Centrum voor Wiskunde en Informatica

REPORT *RAPPORT*

Nonconvex Continuous Models for Combinatorial Optimization Problems with Application to Satisfiability and Node Packing Problems

J.P. Warners

Software Engineering (SEN)

SEN-R9710 July 31, 1997

Report SEN-R9710
ISSN 1386-369X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Nonconvex Continuous Models for Combinatorial Optimization Problems with Application to Satisfiability and Node Packing Problems

Joost P. Warners

CWI

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Department of Probability, Statistics and Operations Research
Faculty of Technical Mathematics and Informatics
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

ABSTRACT

We show how a large class of combinatorial optimization problems can be reformulated as a nonconvex minimization problem over the unit hyper cube with continuous variables. No additional constraints are required; all constraints are incorporated in the nonconvex objective function, which is a polynomial function. The application of the general transform to satisfiability and node packing problems is discussed, and various approximation algorithms are briefly reviewed. To give an indication of the strength of the proposed approaches, we conclude with some computational results on instances of the graph coloring problem.

1991 Mathematics Subject Classification: 90C30, 90C27, 90C09
1991 Computing Reviews Classification System: G.0, F.0
Keywords and Phrases: Modelling, Nonlinear optimization, Combinatorial optimization, Satisfiability
Note: This research was supported by the Dutch Organization for Scientific Research (NWO) under grant SION 612-33-001. Work carried out under project SEN 2.3 Proof searching and proof checking.

1. Introduction

In a number of recent papers [14, 15, 13] we have developed and tested various approximation algorithms for solving a specific class of combinatorial optimization problems (so–called *node packing* problems), based on optimizing a nonconvex quadratic model over the unit hyper cube. In this paper we generalize the ideas leading to the model used in the mentioned papers to show how any 0–1 feasibility problem with linear constraints can be transformed to a nonconvex minimization problem over the unit hypercube. There exist several techniques to arrive at such a model. In this paper we will describe two of these.

In the first technique, the set of linear constraints is replaced by an equivalent set of linear constraints that exhibits certain properties. The method used to obtain an appropriate linear reformulation uses techniques similar to those used in, among others, Hammer and Rudeanu [6], Granot and Hammer [4], Nemhauser and Wolsey [10] and Barth [1]. Subsequently, we show how the set of linear constraints obtained can be transformed to a polynomial function, such that global minimizers of this function yield feasible binary solutions to the original problem. Conversely, each solution to the original problem coincides with a global minimizer of the polynomial function. Unfortunately, the size of the reformulation may be intractable. That is, there are linear inequalities which need an exponential number of linear inequalities to replace them. There does however exist a method that garantuees that the number of linear inequalities needed is linear in the length of the original inequality. It makes use of the binary representation of the coefficients occurring in the inequality and entails the introduction of new variables. In this paper we do not further pursue this approach; we refer to Warners [12]. It may be emphasized that for many important problem classes no linear reformulation is required; in particular the class of node packing problems that we studied earlier [14, 15, 13] and also the *satisfiability* problem [5]. The second technique to construct the nonconvex continuous model makes use of the fact that binary variables are idempotent; it is inspired by the approach of [14].

It turns out that, apart from the problems we studied earlier, also the transformation for satisfiability problems introduced by Gu [5] is a special case of the general transformation scheme.

This paper is organized as follows. In the next section we describe the techniques to obtain the reformulation, and in Section 3 we consider two special cases; some algorithmic approaches to the given minimization problem are discussed and computational results are reported on the graph coloring problem. Concluding remarks are made in the final section.

Acknowledgement Thanks are due to Hans van Maaren and Tamás Terlaky for their comments on earlier versions of this paper.

2. A NONCONVEX CONTINUOUS MODEL

We consider binary feasibility problems of the form

$$(BP) \qquad \text{find } x \in \{0,1\}^m \text{ such that } Ax \leq b.$$

Here $A \in I\!R^{n \times m}$, $b \in I\!R^n$ and it is assumed that all the data are integral. Many combinatorial optimization problems can be put into this form by modelling them as integer linear programming problems and, if required, adding a bound on the objective function value. We have the following theorem.

Theorem 1 *The problem (BP) can be reformulated as a nonconvex optimization problem over the unit hypercube with continuous variables.*

In fact, for discrete (instead of continuous) variables the content of this theorem is almost trivial. It is possible to construct such a nonconvex model by simply enumerating all 2^m vertices of the m–dimensional hyper cube. Let V be the set of vertices v that do not satisfy $Av \leq b$. Consider the following model.

$$(VP) \qquad \begin{array}{ll} \min & \mathcal{P}_V(x) = \sum_{v \in V} \prod_{i=1}^{m} (1 \Leftrightarrow v_i + (2v_i \Leftrightarrow 1)x_i) \\ \text{s.t.} & x \in \{0,1\}^m \end{array}$$

By the following lemma, optimizing (VP) either yields a solution to (BP), or it proves that (BP) does not have solution.

Lemma 1 *For any $x \in \{0,1\}^m$ that does not satisfy $Ax \leq b$, it holds that $\mathcal{P}_V(x) = 1$. If $Ax \leq b$, then $\mathcal{P}_V(x) = 0$.*

Proof: By construction $\mathcal{P}_V(x) \geq 0$. Note that if $x \equiv v$ for some vertex v then

$$p_v(x) = \prod_{i=1}^{m}(1 \Leftrightarrow v_i + (2v_i \Leftrightarrow 1)x_i) = \prod_{i=1}^{m}(1 \Leftrightarrow v_i + 2v_i^2 \Leftrightarrow v_i) = 1,$$

where we use that $v_i^2 = v_i$. On the other hand, if $x \not\equiv v$, there is an i for which $x_i = 1 \Leftrightarrow v_i$ implying that for that particular i it holds that $(1 \Leftrightarrow v_i + (2v_i \Leftrightarrow 1)(1 \Leftrightarrow v_i)) = 0$, from which it follows that $p_v(x) = 0$. Thus $\mathcal{P}_V(x) = 1$ if and only if $x \in V$, otherwise $\mathcal{P}_V(x) = 0$. □

Thus we have the following corollary.

Corollary 1 *(BP) is feasible if and only if the global optimal value of (VP) is equal to zero.*

Using the following theorem, this result can be generalized to continuous variables.

Theorem 2 *Let*

$$\mathcal{P}(x) = \sum_{k=1}^{K} c_k \prod_{i \in I_k} x_i,$$

where $x \in I\!R^m$, $c_k \in I\!R$, and the sets I_k are sets of indices. Given a vector x such that $l \leq x \leq u$ where l and u are vectors of lower and upper bounds on x. By using a greedy rounding strategy, a solution $[x]$ can be constructed, such that for each i, $[x_i]$ is equal to either its lower or upper bound, which has the property that $\mathcal{P}([x]) \leq \mathcal{P}(x)$.

Proof: We are given a vector x; if for each i, x_i is equal to l_i or u_i, we are ready. Otherwise, there is an i such that $l_i < x_i < u_i$. For the moment, fix all variables to their current value, except the variable x_i. It is our purpose to set this variable to either l_i or u_i, such that the value of $\mathcal{P}(x)$ does not increase. This amounts to minimizing a one–variable linear function over the constraint $l_i \leq x_i \leq u_i$. Obviously, the optimal value is attained in one of the extreme points, in which case we set x_i to this extreme value, or on the whole interval; in that case we can arbitrarily set x_i to either its upper or lower bound. This procedure can be carried out for all variables, until a binary solution is constructed such that $\mathcal{P}([x]) \leq \mathcal{P}(x)$. □

We conclude that the integrality constraints in (VP) can be relaxed to linear constraints $0 \leq x \leq e$ to obtain the following corollary. By e an all–one vector is denoted.

Corollary 2 *(BP) is feasible if and only if there exists a vector x, $0 \leq x \leq e$, such that $\mathcal{P}_V(x) < 1$. Here e denotes the all–one vector.*

Thus we have proven Theorem 1. Since its obviously intractable to construct the nonconvex continuous model as we have done above (constructing the model in this way is equivalent to solving the problem), we are interested in finding other techniques to obtain a similar model. In the subsequent sections, we consider two techniques to find such a similar model.

2.1 Using minimal covers

Let us first consider some examples to get a better understanding of the crucial issues.

Example 1 Consider the inequality

$$x_1 + x_2 \leq 1. \tag{2.1}$$

Since we require that both x_1 and x_2 are binary, we can rewrite this inequality in a stronger, but nonlinear, form:

$$x_1 x_2 = 0. \tag{2.2}$$

Clearly, we have for *continuous* variables x_1 and x_2 that

$$x_1 x_2 = 0 \;\Rightarrow\; x_1 + x_2 \leq 1.$$

This shows that (2.2) is stronger than (2.1), as the converse is not necessarily true. The crucial point is that of the two variables occurring, at least *one* should be equal to zero. □

Example 2 Now consider the following inequality:

$$x_1 + x_2 + x_3 + x_4 \leq 2. \tag{2.3}$$

Again, we must have

$$x_1 x_2 x_3 x_4 = 0, \tag{2.4}$$

but unfortunately this does not imply that (2.3) is satisfied. For example, if we take $(x_1, x_2, x_3, x_4) = (1, 1, 1, 0)$, (2.4) holds, but (2.3) is violated. However, we may replace (2.3) by the following four inequalities:

$$\begin{array}{ccccc} & +x_2 & +x_3 & +x_4 & \leq 2 \\ x_1 & & +x_3 & +x_4 & \leq 2 \\ x_1 & +x_2 & & +x_4 & \leq 2 \\ x_1 & +x_2 & +x_3 & & \leq 2 \end{array} \tag{2.5}$$

Note that inequality (2.3) in fact can be interpreted as a *Chvátal cut* [3], that can be obtained by adding the four inequalities (2.5) and rounding the right hand side downwards:

$$3(x_1 + x_2 + x_3 + x_4) \leq 8 \;\Rightarrow\; x_1 + x_2 + x_3 + x_4 \leq \lfloor 8/3 \rfloor = 2.$$

Recall that a cut, if we solve the linear relaxation, forces the variables to integer values. For the integer formulation however, it is redundant. So for a *binary* solution, we have that (2.5) is satisfied if and only if (2.3) is satisfied. Now we can reformulate the original inequality (2.3) as a polynomial function:

$$x_2 x_3 x_4 + x_1 x_2 x_4 + x_1 x_3 x_4 + x_1 x_2 x_3 = 0. \tag{2.6}$$

Obviously, if (2.6) is satisfied by a solution $0 \leq x \leq e$, the system (2.5) is satisfied, so (2.3) is satisfied. □

In the following we formalize what is exposed by the examples. To this end, let us first introduce some notation with the purpose to show that any inequality with binary variables can be rewritten in such a way that all coefficients are nonnegative. Let the j^{th} inequality contained in $Ax \leq b$ be denoted by

$$a_j^T x = \sum_{j=1}^{m} a_{ji} x_i \leq b_j. \tag{2.7}$$

Define $I_j = \{i : a_{ji} > 0\}$, $J_j = \{i : a_{ji} < 0\}$ and $K_j = I_j \cup J_j$. For the sake of brevity we shall adopt the following notation:

$$\overline{x}_i = \begin{cases} x_i & \text{if } i \in I_j \\ 1 \Leftrightarrow x_i & \text{if } i \in J_j \end{cases},$$

and

$$\overline{b}_j = b_j \Leftrightarrow \sum_{i \in J_j} a_{ji}.$$

Then (2.7) can be rewritten as

$$\sum_{i \in K_j} |a_{ji}| \, \overline{x}_i \leq \overline{b}_j. \tag{2.8}$$

We first consider inequalities of a particular form. By e we denote an all–one vector of appropriate length. We give the following definition.

Definition 1 *Given an inequality $e^T x \leq \beta$. The* maximum violation ϑ *of this inequality is defined as*

$$\vartheta = \max_{x \in \{0,1\}^r} e^T x \Leftrightarrow \beta = r \Leftrightarrow \beta.$$

Obviously, if we have $\vartheta \leq 0$, the associated inequality is trivially satisfied. Therefore in the following we only consider inequalities that have a strictly positive maximum violation. Note that for inequality (2.1) of the first example we have that $\vartheta = 1$, while for inequality (2.3) of the second example, $\vartheta = 2$. For the inequalities (2.5) however, we have that $\vartheta = 1$.

Let us prove an easy lemma.

Lemma 2 *Given an inequality $e^T x \leq \beta$. For $0 \leq x \leq e$, the implication*

$$p_j(x) = \prod_{i=1}^{r} x_i = 0 \;\Rightarrow\; e^T x \leq \beta$$

holds if and only if the maximum violation $\vartheta \leq 1$.

Proof: Suppose the implication holds, i.e. for any x with $p_j(x) = 0$ we have that $e^T x \leq \beta$. There exists at least one $i \in \{1, \ldots, r\}$ such that $x_i = 0$. We have $e^T x \leq r \Leftrightarrow 1 \leq \beta$. It follows that $\vartheta = r \Leftrightarrow \beta \leq 1$. On the other hand, suppose that $\vartheta \leq 1$. This implies that $r \Leftrightarrow \beta \leq 1$ so $r \leq \beta + 1$. Now assume that for a given binary x it holds that $p_j(x) = 0$, then there exists an $i \in \{1, \ldots, r\}$ such that $x_i = 0$. Thus $e^T x \leq r \Leftrightarrow 1 \leq \beta$. □

Lemma 2 suggests that given an inequality $|a_j|^T \overline{x} \leq \overline{b}_j$ where the coefficients are arbitrary numbers, we should replace this inequality by a number of inequalities having binary coefficients and maximum violation one. This can be done by making use of *minimal covers* [10].

Definition 2 *Define the sets $K_{jk} \subseteq K_j$, $k = 1, 2, \ldots, N_j$, such that for all k and $l \in K_{jk}$ the following holds:*

$$\sum_{i \in K_{jk} \setminus \{l\}} |a_{ji}| \leq \overline{b}_j < \sum_{i \in K_{jk}} |a_{ji}|.$$

A set K_{jk} is called a minimal cover *of inequality j [10].*

Given an arbitrary inequality, assume that we have constructed all its distinct minimal covers. The number of minimal covers is denoted by N_j. With each of these covers we associate an inequality

$$\sum_{i \in K_{jk}} \overline{x}_i \leq |K_{jk}| \Leftrightarrow 1,$$

and we denote the union of these inequalities by $C\overline{x} \leq c$. It is obvious that all inequalities in $C\overline{x} \leq c$ have a maximum violation of one. Furthermore, we can prove the following equivalency.

Lemma 3 *For binary x, $|a_j|^T x \leq \overline{b}_j$ if and only if $C\overline{x} \leq c$.*

Proof:

- Given x such that $a_j^T x \leq b_j$. Suppose that for a $k = 1, 2, \dots, N_j$ we have that

  $$\sum_{i \in K_{jk}} \overline{x}_i > |K_{jk}| \Leftrightarrow 1.$$

  This implies that $\overline{x}_i = 1$ for all $i \in K_{jk}$, from which it follows that

  $$|a_j|^T \overline{x} \geq \sum_{i \in K_{jk}} |a_{ji}| \, \overline{x}_i = \sum_{i \in K_{jk}} |a_{ji}| > \overline{b}_j.$$

  This contradicts the fact that $a_j^T x \leq b_j$. We conclude that $C\overline{x} \leq c$.

- Now we are given an x such that $C\overline{x} \leq c$. Let $\Omega = \{i \in K_j : \overline{x}_i = 1\}$. Suppose that

  $$|a_j|^T \overline{x} = \sum_{i \in \Omega} |a_{ji}| \, \overline{x}_i = \sum_{i \in \Omega} |a_{ji}| > \overline{b}_j.$$

  Sort the indices $i \in \Omega$ such that $|a_{ji_1}| \geq |a_{ji_2}| \geq \dots \geq |a_{ji_m}|$. Now for some t it holds that $\sum_{l=1}^{t} |a_{ji_l}| \leq \overline{b}_j$, while $\sum_{l=1}^{t+1} |a_{ji_l}| > \overline{b}_j$. By construction, $\{i_1, i_2, \dots, i_{t+1}\} = K_{jk}$ for some $k = 1, 2, \dots, N_j$ which leads again to a contradiction.

□

The above is applied in the following example.

Example 3 Consider the inequality

$$4x_1 + 6x_2 \Leftrightarrow 3x_3 \Leftrightarrow 5x_4 + 10x_5 \leq 7. \tag{2.9}$$

We rewrite this inequality as follows:

$$4\overline{x}_1 + 6\overline{x}_2 + 3\overline{x}_3 + 5\overline{x}_4 + 10\overline{x}_5 \leq 15.$$

This is equivalent with the following set of inequalities:

$$\begin{array}{cccccl}
 & x_2 & & & +x_5 & \leq 1 \\
x_1 & & & +(1 \Leftrightarrow x_4) & +x_5 & \leq 2 \\
x_1 & & +(1 \Leftrightarrow x_3) & & +x_5 & \leq 2 \\
 & & (1 \Leftrightarrow x_3) & +(1 \Leftrightarrow x_4) & +x_5 & \leq 2 \\
x_1 & +x_2 & +(1 \Leftrightarrow x_3) & +(1 \Leftrightarrow x_4) & & \leq 3
\end{array}$$

□

Now we are ready to prove the following important lemma.

Lemma 4 *Any inequality $a_j^T x \le b_j$ is equivalent with a polynomial equation*

$$P_j(x) = \sum_{k=1}^{N_j} p_{jk}(x) = 0,$$

where N_j is the number of minimal covers of inequality j. Here, the $p_{jk}(x)$ are polynomial functions, each corresponding with a minimal cover and constructed in accordance with Lemma 2. For any x with $0 \le x \le e$ we have that $p_{jk}(x) \ge 0$, implying that $P_j(x) \ge 0$, and

$$P_j(x) = 0 \ \Leftrightarrow P_j([x]) = 0 \Leftrightarrow \ a_j^T[x] \le b_j,$$

where by $[x]$ we denote a binary solution that is obtained by rounding all fractional elements of x either up or down.

Proof: Suppose we are given the inequality $a_j^T x \le b_j$. It can be rewritten in the form (2.8), and subsequently replaced by the set of inequalities $C\overline{x} \le c$. The polynomial $P_j(x)$ is given by

$$P_j(x) = \sum_{k=1}^{N_j} p_{jk}(x) = \sum_{k=1}^{N_j} \prod_{i \in K_{jk}} \overline{x}_i.$$

(see also Lemma 2). It is clear that $P_j(x) \ge 0$ for any $0 \le x \le e$. Furthermore, if $P_j(x) = 0$ for some $0 \le x \le e$, this implies that each term $p_{jk}(x)$ contributes zero to the total sum; this implies (by Lemma 2) that $C\overline{x} \le c$. Moreover, we have

$$P_j(x) = 0 \Leftrightarrow P_j([x]) = 0 \Leftrightarrow C[x] \le c \Leftrightarrow a_j^T[x] \le b_j,$$

(using Lemma 3), thus proving the lemma. □

We have the following corollaries.

Corollary 3 *The degree of the polynomial associated with an inequality j is equal to*

$$\max_{k=1,\dots,N_j} |K_{jk}|,$$

i.e. the size of its largest minimal cover.

Corollary 4 *From a (partly) fractional vector x, $0 \le x \le e$, for which it holds that $P_j(x) = 0$, multiple binary solutions $[x]$ such that $a_j^T[x] \le b_j$ can be constructed.*

Example 4 We use inequality (2.9) from the previous example. The largest minimal cover has size 4, so the polynomial $P_j(x)$ has degree 4. It is given by

$$P_j(x) = x_1x_2(1 \Leftrightarrow x_3)(1 \Leftrightarrow x_4) + (1 \Leftrightarrow x_3)(1 \Leftrightarrow x_4)x_5 + x_1(1 \Leftrightarrow x_3)x_5 + x_1(1 \Leftrightarrow x_4)x_5 + x_2x_5.$$

Now to illustrate the 'rounding property', consider the vector $\tilde{x} = (\ * \ \ * \ \ * \ \ 1 \ \ 0 \ )$. It holds that $P(\tilde{x}) = 0$, irrespective of the values x_i, $i = 1, 2, 3$. So these variables can be chosen arbitrarily, yielding 8 distinct solutions. The same holds for e.g. $\tilde{x} = (\ 0 \ \ 0 \ \ 1 \ \ * \ \ * \ )$ and $\tilde{x} = (\ 0 \ \ * \ \ * \ \ * \ \ 0 \ )$, yielding 4 and 8 distinct solutions respectively. □

Now we are ready to construct the nonlinear model of (BP). Each inequality j is replaced by its equivalent set of inequalities with maximum violation one, and the polynomial $P_j(x)$ is constructed. Finally, we take the sum over all these polynomials to obtain the nonconvex minimization problem that is equivalent with (BP):

$$(NMP) \qquad \begin{array}{ll} \min & \mathcal{P}(x) = \sum_{j=1}^{n} P_j(x) = \sum_{j=1}^{n} \sum_{k=1}^{N_j} \prod_{i \in K_{jk}} \overline{x}_i \\ \text{s.t.} & 0 \le x \le e. \end{array}$$

If model (BP) has a feasible solution, the optimal value of (NMP) equals zero, and (by Corollary 4) the corresponding minimizer yields one or possibly multiple feasible solutions when it is rounded to a binary solution.

Note that we can apply Theorem 2 to yield the following corollaries.

Corollary 5 *All* strict *minima of (NMP) have integral values.*

Corollary 6 *If a minimizer of (NMP) has a non–integral objective value, it can be rounded to a binary solution with improved objective value.*

In particular, a fractional solution with objective value smaller than one yields a feasible binary solution to the original problem.

If a binary solution has a positive objective value, it can be interpreted as follows.

Lemma 5 *The objective value of a binary solution x of (NMP) is equal to the number of minimal covers in the reformulation of (BP) that is completely covered by x.*

Proof: Follows directly from the construction of the model. $\square$

Thus the objective value gives an upper bound on the number of constraint violations in the original formulation (BP).

Note that model (NMP) is in this respect different from model (VP), whose objective value always lies in the interval $[0, 1]$.

If the coefficients of a constraint are all equal to -1, 0 or 1 we can say more beforehand about the number of inequalities that need to be generated to replace a given constraint.

Lemma 6 *The number N_j of inequalities with maximum violation one, required to replace a given inequality $a_j^T x \le b_j$, $a_{ji} \in \{-1, 0, 1\}$, written in the form (2.8), is equal to*

$$N_j = \begin{pmatrix} |K_j| \\ \overline{b}_j + 1 \end{pmatrix}.$$

Proof: To construct *all* minimal covers, we need to find all sets K_{jk} such that $|K_{jk}| = \overline{b}_j + 1$. This amounts to finding all combinations of $\overline{b}_j + 1$ elements out of $|K_j|$ elements. $\square$

So for $\overline{b}_j = \frac{1}{2}|K_j|$, the number of inequalities required to replace inequality j is exponential in the length of this inequality. Thus, in specific applications, performing the procedure as previously described will prove to be computationally intractable. However, there is a linear time algorithm to replace an arbitrary inequality by a set of inequalities with maximum violation one [12]. This algorithm makes use of the binary representation of the coefficients in the inequality. The practical drawback of this algorithm is that it requires the introduction of a substantial (albeit linear) number of additional variables and constraints. We do not explain this algorithm here.

In general, to find all minimal covers of a given inequality, implicit enumeration of all possible assignments to the variables occurring in this inequality is required. This can be done by setting up a search tree in the usual way; at each node a variable $\overline{x}_i$ is set to one in its left branch and to zero in its right branch. First sorting the coefficients in descending order, the search tree can be kept relatively small by choosing variable x_i as branching variable at depth $i+1$. A branch is closed when the partial assignment is such that the constraints is violated.

2.2 Using indempotency of the variables

In this section it is shown that using another quite natural approach one arrives at essentially the same model as described in the previous section. This approach is motivated by the one in [14]; see also Section 3.2. In the mentioned paper, for a special class of combinatorial optimization problems a nonconvex quadratic model is constructed. For the problems considered, it holds that for any feasible binary solution also the slacks are binary. By using idempotency of the variables (i.e. $x_i^2 = x_i$) a concise and computationally attractive model is obtained.

For the moment we restrict ourselves to inequalities with binary coefficients. Consider the inequality

$$e^T x = \sum_{i=1}^{r} x_i \leq \beta. \tag{2.10}$$

Since x must be binary, vectors x satisfying this inequality are such that $e^T x$ is equal to either $0, 1, \ldots, \beta \Leftrightarrow 1$ or β (assuming that β is integral). So it holds that

$$e^T x(e^T x \Leftrightarrow 1) \ldots (e^T x \Leftrightarrow \beta) = 0. \tag{2.11}$$

We have the following theorem.

Theorem 3 *Using idempotency of the variables, (2.11) reduces to*

$$\mathcal{P}_\beta(x) = \sum_{i=1}^{N_\beta} \prod_{j \in K_i^\beta} x_j = 0,$$

where N_β is the number of minimal covers of (2.10), and the K_i^β are the index sets of its minimal covers.

Proof: We use induction. Let us compute the product (2.11), 'from left to right'. Consider first the product $e^T x(e^T x \Leftrightarrow 1)$. Using idempotency of the variables, we find

$$\mathcal{P}_2(x) = e^T x(e^T x \Leftrightarrow 1) = \sum_{i=1}^{r} \sum_{j=1}^{r} x_i x_j \Leftrightarrow \sum_{i=1}^{r} x_i = 2 \sum_{i=1}^{r-1} \sum_{j=i+1}^{r} x_i x_j.$$

In the right hand side of this expression, each pair $x_i x_j$, $i, j = 1, \ldots, r, i \neq j$ occurs exactly once, with a coefficient that we may omit in the following. Now suppose that we have carried out the multiplication for the first k factors $(0 < k < \beta)$, and we have obtained the sum over $N_k = \begin{pmatrix} r \\ k \end{pmatrix}$ distinct terms, each consisting of k letters. We denote this by

$$\mathcal{P}_k(x) = \sum_{i=1}^{N_k} \prod_{j \in K_i^k} x_j.$$

Note that each of these terms represents in fact a minimal cover of the inequality $e^T x \leq k \Leftrightarrow 1$, i.e. K^k contains all minimal covers of this inequality. Now consider the multiplication of $\mathcal{P}_k(x)$ with $e^T x$. Again using the idempotency of the variables, each of the terms with length k will get a coefficient k. So each of these will be eliminated, after multiplication of $\mathcal{P}_k(x)$ by $\Leftrightarrow k$. Thus we find that

$$\mathcal{P}_{k+1}(x) = \mathcal{P}_k(x)\left(\sum_{i=1}^{r} x_i \Leftrightarrow k\right) = k\mathcal{P}_k(x) + (k+1)\sum_{i=1}^{N_{k+1}} \prod_{j \in K_i^{k+1}} x_j \Leftrightarrow k\mathcal{P}_k(x) = (k+1)\sum_{i=1}^{N_{k+1}} \prod_{j \in K_i^{k+1}} x_j.$$

Taking $k + 1 = \beta$ proves the theorem. □

In general, given an inequality $|a_j|^T \overline{x} \leq \overline{b}_j$, let the set $\mathcal{S}$ contain all the distinct values that the left hand side can take for feasible vectors $\overline{x}$. It may be noted that the number of distinct values can be exponential. The polynomial $P_j(x)$ can be obtained by computing

$$\overline{P}_j(x) = \prod_{s \in \mathcal{S}} (|a_j|^T \overline{x} \Leftrightarrow s), \tag{2.12}$$

and subsequently setting all coefficients to 1 and removing the terms that are *dominated* by others. A term $p_j(x)$ dominates a term $p_k(x)$ if the variables occurring in $p_j(x)$ are a subset of the variables occurring in $p_k(x)$. It is obvious that $p_j(x) = 0$ then implies that $p_k(x) = 0$, while the reverse is not necessarily true.

The reader may verify that all 'non–covers' get a coefficient '0' in the process of expanding (2.12), while it is impossible that any of the covers gets a negative coefficient. We suffice with giving an example.

Example 5 Consider $x_1 + 2x_2 + 3x_3 \leq 3$. The feasible vectors for this equation yield values $\mathcal{S} = \{0, 1, 2, 3\}$. Computing the product

$$(x_1 + 2x_2 + 3x_3)(x_1 + 2x_2 + 3x_3 \Leftrightarrow 1)(x_1 + 2x_2 + 3x_3 \Leftrightarrow 2)(x_1 + 2x_2 + 3x_3 \Leftrightarrow 3),$$

one obtains

$$\overline{P}(x) = x_1 x_3 + 6x_2 x_3 + 6x_1 x_2 x_3 = 0.$$

Obviously, this is a valid equation, but the last term is dominated by each of the first two. Consequently, $\{1, 2, 3\}$ is not a minimal cover of the given equality. So $P(x) = x_1 x_3 + x_2 x_3$. □

In general, it is recommendable to use the method of minimal covers to obtain a model of the form (NMP), since it is more efficient.

2.3 Transforming equality constraints

So far we have only considered the transformation of *in*equality constraints to a polynomial function. Now we consider the problem in which equality constraints occur as well.

$$(BP') \qquad \text{find } x \in \{0,1\}^m \text{ such that } Ax \leq b \text{ and } Bx = d.$$

Here $A \in I\!R^{n\times m}$, $B \in I\!R^{p\times m}$, $b \in I\!R^n$ and $d \in I\!R^p$. The obvious way to transform the set of equality constraints $Bx = d$ is by replacing it by $Bx \geq d$ and $Bx \leq d$ and subsequently applying the procedure for transforming inequality constraints.

There are cases however in which it might be beneficial to deal with the equality constraints in an alternative way. For example, in [14, 15, 13] the inequality constraints constitute a *quadratic* model, whereas adding the equality constraints to the polynomial as well would result in a higher order model. Due to the structure of the equality constraints, it is possible to not include them in the polynomial function, but still maintain the nice rounding property stated and proved in Theorem 2, i.e. that any fractional solution can be rounded to a binary solution without increasing the objective value.

For specific cases we can generalize Theorem 2. We first introduce some additional notation and make two assumptions. Let the sets E_t, $t = 1, \ldots, p$ be a partition of the index set $\{1, \ldots, m\}$ and let $\mathcal{P}(x)$ denote the polynomial function induced by the inequality constraints in (BP'). Finally, let x be an arbitrary point in the unit hyper cube such that $Bx = d$.

Assumption 1 *Each equality constraint in the set $Bx = d$ concerns only the variables in exactly one set E_t, $t \in \{1, \ldots, p\}$. Furthermore, B is* totally unimodular*.*

For the definition of total unimodularity see e.g. Schrijver [11].

Assumption 2 *For all $t \in \{1, \ldots, p\}$ the following holds. If all variables x_i, $i \notin E_t$ are fixed to their current value, while only the variables x_i, $i \in E_t$, remain free, the polynomial function $\mathcal{P}(x)$ reduces to a* linear *function in the variables x_i, $i \in E_t$.*

Under these assumptions we can prove the following theorem.

Theorem 4 *An arbitrary solution x with $0 \leq x \leq e$ and $Bx = d$ can be rounded to a binary solution $[x]$ such that $\mathcal{P}([x]) \leq \mathcal{P}(x)$.*

Proof: The solution $[x]$ can be constructed by subsequently solving p linear programs. Let x^{t-1} denote the intermediate solution, obtained after solving $t \Leftrightarrow 1$ linear programs. Linear program t, $1 \leq t \leq p$, is constructed by fixing all variables x_i^{t-1}, $i \notin E_t$, while letting the remaining variables free. Due to the unimodularity of the resulting constraint matrix (Assumption 1), there exists a binary optimal solution $\tilde{x}$ to this linear program [11]. The next solution x^t is obtained by substituting the binary values of $\tilde{x}$ in the corresponding entries of x^{t-1}. It is easily verified that in each step $\mathcal{P}(x^t) \leq \mathcal{P}(x^{t-1})$. □

Let us emphasize that the linear programs mentioned in this proof in general can be solved more efficiently than general LP problems by making use of their special structure [14].

It may be noted that the rounding scheme proposed in the proof of this theorem, and Theorem 2 can be viewed as approximation algorithms for solving (BP') c.q. (BP).

To finish this section we mention yet another technique to deal with the equality constraints. Let us consider equality constraint j, $b_j^T x = d_j$. Obviously, it holds that $(b_j^T x \Leftrightarrow d_j)^2 \geq 0$, where the squared expression equals zero if and only equality constraint j is satisfied. Using idempotency of the variables, this can be rewritten as

$$\sum_{i=1}^{m-1} \sum_{k=i+1}^{m} b_{ji} b_{jk} x_i x_k + \sum_{i=1}^{m} b_{ji}(b_{ji} \Leftrightarrow 2d_j) x_i + d_j^2.$$

In this way, the equality constraints can be added to the objective function $\mathcal{P}(x)$ yielding only bilinear and linear additional terms, while preserving the rounding property of Theorem 2.

3. Two specific applications

In this section we will discuss a number of specific applications of the transform we described in the previous section, and we will mention a number of algorithms that have been applied to solve them. It may be stressed that even though any problem of the form (BP) or (BP') can be transformed to a problem of the form (NMP), in general this will only be worthwhile if all or most constraints involved have one or at most few minimal covers. Moreover, the model is particularly suited to solve *feasibility* problems rather than problems in which some linear objective function needs to be optimized. The examples we discuss in this section all satisfy these desirable criteria.

3.1 The satisfiability problem

We consider instances of the satisfiability problem in *conjunctive normal form* (CNF).

Definition 3 *A formula ϕ is said to be in* conjunctive normal form *if*

$$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_n,$$

where '$\wedge$' denotes the binary conjunction operator. Each C_i is called a clause, *and each clause is the disjunction of a number of* literals*:*

$$C_j = \bigvee_{i \in R_j} x_i \vee \bigvee_{i \in S_j} \neg x_i,$$

where '$\vee$' is the binary disjunction operator, and '$\neg$' is the negation. The formula ϕ is satisfiable *if there is an assignment of the values* true *and* false *to the variables, such that each clause evaluates to* true*.*

In the following, we shall associate the value 1 with true, and the value 0 with false.

We can write each clause as a linear inequality. Consider the clause

$$C_j = \bigvee_{i \in R_j} x_i \vee \bigvee_{i \in S_j} \neg x_i,$$

we can rewrite C_j in the following way (see also Hooker [8]).

$$\sum_{i \in R_j} x_i + \sum_{i \in S_j} (1 \Leftrightarrow x_i) \geq 1. \tag{3.1}$$

Clearly, if (3.1) is satisfied for some binary x, at least one of the terms in the left hand side contributes a '1', so there is a $i \in R_j$ such that $x_i = 1$ (true) and/or a $i \in S_j$ such that $x_i = 0$ (false); therefore the clause C_j is true.

The next lemma shows that we can write an arbitrary clause C_j as a polynomial, similar to the one given in Lemma 2.

Lemma 7 *Given a clause C_j. Then we have the following implication. Let $0 \leq x \leq e$,*

$$p_j(x) = \prod_{i \in R_j} (1 \Leftrightarrow x_i) \prod_{i \in S_j} x_i = 0 \;\Rightarrow\; C_j \textit{ is satisfied by } [x].$$

Proof: Suppose we have written the clause as a linear inequality of the form (3.1). Obviously, this has a maximum violation $\vartheta_j = |R_j| + |S_j| \Leftrightarrow |R_j| \Leftrightarrow |S_j| + 1 = 1$, so Lemmas 2, 3 and 4 apply. □

It follows that for a given formula in CNF, we can straightforwardly formulate a nonconvex minimization problem as described in the previous section, since each inequality has exactly one minimal cover. So the satisfiability problem becomes:

$$(SAT) \qquad \begin{array}{ll} \min & \mathcal{P}_\phi(x) = \sum_{j=1}^{n} p_j(x) \\ \text{s.t.} & 0 \leq x \leq e. \end{array}$$

If we are given an optimal solution x of (SAT), then $[x]$ is also an optimal solution.

Note that the transformation we have introduced in the previous section, for the satisfiability problem boils down to the transformation used by Gu [5]. The same formulation is used in [2] to obtain bounds and algorithms for the maximum satisfiability problem.

As pointed out in the previous section, if we are given a contradictory formula ϕ, for a given truth value assignment x, $\mathcal{P}_\phi(x)$ is equal to the number of clauses that is not satisfied. We give an example.

Example 6 For a specific class of contradictory formulas, the transform has a nice property, namely that its associated polynomial $\mathcal{P}(x) \equiv 1$, so it is immediately clear that the formulas are not satisfiable, and that each truth value assignment satisfies all but one clause. Let the formula H_k be the formula containing all 2^k different clauses of k literals that can be constructed using the variables $x_1, \ldots, x_k$. Clearly, H_k is not satisfiable. Now we show that the polynomial $\mathcal{P}_{H_k}$ associated with H_k is equal to one, by induction on k. For $k = 1$ we have $H_1 = x_1 \wedge \neg x_1$, which implies that $\mathcal{P}_{H_1} = (1 \Leftrightarrow x_1) + x_1 = 1$. So assume that the claim holds for k. Consider H_{k+1}. Clearly, we have

$$H_{k+1} = (x_{k+1} \vee H_k) \wedge (\neg x_{k+1} \vee H_k).$$

This implies that

$$\mathcal{P}_{H_{k+1}} = (1 \Leftrightarrow x_{k+1})\mathcal{P}_{H_k} + x_{k+1}\mathcal{P}_{H_k}.$$

Since $\mathcal{P}_{H_k} \equiv 1$ we find that also $\mathcal{P}_{H_{k+1}} \equiv 1$. This concludes the example. □

Gu [5] proposes several algorithms for solving the 'global optimization version' (SAT) of the satisfiability problem. One of these algorithms makes in fact use of the same observation we used to prove Theorem 2. The function $\mathcal{P}_\phi$ is iteratively minimized by in each iteration choosing a variable and setting this to either 0 or 1, according to which gives the biggest improvement in terms of objective value. Note that this rule may be considered as a 'branching rule', and thus the global optimization algorithm as a branching algorithm (without backtracking). It can obviously also be applied within a backtracking algorithm, in order to obtain a complete algorithm. The algorithms Gu proposes prove to be quite effective. Using a SUN SPARC workstation 2, problems up to a size of 50000 clauses and 5000 variables are solved in a matter of (fractions of) seconds [5].

We conclude this subsection by mentioning that the approximation algorithm for satisfiability problems proposed by Johnson [9] also follows by using the transformation scheme in conjunction with the greedy rounding procedure. If we consider the polynomial representation of a pure k–SAT formula (i.e. all clauses have length exactly k) with n clauses and substitute $x = \frac{1}{2}e$, this solution has an objective value equal to $n2^{-k}$. Using the greedy rounding procedure a binary solution with objective value smaller than or equal to $n2^{-k}$ is obtained, implying that at least $(1 \Leftrightarrow 2^{-k})n$ clauses are satisfied. This is the same bound that Johnson obtains and in fact the algorithms coincide, although Johnsons does not make (explicit) use of a polynomial representation of the satisfiability problem.

Incidentally, the bound obtained is also equivalent to the expected quality of a solution obtained by applying the randomized algorithm in which each variable is set to 1 or 0, both with probability $\frac{1}{2}$. It has been shown recently for pure 3–SAT that no polynomial time algorithm with a better performance guarantee exists [7].

3.2 Node packing problems

In this section we consider node packing problems, and as an example we discuss the *graph coloring problem*. The *feasibility version* of the GCP can be formulated as follows: *Given an undirected graph $G = (V, E)$, with V the set of vertices and E the set of edges, and a set of colors C, find a coloring of the vertices of the graph such that any two* connected *vertices have different colors.*

The GCP can be modelled as follows. Defined are the binary decision variables:

$$x_{vc} = \begin{cases} 1 & \text{color } c \text{ is assigned to vertex } v, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall v \in V,\ \forall c \in C.$$

The following constraints must be satisfied. First, we have to assign exactly one color to each vertex:

$$\sum_{c \in C} x_{vc} = 1, \qquad \forall v \in V. \tag{3.2}$$

Second, two connected vertices may not get the same color:

$$x_{uc} + x_{vc} \leq 1, \qquad \forall (u, v) \in E,\ \forall c \in C. \tag{3.3}$$

Since the latter inequalities obviously have a maximum violation of 1, we straightforwardly derive the following model for the (GCP). Note that we choose not to include the equality constraints in the objective function, for reasons explained is Section 2.3.

$$(GCP) \qquad \begin{array}{lll} \min & \mathcal{P}_G(x) = \sum_{(u,v)\in E} \sum_{c\in C} x_{uc}x_{vc} & \\ \text{s.t.} & \sum_{c\in C} x_{vc} = 1, & \forall v \in V \\ & 0 \leq x \leq e. & \end{array}$$

In [14] an alternative expression for the objective function is derive Denoting the set of linear inequalities by $Ax \leq e$, then the objective function of (GCP) is given by $x^T[A^TA \Leftrightarrow \text{diag}(A^TA)]x$. Here $\text{diag}(A^TA)$ denotes the diagonal matrix with on its diagonal the diagonal entries of the matrix A^TA.

Let us briefly review two approximation algorithms that we have applied to models of the form (GCP): a *potential reduction algorithm* (for details see [14, 15]) and a *gradient descent algorithm* [13]. Both have the following structure.

Starting from a feasible interior point:

1. Find a descent direction for the objective function.
2. Compute the new iterate by finding the minimum of the objective function along the descent direction.
3. If the new iterate has an objective value smaller than one, a feasible coloring can be constructed by applying Theorem 4; else go to step 1.
4. If the algorithm is trapped in a local minimum, modify the problem by increasing the weights in $\mathcal{P}_G(x)$ of the edges $e = (u,v)$ for which u and v have the same color, and restart the process.

The main difference of the two algorithms is the way in which the descent direction is computed:

Potential Reduction: Solving (GCP) is equivalent to minimizing the following potential function:

$$\psi(x;w) = \mathcal{P}_G(x) \Leftrightarrow \sum_{i=1}^{2m} w_i \log s_i,$$

where $w \in I\!R^{2m}$ is a positive weight vector, and the values s_i are the slacks of the constraints $0 \leq x \leq e$. In each iteration, a *quadratic* approximation of ψ is minimized over an ellipsoid centered at the current solution vector to obtain a descent direction. This requires factorizing a (sparse) $m \times m$ matrix at least once, which has a worst case complexity of $\mathcal{O}(m^3)$.

Gradient Descent: The descent direction Δx is computed by minimizing the gradient at the current iterate under the constraint that $x + \Delta x$ remains feasible. This can be done in $\mathcal{O}(m \log m)$ time.

We conclude that the computation of the descent direction for the gradient descent algorithm is done considerably more efficiently. To increase the solution speed of the potential reduction algorithm, in each iteration the current iterate is rounded to a binary solution; as soon as a feasible coloring has been found, the algorithm stops.

Computational results To give an indication of the effectiveness of the algorithms we have mentioned above, we present some computational results on a number of randomly generated instances of the graph coloring problem. The same instances were used in [14]. The optimal solutions of these instances are known; when running the algorithm the number of colors available was set equal to this optimal solution. Thus the number of variables for each instance is equal to $|V||C|$. Each instances was solved a hundred times with both algorithms, from different starting points. In Table 1 the minimal, mean and maximal solution times are reported. The algorithms succeeded in finding a feasible (optimal) coloring in all cases. It is emphasized that the computational results we obtained using an *ad hoc* MATLAB™ implementation (although it does use some FORTRAN subroutines provided by Zhang [16]); the computation times could be vastly improved by using a low level language implementation. All tests were run on a HP9000/720 workstation, 144 Mb, 50 mHz. In our computational tests, the potential reduction algorithm never got stuck in a local

		MIN		MEAN		MAX	
G$\|V\|.\|C\|$	$\|E\|$	PR	GD	PR	GD	PR	GD
G50.7	161	.70	.95	1.76	1.23	4.85	1.48
G50.10	229	1.15	1.40	1.99	1.78	6.76	4.43
G50.12	237	1.52	1.15	1.80	1.52	4.70	1.94
G50.15	224	1.93	1.58	2.42	1.69	8.41	1.87
G50.18	288	3.82	1.79	3.98	1.91	7.67	2.08
G100.5	251	3.89	1.67	9.02	2.62	25.28	10.94
G100.8	379	2.16	2.10	8.42	3.02	23.62	3.80
G100.12	484	6.32	2.13	7.32	2.85	13.66	3.57
G100.16	449	9.82	3.31	10.75	4.21	17.20	6.14
G150.6	438	5.76	5.22	17.51	7.05	42.67	19.06
G150.9	612	4.76	5.47	14.28	6.97	66.28	8.84
G150.16	663	25.14	5.17	29.61	6.77	82.83	8.03
G200.5	504	11.96	5.07	35.70	8.45	81.42	15.30
G200.10	954	10.10	9.52	39.19	11.67	109.24	17.33
G200.14	952	39.44	7.78	51.92	9.72	127.50	12.24
G300.10	1383	32.12	17.63	91.39	20.97	194.16	20.62
G500.8	1891	248.40	27.39	656.03	39.19	1477.14	97.56

Table 1: Computational results on a number of undirected graphs $G = (V, E)$. Solution times are in seconds.

minimum, while the gradient descent algorithm over all 1700 runs converged 57 times to a local minimum (after which it proceeded to converge to global minima). In general, the gradient descent algorithm appears to be a bit faster and more robust than the potential reduction algorithm.

4. Concluding remarks

In this paper we have described a way to transform a combinatorial optimization problem to a continuous nonlinear nonconvex optimization problem over the unit hyper cube. We have shown that for specific applications this transformation yields strong models that can be efficiently solved by appropriate approximation algorithms. This observation is also confirmed by results we obtained on another node packing problem, namely the *frequency assignment problem* [15, 13]. The algorithms based on the polynomial transform are the most effective when the instances of the feasibility problem under consideration have a 'reasonably large' number of feasible solutions. This conclusion can also be drawn from the results presented by Gu [5]. Future research includes the application of the proposed transformation and algorithms to other types of combinatorial optimization problems.

References

1. P. Barth. Linear 0–1 inequalities and extended clauses. Technical Report MPI–I–94–216, Max–Planck–Institut für Informatik, Im Stadtwald, D66123 Saarbrücken, Germany, 1994.

2. E. Boros and A. Prékopa. Probabilistic bounds and algorithms for the maximum satisfiability problem. *Annals of Operations Research*, 21:109–126, 1989.

3. V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337, 1973.

4. F. Granot and P.L. Hammer. On the use of boolean functions in 0-1 programming. *Methods of Operations Research*, 12:154–184, 1971.

5. J. Gu. Global optimization for satisfiability (SAT) problem. *IEEE Transactions on Knowledge and Data Engineering*, 6(3):361–381, 1994.

6. P.L. Hammer and S. Rudeanu. *Boolean Methods in Operations Research and Related Areas.* Springer-Verlag, 1968.

7. J. Håstad. Some optimal in–approximability results. In *STOC '96*, 1996.

8. J.N. Hooker. Generalized resolution and cutting planes. *Annals of Operations Research*, 12:217–239, 1988.

9. D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

10. G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* Wiley-Interscience, 1988.

11. A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, New York, 1986.

12. J.P. Warners. A linear–time transformation of linear inequalities into conjunctive normal form. Technical Report CS-R9631, Centre for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands, 1996.

13. J.P. Warners. A nonlinear approach to a class of combinatorial optimization problems. Technical Report Technical Report 96–119, Delft University of Technology, Delft, The Netherlands, 1996. To appear in *Statistica Neerlandica.*

14. J.P. Warners, T. Terlaky, C. Roos, and B. Jansen. Potential reduction algorithms for structured combinatorial optimization problems. Technical Report 95-88, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1995. To appear in *Operations Research Letters.*

15. J.P. Warners, T. Terlaky, C. Roos, and B. Jansen. A potential reduction approach to the

frequency assignment problem. Technical Report 95-98, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1995. To appear in *Discrete Applied Mathematics.*

16. Y. Zhang. Solving large–scale linear programs by interior point methods under the MATLAB environment. Technical Report TR96–01, Department of Mathematics and Statistics, University of Baltimore County, Baltimore, Maryland 21222–5398, 1996.