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

Nike Dattani[1] and Andreas Soteriou[2]
[1]*Harvard-Smithsonian Center for Astrophysics, USA*
[2]*Surrey University, Department of Mathematics, UK*

We present a quadratization method that we have found to be superior to 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 much more quickly (in terms of computation runtime) than quadratizatons for functions of similar complexity have traditionally been in the past. The idea is that many *very* simple quadratic functions might *almost* reproduce a degree-k function perfectly, but each of these *almost* perfect quadratic functions can compensate for each other's 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 quadratizations*. We are able to quadratize positive monomials with quadratic functions that have exponentially 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, in addition to requiring fewer auxiliary variables. We will show an example in which an 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 degee-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. 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 [1], 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 cut by roughly one half in 2011 by Ishikawa, who introduced a quadratization for a positive degree-k monomial, which needed only $\frac{k-1}{2}$ auxiliary variables (rounded down if k were even). In 2018 several new quadratizations were introduced for positive degree-k monomials: one required only $k/4$ auxiliary variables, one required only $\log k$ (both of these were presented in[2]), and while it might be surprising that an improvement can be made over $\log k$, a subsequent 2018 paper finally showed a quadratization which required only $\log(k/2)$ [3] (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 $\log(k/2)$ 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 [4–8], 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 [8], and in many cases they might be very expensive. One of these zero-auxiliary-variable methods that does reproduce the full spectrum, the split-reduc method [9], can be considered a special case of our *envelope quadratization* method: While split-reduc can be very powerful when quadratizing multiple terms that have common variables, for the degree-k positive monomial alone, its application can be far more costly than the methods described in the previous paragraph (this will be discussed more later in the paper when we compare and contrast split-reduc, methods involving auxiliaries, and our new concept of envelope quadratizations).

In this work we introduce the concept of envelope quadratizations, which seem to be capable of 'perfectly' quadratizing functions more efficiently than any of the above methods (by 'perfectly' we mean, reproducing the whole spectrum and not just the ground state). Even for just preserving the ground state, envelope quadratizations are often the most efficient way to quadratize a function, and in other cases 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 is not *strictly* more efficient than *all* of the previously known quadratization methods.

II. A QUICK EXAMPLE

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

$$b_1b_2b_3 = \min_{b_a}\left(b_1b_a + b_2b_3 - 2b_2b_a - 2b_3b_a + 3b_a\right), \quad (1)$$

where all b_i are either 0 or 1 and the subscript a in b_a indicates that this 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:

b_1	b_2	b_3	$b_1b_2b_3$
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**

b_1	b_2	b_3	b_a	$b_1b_a + b_2b_3 - 2b_2b_a - 2b_3b_a + 3b_a$	$\min_{b_a}$
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 impossible 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_1b_2 and the linear function b_3, perfectly yields the full spectrum of $b_1b_2b_3$ without introducing any auxiliary variables:

b_1	b_2	b_3	b_1b_2	b_3	$\min(b_1b_2, b_3)$
0	0	0	0	0	**0**
0	0	1		1	**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		1	**1**

We call the set $\{b_1b_2, b_3\}$ an *envelope quadratization* for $b_1b_2b_3$. At each point (b_1, b_2, b_3), the minimum (common tangent) over all functions is equal to the degree-k function $b_1b_2b_3$.

III. GENERALIZATIONS OF THE QUICK EXAMPLE

The simple example from the previous section, of a quadratic envelope reproducing $b_1b_2b_3$, can be generalized:

$$b_1b_2b_3\ldots b_k = \min\left(b_1b_2\ldots b_{k_1}, b_{k_1+1}b_{k_1+2}\ldots b_{k_2}, b_{k_2+1}b_{k_2+2}\ldots b_{k_3}, \ldots, b_{k_p+1}b_{k_p+2}\ldots b_k\right), \quad (2)$$

where $k_1, k_2, \ldots, k_p$ can be chosen arbitrarily between 1 and k (if $k_1 > 2$ or if any $k_i > k_{i-1}+2$ then it is no longer a quadratic envelope but an envelope of a higher degree). One interesting quadratic envelope which is an example of Eq. 2 is:

$$b_1b_2b_3\ldots b_k = \min\left(b_1, b_2, b_3, \ldots, b_k\right), \quad (3)$$

which is also an example of a *linear envelope.*

IV. USAGE

If we wished to minimize the function of binary variables:

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

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 quadratic envelope $b_1b_2b_3 = \min(b_1b_2, b_3)$, and we get two objective functions to consider:

$$3b_1b_2 - 2b_1b_2 - 2b_1b_3 + 4b_2b_4 - 3b_4 - 3b_2b_5 \quad (5)$$

$$3b_3 - 2b_1b_2 - 2b_1b_3 + 4b_2b_4 - 3b_4 - 3b_2b_5. \quad (6)$$

Running the optimization procedure (for example QPBO or quantum annealing) on both functions would result in the following minima:

$$\text{Eq.5} : -5 \text{ at } b_1,b_2,b_3,b_4,b_5 = 10110 \text{ \& } 10111 \quad (7)$$
$$\text{Eq.6}: -5 \text{ at } b_1,b_2,b_3,b_4,b_5 = 11001. \quad (8)$$

A. 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:

$$2b_1b_2b_3b_4 - 3b_2b_4 + 3b_1b_4b_6 - 2b_3b_6 - 3b_1b_5 + b_5 + b_6 - b_3. \quad (9)$$

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:

$$2b_1b_2 - 3b_2b_4 + 3b_1b_4 - 2b_3b_6 - 3b_1b_5 + b_5 + b_6 - b_3 \quad (10)$$

$$2b_1b_2 - 3b_2b_4 + 3b_6 - 2b_3b_6 - 3b_1b_5 + b_5 + b_6 - b_3 \quad (11)$$

$$2b_3b_4 - 3b_2b_4 + 3b_1b_4 - 2b_3b_6 - 3b_1b_5 + b_5 + b_6 - b_3 \quad (12)$$

$$2b_3b_4 - 3b_2b_4 + 3b_6 - 2b_3b_6 - 3b_1b_5 + b_5 + b_6 - b_3 \quad (13)$$

Running the optimization procedure on all four functions would result in the following minima:

$$\text{Eq.10} : -5 \text{ at } b_1,b_2,b_3,b_4,b_5,b_6 = 011101 \quad (14)$$
$$\text{Eq.11}: -4 \text{ at } b_1,b_2,b_3,b_4,b_5,b_6 = 011100 \text{ \& } 111110 \quad (15)$$
$$\text{Eq.12}: -4 \text{ at } b_1,b_2,b_3,b_4,b_5,b_6 = 101011 \text{ \& } 111011 \quad (16)$$
$$\text{Eq.13}: -5 \text{ at } b_1,b_2,b_3,b_4,b_5,b_6 = 111011. \quad (17)$$

The functions which have the lowest minima have values of -5 at 011101 and 111011 respectively, so these are the global minima of the original degree-k function (again, this can be verified by simply optimizing the original quartic function, but for more complicated functions, 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 $2^6 = 64$ 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. 9 and f_1, f_2, f_3, f_4 are the functions in Eqs. 10-13 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 our Appendix.

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^n2^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 cost $\mathcal{O}(2^x)$ for x 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 we claim 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 of the added cost to the optimization due to adding auxiliary variables) .

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 m auxiliary variables, then the following is an upper bound on the added cost C due to quadratizing:

$$C \leq r2^m. \quad (18)$$

In practice, some runs may require fewer auxiliary variables than others (or none at all), so the added cost can be lower than $r2^m$.

VI. CLAIMS OF SUPERIORITY OVER ALL 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. The state-of-the-art method for this was the 2018 method requiring only $\log_2(k/2)$ auxiliary variables (rounded up) [2, 3], and in the handbook of quadratizations [8] it 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 (which was previously the state-of-the-art). PTR-BCR-4 adds $m = \lceil \log_2(k/2) \rceil$ auxiliary variables, which means that the added cost of optimization can be increased by up to a factor of k (see footnote: [1]).

Using Eq. 2 we can split the positive monomial into k/p pieces, each with maximum degree p, and each piece can be quadratized with PTR-BCR-4 with $\log_2\lceil p/2 \rceil$ auxiliary variables, incurring a cost of $\lceil p/2 \rceil$ for each of the k/p pieces. Since to optimize the original function containing $b_1 b_2 \dots b_k$, we would need to do $r = k/p$ runs of our optimization procedure, the total added cost would then be $r 2^m = k/p \lceil p/2 \rceil = k/2$.

For example if we wish to optimize a function in which we quadratize a positive degree-40 monomial using PTR-BCR-4, we would be adding $\lceil \log_2 40/2 \rceil = \lceil \log_2 20 \rceil = 5$ auxiliary variables, and therefore increase the cost of optimizing the original function by a factor of $2^5 = 32$. However the positive degree-40 monomial can be split into 20 quadratic pieces: $\{b_1 b_2, b_3 b_4, \cdots, b_{39} b_{40}\}$, and doing $r = 20$ different optimizations (replacing $b_1 b_2 \dots b_{40}$ in the original function by one of each of the 20 total pieces) would cost at most only 20 times more than doing one such optimization. Likewise the positive degree-40 monomial can be split into $r = 10$ pieces, each of degree-4: $\{b_1 b_2 b_3 b_4, b_5 b_6 b_7 b_8, \cdots, b_{37} b_{38} b_{39} b_{40}\}$, and then each dergree-4 term can be quadratized using PTR-BCR-4 with only $m = 1$ auxiliary variable, resulting again in a cost of $10 \times 2^1 = 20$. Finally the same would be true if we used $r = 5$ pieces, each of degree-8: $\{b_1 b_2 \cdots b_8, b_9 b_{10} \cdots b_{16}, \cdots, b_{33} b_{34} \cdots b_{40}\}$,and then quadratized each degree-8 term using PTR-BCR-4 with only $m = 2$ auxiliary variables each, resulting *again* in a cost of $5 \times 2^2 = 20$.

In general we can get up to a factor of 2 improvement over PTR-BCR-4, equivalent to effectively using only log(k/4) auxiliaries? and even better if the other variables don't appear in positive/negative coefficients of other terms... i.e. negative number of auxiliaries... Furthermore compare the coefficients and connectivity in BCR to COMIGs.

We note that we can also give COMIGs that are *not* strictly better in terms of the added cost in the spirit of Equation..., for example for k=40 if we did r=8 instead, we would get C=32 which is the same as PTR-BCR-4, but in that case our number of quadratic terms is, and our gadget graph is, and the coefficients are.....

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 the cost added to the optimization process. What about when quadratizing not only a single monomial but multiple terms in a function? PTR-BCR-4 is no longer necessarily the state-of-the-art. For example, quadratizing $b_1 b_2 b_3 b_4 b_5 + b_2 b_3 b_4 b_5 b_6$ would require 4 auxiliary variables, but Rosenberg's substitution would only need two auxiliary variables: $b_{a_1} = b_2 b_3$ and $b_{a_2} = b_4 b_5$. 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 [put book on arXiv and cite it], we are also able to obtain an envelope quadratization with $r \geq 1$ that is strictly better than the previous state-of-the-art quadratization. There are some cases in which the cost, in the spirit of Eq. 18 can be matched by previously known methods, but in those cases we have fewer quadratic terms and smaller coefficients and fewer non-submodular terms, see below:

VII. MORE EXAMPLES AND POTENTIAL APPLICATIONS

A. Positive degree-8 monimial

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. 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:

[1] $x \leq \lceil x \rceil \leq 1 + \lfloor x \rfloor$, because if x is an integer then $\lceil x \rceil = x = \min\lceil x \rceil$ and if x is not an integer then we have $\lceil x \rceil = 1 + \lfloor x \rfloor = \max\lceil x \rceil$. So if we have $m = \lceil \log_2 k/2 \rceil = \lceil \log_2 k \rceil - 1$ and we are interested in 2^m, we have $\min(2^m) = 2^{\min\lceil \log_2 k \rceil - 1} = 2^{\log_2 k - 1} = k/2$ in the case of k being a power of 2, and $\max(2^m) = 2^{\max\lceil \log_2 k \rceil - 1} = 2^{1 + \lfloor \log_2 k \rfloor - 1} = 2^{\lfloor \log_2 k \rfloor} =$.

$$b_1b_2b_3b_4b_5b_6b_7b_8 : \quad (19)$$
$$\longrightarrow 3b_a + b_1b_2 + b_1b_3 + b_1b_4 + b_2b_3 + b_2b_4 \quad (20)$$
$$+ b_3b_4 - 2b_a(b_1 + b_2 + b_3 + b_4) \quad (21)$$
$$\longrightarrow 3b_a + b_5b_6 + b_5b_7 + b_5b_8 + b_6b_7 + b_6b_8 \quad (22)$$
$$+ b_7b_8 - 2b_a(b_5 + b_6 + b_7 + b_8) \quad (23)$$

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 the above envelope quadratization would have coefficients ranging from X to X, and in the above envelope quadratization, they range from -2 to 1, which means the coefficient range is X times smaller in the envelope quadratization case. Submodularity?

B. Application to a computer vision algorithm

The image deblurring algorithm of [] involves degree-k terms with $k = d^2$ with $d \geq 2$ being the length in pixels of the square-shaped mask) [16, 17]. In all known applications so far, $d = 2$ and $k = 4$ (though larger masks can be used, it would just be more difficult since the optimization problem would involve much higher-degree terms). With $d = 2$, this optimizaton problem involves degree-4 terms for every set of pixels that can be covered by the 2×2 pixel mask, so we can have neighboring degree-4 terms such as.....

$$b_1b_2b_3b_4 + b_3b_4b_5b_6 = \min(b_2b_3 + b_3b_6, b_1b_4 + b_4b_5, \quad (24)$$
$$b_1b_2 + b_5b_6 - b_3 - b_4 + 2) \quad (25)$$

3-runs/0-aux case:

$$b_1b_2b_3b_4 + b_2b_3b_4b_5 + b_3b_4b_5b_6 : \quad (26)$$
$$\longrightarrow 3b_3b_4 + b_3b_5 + b_4b_5 - b_3 - b_4 - b_5 + 1 \quad (27)$$
$$\longrightarrow b_1b_4 + b_3b_5 + b_4b_5 \quad (28)$$
$$\longrightarrow b_1b_2 + b_2b_6 + b_3b_5 + b_5b_6 + b_2 - b_3 - b_4 - b_5 + 2 \quad (29)$$

C. One more example which is the best one we can come up with for cost savings.

VIII. DISCUSSION

Negative variables. Reinforcement learning. Conjecture about this always being better than other methods. This is still not the end of the journey because we're reproducing the full spectrum which is too much.

IX. ACKNOWLEDGMENTS

[1] I. G. Rosenberg, Cahiers du Centre d'Etudes de Recherche Operationnelle **17**, 71 (1975).
[2] 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).
[3] E. Boros, Y. Crama, and E. Rodriguez Heck, *Compact quadratizations for pseudo-Boolean functions* (2018).
[4] H. Ishikawa, in *2014 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2014) pp. 1362–1369.
[5] R. Tanburn, E. Okada, and N. Dattani, (2015), arXiv:1508.04816.
[6] E. Okada, R. Tanburn, and N. S. Dattani, (2015), arXiv:1508.07190.
[7] R. Dridi and H. Alghassi, Scientific Reports **7**, 43048 (2017).
[8] N. Dattani, *Quadratization in discrete optimization and quantum mechanics* (2019) arXiv:1901.04405.
[9] E. Okada, R. Tanburn, and N. S. Dattani, *Reducing multi-qubit interactions in adiabatic quantum computation without adding auxiliary qubits. Part 2: The "split-reduc" method and its application to quantum determination of Ramsey numbers* (2015) arXiv:1508.07190.
[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] H. Ishikawa, IEEE Transactions on Pattern Analysis and Machine Intelligence **33**, 1234 (2011).

[17] A. Fix, A. Gruber, E. Boros, and R. Zabih, in *2011 International Conference on Computer Vision* (IEEE, 2011) pp. 1020–1027.

Appendix A: Proof of...

b_1	b_2	b_3	b_4	b_5	b_6	f_1 (Eq. 10)	f_2 (Eq. 11)	f_3 (Eq. 12)	f_4 (Eq. 13)	$\min_{f_1 f_2 f_3 f_4}$	f (Eq. 9)
0	0	0	0	0	0	0				**0**	
0	0	0	0	0	1	3					
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
0	0	0	0	0	1						
1	0	0	0	0	0						
1	0	0	0	0	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						
1	1	1	1	1	1						