a function, then it would be 'overkill' to demand that the quadratization exactly preserves the ground state, and this extra and unnecessary demand ought to be limiting us in terms of the efficiency of quadratizations available for us to find (for example we must be able to find quadratizations that do not require as many auxiliary variables, do not require such large coefficients, do not have such dense connectivity in the graph describing the quadratic terms, and do not have so many non-submodular terms, as our currently known quadratizations which do more than we need).

Envelope quadratizations not only reproduce the ground state perfectly (which is more than what we usually want), they actually reproduce the entire spectrum which contains 2^n states (when the original degree-k optimization problem has n variables) and their corresponding energies (which is *far* more than what we seek in almost all real-world applications). Therefore we have accidentally found something better than what we were seeking, but ultimately we should be able to reduce costs even more when we (successfully!) relax the requirement that all 2^n states of the original function are reproduced in the very end (we used in parentheses the adjective 'successfully' because envelope quadratizations were in fact *born* out of our pursuit to relax the requirement that all 2^n states are reproduced, and while we were very happy to discover envelope quadratizations, they turned out to satisfy the requirement that we were originally hoping to relax).

We hope that by relaxing the requirement that a quadratization of an n-variable function must reproduce all 2^n states after minimizing over all runs, we may *further* improve the cost factor that our new (and likely state-of-the-art) *envelope quadratization* concept adds to the solving of degree-k binary optimization problems when turning them into quadratic optimization problems.

IX. ACKNOWLEDGMENTS

We thank Endre Boros for suggesting the name *quadratic envelopes*. We originally used the acronym COMIG (**com**plimentary **i**mperfect **g**adgets) since our method uses multiple quadratic gadgets which are imperfect in that they do not on their own reproduce the *full* spectrum of the original degree-k function, but they are complimentary in that we can reproduce the de-

sired full spectrum when all imperfect gadgets complementing each other are combined and we minimize over them). Overall we prefer the term *envelope quadratizations* the most since it is the envelope of many quadratic functions which is our final quadratization that exactly reproduces the original degree-k function.

* ndattani@uwaterloo.ca
† as03205@surrey.ac.uk

[1] N. Dattani, *Quadratization in discrete optimization and quantum mechanics* (2019) arXiv:1901.04405.
[2] I. G. Rosenberg, Cahiers du Centre d'Etudes de Recherche Operationnelle **17**, 71 (1975).
[3] H. Ishikawa, IEEE Transactions on Pattern Analysis and Machine Intelligence **33**, 1234 (2011).
[4] E. Boros, Y. Crama, and E. Rodríguez-Heck, *Quadratizations of symmetric pseudo-Boolean functions: sub-linear bounds on the number of auxiliary variables*, Tech. Rep. (2018).
[5] E. Boros, Y. Crama, and E. Rodriguez Heck, *Compact quadratizations for pseudo-Boolean functions* (2018).
[6] H. Ishikawa, in *2014 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2014) pp. 1362–1369.
[7] R. Tanburn, E. Okada, and N. Dattani, (2015), arXiv:1508.04816.
[8] E. Okada, R. Tanburn, and N. S. Dattani, (2015), arXiv:1508.07190.
[9] R. Dridi and H. Alghassi, Scientific Reports **7**, 43048 (2017).
[10] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer, in *2007 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2007) pp. 1–8.
[11] A. D. King, J. Carrasquilla, J. Raymond, I. Ozfidan, E. Andriyash, A. Berkley, M. Reis, T. Lanting, R. Harris, F. Altomare, K. Boothby, P. I. Bunyk, C. Enderud, A. Fréchette, E. Hoskinson, N. Ladizinsky, T. Oh, G. Poulin-Lamarre, C. Rich, Y. Sato, A. Y. Smirnov, L. J. Swenson, M. H. Volkmann, J. Whittaker, J. Yao, E. Ladizinsky, M. W. Johnson, J. Hilton, and M. H. Amin, Nature **560**, 456 (2018).
[12] N. S. Dattani, S. Szalay, and N. Chancellor, Pegasus: The second connectivity graph for large-scale quantum annealing hardware (2019), arXiv:1901.07636.
[13] N. Dattani and N. Chancellor, (2019), arXiv:1901.07676.
[14] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, Physical Review X **6**, 031015 (2016).
[15] S. Mandrà, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber, Physical Review A **94**, 022337 (2016).
[16] M. Anthony, E. Boros, Y. Crama, and A. Gruber, Mathematical Programming **162**, 115 (2017).
[17] A. G. Gruber, (2015), doi:10.7282/T3PZ5BHQ.
[18] S. Roth and M. Black, in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2 (IEEE, 2005) pp. 860–867.
[19] N. Dattani and H. T. Chau, (2019), arXiv:1910.13583.

Appendix A: Proof that the full spectrum is reproduced in the envelope quadratization of Section IV A

b_1	b_2	b_3	b_4	b_5	f_1 Eq. 9	f_2 Eq. 10	$\min_{f_1 f_2}$	f Eq. 8
0	0	0	0	0	0	0	**0**	**0**
0	0	0	0	1	0	0	**0**	**0**
0	0	0	1	0	-3	-3	**-3**	**-3**
0	0	0	1	1	-3	-3	**-3**	**-3**
0	0	1	0	0	0	3	**0**	**0**
0	0	1	0	1	0	3	**0**	**0**
0	0	1	1	0	-3	0	**-3**	**-3**
0	0	1	1	1	-3	0	**-3**	**-3**
0	1	0	0	0	0	0	**0**	**0**
0	1	0	0	1	-2	-2	**-2**	**-2**
0	1	0	1	0	1	1	**1**	**1**
0	1	0	1	1	-1	-1	**-1**	**-1**
0	1	1	0	0	0	3	**0**	**0**
0	1	1	0	1	-2	1	**-2**	**-2**
0	1	1	1	0	1	4	**1**	**1**
0	1	1	1	1	-1	2	**-1**	**-1**
1	0	0	0	0	0	0	**0**	**0**
1	0	0	0	1	0	0	**0**	**0**
1	0	0	1	0	-3	-3	**-3**	**-3**
1	0	0	1	1	-3	-3	**-3**	**-3**
1	0	1	0	0	-2	1	**-2**	**-2**
1	0	1	0	1	-2	1	**-2**	**-2**
1	0	1	1	0	-5	-2	**-5**	**-5**
1	0	1	1	1	-5	-2	**-5**	**-5**
1	1	0	0	0	1	-2	**-2**	**-2**
1	1	0	0	1	-1	-4	**-4**	**-4**
1	1	0	1	0	2	-1	**-1**	**-1**
1	1	0	1	1	0	-3	**-3**	**-3**
1	1	1	0	0	-1	-1	**-1**	**-1**
1	1	1	0	1	-3	-3	**-3**	**-3**
1	1	1	1	0	0	0	**0**	**0**
1	1	1	1	1	-2	-2	**-2**	**-2**

Appendix B: Proof that the full spectrum is reproduced in the envelope quadratization of Section IV B

b_1	b_2	b_3	b_4	b_5	b_6	f_1 Eq. 14	f_2 Eq. 15	f_3 Eq. 16	f_4 Eq. 17	$\min_{f_1 f_2 f_3 f_4}$	f Eq. 13
0	0	0	0	0	0	0	0	0	0	**0**	**0**
0	0	0	0	0	1	1	4	1	4	**1**	**1**
0	0	0	0	1	0	1	1	1	1	**1**	**1**
0	0	0	0	1	1	2	5	2	5	**2**	**2**
0	0	0	1	0	0	0	0	0	0	**0**	**0**
0	0	0	1	0	1	1	4	1	4	**1**	**1**
0	0	0	1	1	0	1	1	1	1	**1**	**1**
0	0	0	1	1	1	2	5	2	5	**2**	**2**
0	0	1	0	0	0	-1	-1	-1	-1	**-1**	**-1**
0	0	1	0	0	1	-2	1	-2	1	**-2**	**-2**
0	0	1	0	1	0	0	0	0	0	**0**	**0**
0	0	1	0	1	1	-1	2	-1	2	**-1**	**-1**
0	0	1	1	0	0	-1	-1	1	1	**-1**	**-1**
0	0	1	1	0	1	-2	1	0	3	**-2**	**-2**
0	0	1	1	1	0	0	0	2	2	**0**	**0**
0	0	1	1	1	1	-1	2	1	4	**-1**	**-1**
0	1	0	0	0	0	0	0	0	0	**0**	**0**
0	1	0	0	0	1	1	4	1	4	**1**	**1**
0	1	0	0	1	0	1	1	1	1	**1**	**1**
0	1	0	0	1	1	2	5	2	5	**2**	**2**
0	1	0	1	0	0	-3	-3	-3	-3	**-3**	**-3**
0	1	0	1	0	1	-2	1	-2	1	**-2**	**-2**
0	1	0	1	1	0	-2	-2	-2	-2	**-2**	**-2**
0	1	0	1	1	1	-1	2	-1	2	**-1**	**-1**
0	1	1	0	0	0	-1	-1	-1	-1	**-1**	**-1**
0	1	1	0	0	1	-2	1	-2	1	**-2**	**-2**
0	1	1	0	1	0	0	0	0	0	**0**	**0**
0	1	1	0	1	1	-1	2	-1	2	**-1**	**-1**
0	1	1	1	0	0	-4	-4	-2	-2	**-4**	**-4**
0	1	1	1	0	1	-5	-2	-3	0	**-5**	**-5**
0	1	1	1	1	0	-3	-3	-1	-1	**-3**	**-3**
0	1	1	1	1	1	-4	-1	-2	1	**-4**	**-4**
1	0	0	0	0	0	0	0	0	0	**0**	**0**
1	0	0	0	0	1	1	4	1	4	**1**	**1**
1	0	0	0	1	0	-2	-2	-2	-2	**-2**	**-2**
1	0	0	0	1	1	-1	2	-1	2	**-1**	**-1**
1	0	0	1	0	0	3	0	3	0	**0**	**0**
1	0	0	1	0	1	4	4	4	4	**4**	**4**
1	0	0	1	1	0	1	-2	1	-2	**-2**	**-2**
1	0	0	1	1	1	2	2	2	2	**2**	**2**
1	0	1	0	0	0	-1	-1	-1	-1	**-1**	**-1**
1	0	1	0	0	1	-2	1	-2	1	**-2**	**-2**
1	0	1	0	1	0	-3	-3	-3	-3	**-3**	**-3**
1	0	1	0	1	1	-4	-1	-4	-1	**-4**	**-4**
1	0	1	1	0	0	2	-1	4	1	**-1**	**-1**
1	0	1	1	0	1	1	1	3	3	**1**	**1**
1	0	1	1	1	0	0	-3	2	-1	**-3**	**-3**
1	0	1	1	1	1	-1	-1	1	1	**-1**	**-1**
1	1	0	0	0	0	2	2	0	0	**0**	**0**
1	1	0	0	0	1	3	6	1	4	**1**	**1**
1	1	0	0	1	0	0	0	-2	-2	**-2**	**-2**
1	1	0	0	1	1	1	4	-1	2	**-1**	**-1**
1	1	0	1	0	0	2	-1	0	-3	**-3**	**-3**
1	1	0	1	0	1	3	3	1	1	**1**	**1**
1	1	0	1	1	0	0	-3	-2	-5	**-5**	**-5**
1	1	0	1	1	1	1	1	-1	-1	**-1**	**-1**
1	1	1	0	0	0	1	1	-1	-1	**-1**	**-1**
1	1	1	0	0	1	0	3	-2	1	**-2**	**-2**
1	1	1	0	1	0	-1	-1	-3	-3	**-3**	**-3**
1	1	1	0	1	1	-2	1	-4	-1	**-4**	**-4**
1	1	1	1	0	0	1	-2	1	-2	**-2**	**-2**
1	1	1	1	0	1	0	0	0	0	**0**	**0**
1	1	1	1	1	0	-1	-4	-1	-4	**-4**	**-4**
1	1	1	1	1	1	-2	-2	-2	-2	**-2**	**-2**

Appendix C: Cost factor for PTR-BCR-4

If x is an integer then $\lceil x \rceil = x$, otherwise we have $x < \lceil x \rceil < x + 1$. Let us consider $x \equiv \log_2 k/2 = \log_2 k - 1$, and remember that the number of auxiliary variables in PTR-BCR-4 is $m = \lceil \log_2 k/2 \rceil$ in terms of the degree k of the monomial being quadratized, and the added cost factor for a 1-run quadratization such as PTR-BCR-4 is $C = 2^m$ for m auxiliary variables.

When k is an integer power of 2, we have that our x, which was defined as $(\log_2 k) - 1$, is an integer. Therefore we have $\lceil \log_2 k - 1 \rceil = \log_2 k - 1 = \log_2 k/2$ and $C = 2^m = 2^{\lceil \log_2 k/2 \rceil} = 2^{\log_2 k/2} = k/2$.

When k is *not* an integer power of 2, we have that our x, which was defined as $(\log_2 k) - 1 = \log_2 k/2$, is *not* an integer. Therefore we have $\log_2 k/2 < \lceil \log_2 k/2 \rceil < \log_2 k/2 + 1$. Since the exponential function is monotonic, we can exponentiate both sides of either inequality and obtain:

$$2^{\log_2 k/2} < 2^{\lceil \log_2 k/2 \rceil} < 2^{(\log_2 k/2)+1} \tag{C1}$$

$$k/2 < 2^{\lceil \log_2 k/2 \rceil} < k. \tag{C2}$$

Using again the definition of m we now have that 2^m lies in the interval $(k/2, k)$ when k is not an integer power of 2. We can also write *tighter* bounds because the degree k of the monomial must be an integer: The last integer smaller than k is $k - 1$, so we can write a tighter upper bound of $C = 2^m \leq k - 1$; likewise the first integer bigger than $k/2$ is $\lceil k/2 \rceil = \frac{k+1}{2} = k/2 + 1/2$ if k is odd and $k/2 + 1$ if k is even, so the lower bound on $C = 2^m$ is $k/2 + 1/2$ if k is odd, $k/2 + 1$ if k is even but not an integer power of 2, and we have from the earlier paragraph that $C = k/2$ (no lower or upper bound necessary because we have equality) when k is an integer power of 2.

In summary, if k is an integer power of 2, then PTR-BCR-4 adds a cost factor of $C = k/2$. Otherwise, $\frac{k+1}{2} \leq C \leq k-1$. If we do not know whether or not k is an integer power of 2, we can say the cost factor lies somewhere in $[k/2, k-1]$, with the most favorable case of $k/2$ occurring when k is an integer power of 2.

* ndattani@uwaterloo.ca
† as03205@surrey.ac.uk

[1] N. Dattani, *Quadratization in discrete optimization and quantum mechanics* (2019) arXiv:1901.04405.
[2] I. G. Rosenberg, Cahiers du Centre d'Etudes de Recherche Operationnelle **17**, 71 (1975).
[3] H. Ishikawa, IEEE Transactions on Pattern Analysis and Machine Intelligence **33**, 1234 (2011).
[4] E. Boros, Y. Crama, and E. Rodríguez-Heck, *Quadratizations of symmetric pseudo-Boolean functions: sub-linear bounds on the number of auxiliary variables*, Tech. Rep. (2018).
[5] E. Boros, Y. Crama, and E. Rodriguez Heck, *Compact quadratizations for pseudo-Boolean functions* (2018).
[6] H. Ishikawa, in *2014 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2014) pp. 1362–1369.
[7] R. Tanburn, E. Okada, and N. Dattani, (2015), arXiv:1508.04816.
[8] E. Okada, R. Tanburn, and N. S. Dattani, (2015), arXiv:1508.07190.
[9] R. Dridi and H. Alghassi, Scientific Reports **7**, 43048 (2017).
[10] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer, in *2007 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2007) pp. 1–8.
[11] A. D. King, J. Carrasquilla, J. Raymond, I. Ozfidan, E. Andriyash, A. Berkley, M. Reis, T. Lanting, R. Harris, F. Altomare, K. Boothby, P. I. Bunyk, C. Enderud, A. Fréchette, E. Hoskinson, N. Ladizinsky, T. Oh, G. Poulin-Lamarre, C. Rich, Y. Sato, A. Y. Smirnov, L. J. Swenson, M. H. Volkmann, J. Whittaker, J. Yao, E. Ladizinsky, M. W. Johnson, J. Hilton, and M. H. Amin, Nature **560**, 456 (2018).
[12] N. S. Dattani, S. Szalay, and N. Chancellor, Pegasus: The second connectivity graph for large-scale quantum annealing hardware (2019), arXiv:1901.07636.
[13] N. Dattani and N. Chancellor, (2019), arXiv:1901.07676.
[14] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, Physical Review X **6**, 031015 (2016).
[15] S. Mandrà, Z. Zhu, W. Wang, A. Perdomo-Ortiz, and H. G. Katzgraber, Physical Review A **94**, 022337 (2016).
[16] M. Anthony, E. Boros, Y. Crama, and A. Gruber, Mathematical Programming **162**, 115 (2017).
[17] A. G. Gruber, (2015), doi:10.7282/T3PZ5BHQ.
[18] S. Roth and M. Black, in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2 (IEEE, 2005) pp. 860–867.
[19] N. Dattani and H. T. Chau, (2019), arXiv:1910.13583.