An Interior Point Approach to the Maximum Independent Set Problem in Dense Random Graphs[*]

Narendra Karmarkar, Mauricio G.C. Resende and K.G. Ramakrishnan

Mathematical Sciences Research Center

AT&T Bell Laboratories

Murray Hill, NJ 07974 - USA

Abstract

We present an interior point approach to the zero-one integer programming feasibility problem based on the minimization of an appropriate potential function. Given a polytope defined by a set of linear inequalities, this procedure generates a sequence of strict interior points of this polytope, such that each consecutive point reduces the value of the potential function. An integer solution (not necessarily feasible) is generated at each iteration by a rounding scheme. The direction used to determine the new iterate is computed by solving a nonconvex quadratic program on an ellipsoid. We illustrate the approach by considering a class of difficult NP-complete problems: finding a maximum independent set of a dense random graph. Some implementation details are discussed and preliminary computational results are presented. We solve several large independent set problems in graphs having up to 1000 vertices and over 250,000 edges.

Key words: Integer programming, interior point method, maximum independent set, dense random graphs.

1. Introduction

In this paper we consider the following integer programming problem:
INTEGER PROGRAMMING: Let $B \in \Re^{m\times n}$ and $b \in \Re^n$. Find $w \in \Re^m$ such that:

$$B^T w \leq b \qquad (1)$$

[*] Talk given by M.G.C. Resende at the IX International Conference of the Chilean Computer Science Society and XV Latin American Conference on Informatics, Santiago, Chile, 11 July 1989.

$$w_i \quad = \quad \{-1,1\},\ i=1,\dots,m. \tag{2}$$

The more common form of integer programming, where variables x_i, $i=1,\dots,m$, take on (0,1) values, can be converted to the above form with the change of variables

$$x_i = \frac{1+w_i}{2},\ i=1,\dots,m.$$

We propose an interior point approach to solve (1-2), i.e. a heuristic that generates a sequence of points $(w^0, w^1, \dots, w^k, \dots)$ where for all $k=0,1,\dots$

$$w^k \in \left\{ w \in \Re^m | B^T w < b;\ \ -e < w < e \right\},$$

where $e^T = (1,\dots,1)$. In practice, this sequence often converges to a point from which one can jump to a (-1,1) integer solution to (1-2). No guarantee can be made as to whether the heuristic will be successful, but we provide several instances of large maximum independent set problems in dense random graphs where it succeeds in providing optimal solutions.

To simplify notation, let I denote an $m \times m$ identity matrix,

$$A = \left[B \vdots I \vdots -I \right]$$

and

$$c = \begin{bmatrix} b \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

and let

$$\mathcal{I} = \left\{ w \in \Re^m \mid A^T w \le c \ \text{ and } \ w_i = \{-1,1\} \right\}.$$

With this notation, INTEGER PROGRAMMING can be restated as: Find $w \in \mathcal{I}$.

We make the following assumptions:

Assumption 1.1 $\mathcal{I} \neq \emptyset$.
Assumption 1.2 A^T has full rank.

Let

$$\mathcal{L} = \left\{ w \in \Re^m \mid A^T w \le c \right\}$$

and consider the linear programming relaxation of (1-2), i.e. find $w \in \mathcal{L}$. One way of selecting (-1,1) integer solutions over fractional solutions in linear programming is to introduce the quadratic objective function

$$\text{maximize } w^T w = \sum_{i=1}^{m} w_i^2$$

and solve the NP-complete [14] nonconvex quadratic programming problem (QUADRATIC PROGRAMMING)

$$\text{maximize } w^T w \tag{3}$$

$$\text{subject to : } A^T w \leq c \tag{4}$$

Note that an upper bound on the objective function (3) is m. The following proposition establishes the relationship between QUADRATIC PROGRAMMING and INTEGER PROGRAMMING.

Proposition 1.1 *Let $w \in \mathcal{L}$. Then $w \in \mathcal{I} \iff w^T w = m$.*
Proof: ($\Longrightarrow$) Clearly, if $w \in \mathcal{I}$ then $w \in \mathcal{L}$ and $w_i = \{-1, 1\}$. Hence $w^T w = m$.
($\Longleftarrow$) By assumption $w \in \mathcal{L}$. If $w^T w = m$ then $w_i = \{-1, 1\}$, $\forall i = 1, \ldots, m$ and therefore $w \in \mathcal{I}$. □

In the remainder of this paper we discuss how to approximate a polytope with an inscribed ellipsoid and optimize a potential function closely related to the objective function of QUADRATIC PROGRAMMING. We then present an algorithm in pseudo-code level for this optimization. Finally we illustrate how this algorithm can be applied to a difficult combinatorial problem: Finding an independent k-set of a dense random graph.

2. Nonconvex Quadratic Programming

Karmarkar [10] suggested an interior point approach to NP-complete problems. In this section we provide an algorithm in pseudo-code form detailing that approach. We also provide details on applying this algorithm to INTEGER PROGRAMMING. We wish to solve (3-4). Let

$$w^0 \in \mathcal{L}_s = \left\{ w \in \Re^m \mid A^T w < c \right\}$$

be a given initial interior point. The algorithm generates a sequence of interior points of $\mathcal{L}$. Let $w^k \in \mathcal{L}_s$ be the k-th iterate. Around w^k we construct a quadratic approximation (truncated Taylor series of order 2) of the potential function

$$\varphi(w) = \log \sqrt{m - w^T w} - \frac{1}{n} \sum_{k=1}^{n} \log d_k(w)$$

where

$$d_k(w) = c_k - a_k^T w, \ \ \forall k = 1, \ldots, n.$$

Let $D = \text{diag}(d_1(w), \ldots, d_n(w))$, $e = (1, \ldots, 1)$, $f_0 = m - w^T w$ and C be a constant. The quadratic approximation of $\varphi(w)$ around w^k is given by

$$Q(w) = \frac{1}{2}(w - w^k)^T H (w - w^k) + h^T (w - w^k) + C \tag{5}$$

where the Hessian

$$H = -\frac{2}{f_0}I - \frac{4}{f_0^2}ww^T + \frac{1}{n}AD^{-2}A^T \tag{6}$$

and the gradient

$$h = -\frac{1}{f_0}w + \frac{1}{n}AD^{-1}e. \tag{7}$$

Minimizing (5) subject to $A^T w \leq c$ is NP-complete. However, if the polytope is substituted by an inscribed ellipsoid, the resulting approximate problem is easy. Ye [17] has proposed a polynomial time algorithm for nonconvex quadratic programming on an ellipsoid that differs from the approach to be presented here. Our approach is similar to the classical Levenberg-Marquardt methods, [11], [12], first suggested in the context of nonlinear least squares.

The following proposition describes a suitable inscribed ellipsoid.

Proposition 2.1 *Consider the polytope defined as*

$$\mathcal{L} = \left\{ w \in \Re^m | A^T w \leq c \right\}$$

and let $w^k \in \mathcal{L}_s = \mathrm{int}(\mathcal{L})$ be an interior point of $\mathcal{L}$. Consider the ellipsoid

$$\mathcal{E}(r) = \left\{ w \in \Re^m | (w - w^k)^T AD^{-2}A^T(w - w^k) \leq r^2 \right\}.$$

Then for $r \leq 1$, $\mathcal{E}(r) \subset \mathcal{L}$, i.e. $\mathcal{E}(r)$ is an inscribed ellipsoid in $\mathcal{L}$.

Proof: It is sufficient to prove case when $r = 1$, since $\mathcal{E}(r) \subset \mathcal{E}(1)$, for $0 \leq r < 1$. Assume $y \in \mathcal{E}(1)$. Then

$$(y - w^k)^T AD^{-2}A^T(y - w^k) \leq 1$$

and consequently

$$D^{-1}A^T(y - w^k) \leq 1.$$

Denoting the i-th row of A^T by $a_{i\cdot}^T$, we have

$$\begin{aligned} \frac{1}{c_i - a_{i\cdot}^T w^k} a_{i\cdot}^T (y - w^k) &\leq 1, \quad \forall i = 1, \ldots, n \\ a_{i\cdot}^T (y - w^k) &\leq c_i - a_{i\cdot}^T w^k, \quad \forall i = 1, \ldots, n \\ a_{i\cdot}^T y &\leq c_i, \quad \forall i = 1, \ldots, n. \end{aligned}$$

Therefore $A^T y \leq c$. Consequently $y \in \mathcal{L}$. $\square$

Substituting the polytope by the appropriate ellipsoid and letting $\Delta w \equiv w - w^k$ results in the following optimization problem

$$\text{minimize } \frac{1}{2}(\Delta w)^T H \Delta w + h^T \Delta w \tag{8}$$

$$\text{subject to} : (\Delta w)^T AD^{-2}A^T(\Delta w) \leq r^2. \tag{9}$$

The optimal solution to (8-9), Δw^*, is a descent direction of $Q(w)$ from w^k. For a given radius $r > 0$, the value of the original potential function $\varphi(w)$ may increase by moving in the direction Δw^*, because of the higher order terms ignored in the approximation. It can be easily verified, however, that if the radius is decreased sufficiently, the potential function will improve by moving in the new Δw^* direction. In practice, we shall say a *local minimum* has been found if the radius must be reduced below a tolerance ϵ to achieve a reduction in the value of the potential function.

The following theorem, proved in [10] characterizes the optimal solution of (8-9). In place of the ellipsoid

$$\left\{x \in \Re^m \mid x^T AD^{-2}A^T x \leq r^2\right\}$$

the theorem considers the sphere

$$\left\{x \in \Re^m \mid x^T x \leq r^2\right\} \tag{10}$$

without loss of generality since $AD^{-2}A^T$ is, by assumption, positive definite and can be converted to (10) by means of a nonsingular linear transformation of the space.

Theorem 2.2 *Consider the following problem:*

$$\textit{minimize } \frac{1}{2}x^T Hx + h^T x \tag{11}$$

$$\text{subject to} : x^T x \leq r^2 \tag{12}$$

where $H \in \Re^{m\times m}$ is symmetric and indefinite, $x, h \in \Re^m$ and $0 < r \in \Re$. Let $u_1, \ldots, u_m$ denote a full set of orthonormal eigenvectors spanning $\Re^m$ and let $\lambda_1, \ldots, \lambda_m$ be the corresponding eigenvalues ordered so that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{m-1} \leq \lambda_m$. Denote $0 > \lambda_{min} = min\{\lambda_1, \ldots, \lambda_m\}$ and u_{min} the corresponding eigenvector. To describe the solution to (11-12) we shall consider two cases:

Case 1: Assume $\sum_{i=1}^{k}(h^T u_i)^2 > 0$. Let the scalar $\lambda \in (-\infty, \lambda_{min})$ and consider the parametric family of vectors

$$x(\lambda) = -\sum_{i=1}^{m} \frac{(h^T u_i)u_i}{\lambda_i - \lambda}.$$

For any $r > 0$, denote by $\lambda(r)$ the unique solution of the equation $(x(\lambda))^T x(\lambda) = r^2$ in λ. Then $x(\lambda(r))$ is the unique optimal solution of (11-12).

Case 2: Assume $h^T u_i = 0, \forall i = 1, \ldots, k$. Let the scalar $\lambda \in (-\infty, \lambda_{min})$ and consider the parametric family of vectors

$$x(\lambda) = -\sum_{i=k+1}^{m} \frac{(h^T u_i)u_i}{\lambda_i - \lambda}.$$

Let

$$r_{max} = \|x(\lambda_{min})\|_2.$$

If $r < r_{max}$ *then for any* $0 < r < r_{max}$*, denote by* $\lambda(r)$ *the unique solution of the equation* $(x(\lambda))^T x(\lambda) = r^2$ *in* λ*. Then* $x(\lambda(r))$ *is the unique optimal solution of (11-12).*
If $r \geq r_{max}$*, then let* $\alpha_1, \alpha_2, \ldots, \alpha_k$ *be any real scalars such that*

$$\sum_{i=1}^{k} \alpha_i^2 = r^2 - r_{max}^2.$$

Then

$$x = \sum_{i}^{k} \alpha_i u_i - \sum_{i=k+1}^{m} \frac{(h^T u_i) u_i}{(\lambda_i - \lambda_{min})}$$

is an optimal solution of (11-12). Since the choice of α_i*'s was arbitrary, this solution is not unique.*

For the purpose of this paper, the key result of Theorem 2.2 is the existence of a unique optimal solution to (11-12) if $r < r_{max}$. The proof of Theorem 2.2 relies on a lemma that is used to develop the algorithm that is described in this paper.

Lemma 2.3 *Let the length of* $x(\lambda)$ *be*

$$l\left(x(\lambda)\right) \equiv \|x(\lambda)\|_2^2 = \left(x(\lambda)\right)^T x(\lambda).$$

Part 1: Assume $\sum_{i=1}^{k} (h^T u_i)^2 > 0$*. Consider the parametric family of vectors*

$$x(\lambda) = -\sum_{i=1}^{m} \frac{(h^T u_i) u_i}{\lambda_i - \lambda},$$

for $\lambda \in (-\infty, \lambda_{min})$*. Then* $l\left(x(\lambda)\right)$ *is monotonically increasing in* λ *in the interval* $\lambda \in (-\infty, \lambda_{min})$*.*
Part 2: Assume $h^T u_i = 0, \forall i = 1, \ldots, k$ *and consider the parametric family of vectors*

$$x(\lambda) = -\sum_{i=k+1}^{m} \frac{(h^T u_i) u_i}{\lambda_i - \lambda},$$

for $\lambda \in (-\infty, \lambda_{min})$*. Furthermore, assume*

$$r < \|x(\lambda_{min})\|_2.$$

Then $l\left(x(\lambda)\right)$ *is monotonically increasing in* λ *in the interval* $\lambda \in (-\infty, \lambda_{min})$*.*

Proof: (Part 1) Since

$$x(\lambda) = -\sum_{i=1}^{m} \frac{(h^T u_i)u_i}{\lambda_i - \lambda},$$

then

$$l\,(x(\lambda)) = -\sum_{i=1}^{m} \frac{(h^T u_i)^2}{(\lambda_i - \lambda)^2}.$$

In the range $\lambda \in (-\infty, \lambda_{min})$ each of the terms $1/(\lambda_i - \lambda)^2$ $(i = 1, \ldots, m)$ is a strictly monotonically increasing function of λ, each of the numerators $(h^T u_i)^2$ $(i = 1, \ldots, m)$ is nonnegative and at least one of the numerators is strictly positive. Hence $l\,(x(\lambda))$ is monotonically increasing in λ. $\square$

(Part 2) As in Part 1, since

$$x(\lambda) = -\sum_{i=k+1}^{m} \frac{(h^T u_i)u_i}{\lambda_i - \lambda},$$

then

$$l\,(x(\lambda)) = -\sum_{i=k+1}^{m} \frac{(h^T u_i)^2}{(\lambda_i - \lambda)^2}.$$

In the range $\lambda \in (-\infty, \lambda_{min})$ each of the terms $1/(\lambda_i - \lambda)^2$ $(i = k+1, \ldots, m)$ is a strictly monotonically increasing function of λ. Since $r > 0$, then $\|x(\lambda_{min})\|_2 > 0$ and therefore $\exists\ i > k$ such that $h^T u_i \neq 0$. Hence $|h^T u_i| > 0$ and consequently $l\,(x(\lambda))$ is monotonically increasing in λ. $\square$

Theorem 2.2 suggests an approach to solve the nonconvex quadratic programming problem (3-4). At each iteration a quadratic approximation of the potential function $\varphi(w)$ around the iterate w^k is optimized on an ellipsoid centered at w^k and that inscribes the polytope. Either a descent direction Δw^* of $\varphi(w)$ is produced by this optimization or w^k is said to be a local minimum. A new iterate w^{k+1} is computed such that $\varphi(w^{k+1}) < \varphi(w^k)$ by moving from w^k in the direction Δw^*. At each iteration the current iterate w^k is rounded off to the nearest (-1,1) vertex: $\tilde{w}^k = (\pm 1, \ldots, \pm 1)$. If $\tilde{w}^k$ is such that $A^T \tilde{w}^k \leq c$ then $\tilde{w}^k$ is a global optimal solution of QUADRATIC PROGRAMMING. If a local minimum is found, several strategies can be considered. In one approach, the problem can be modified (by adding a cut, for example) and the algorithm is applied to the new problem. Another strategy is to use the information generated by the local minimum to solve a smaller problem. These local minimum strategies are problem specific. In this paper we shall describe an approach that we have used when solving independent k-set problems.

3. The Algorithm

In this section we describe an algorithm to solve INTEGER PROGRAMMING based on nonconvex quadratic programming. As discussed previously, the algorithm considers the optimization problem

$$\text{minimize } \frac{1}{2}(\Delta w)^T H \Delta w + h^T \Delta w \tag{13}$$

$$\text{subject to : } (\Delta w)^T AD^{-2}A^T \Delta w \le r^2 \le 1 \tag{14}$$

to produce a descent direction Δw^* for the potential function $\varphi(w)$. A solution $\Delta w^* \in \Re^m$ to (13-14) is optimal if and only if there exists $\mu \ge 0$ such that:

$$\Delta w^* \left(H + \mu AD^{-2}A^T\right) = -h \tag{15}$$

$$\mu \left((\Delta w^*)^T AD^{-2}A^T \Delta w^* - r^2\right) = 0 \tag{16}$$

$$H + \mu AD^{-2}A^T \text{is positive definite.} \tag{17}$$

With the change of variables $\gamma = 1/(\mu + 1/n)$ and substituting (6) and (7) into (15) we obtain an expression for Δw^* satisfying (15):

$$\Delta w^* = -\left(AD^{-2}A^T - \frac{4\gamma}{f_0^2} w^k {w^k}^T - \frac{2\gamma}{f_0} I\right)^{-1} \gamma \left(-\frac{1}{f_0} w^k + \frac{1}{n} AD^{-1}e\right) \tag{18}$$

Note that r does not appear in (18). However, (18) is not defined for all values of r. Theorem 2.2 guarantees that if the radius r of the ellipsoid (14) is kept within a certain bound, then there exists an interval $0 \le \gamma \le \gamma_{max}$ such that

$$AD^{-2}A^T - \frac{4\gamma}{f_0^2} w^k {w^k}^T - \frac{2\gamma}{f_0} I$$

is nonsingular. The following proposition establishes that for γ small enough Δw^* is a descent direction of $\varphi(w)$, i.e. $h^T \Delta w^* < 0$.

Proposition 3.1 *There exists $\gamma > 0$ such that the direction Δw^*, given in (18), is a descent direction of $\varphi(w)$.*

Proof:

$$\begin{aligned}
\Delta w^* &= -\left(AD^{-2}A^T - \frac{4\gamma}{f_0^2} w^k {w^k}^T - \frac{2\gamma}{f_0} I\right)^{-1} \gamma \left(-\frac{1}{f_0} w^k + \frac{1}{n} AD^{-1}e\right) \\
&= -\left[AD^{-2}A^T \left\{I - \gamma (AD^{-2}A^T)^{-1} \left(-\frac{4}{f_0^2} w^k {w^k}^T - \frac{2}{f_0} I\right)\right\}\right]^{-1} \times \\
&\quad \gamma \left(-\frac{1}{f_0} w^k + \frac{1}{n} AD^{-1}e\right)
\end{aligned}$$

$$
\begin{aligned}
&= -\gamma \left[I + \gamma (AD^{-2}A^T)^{-1} \left(\frac{4}{f_0^2} w^k {w^k}^T + \frac{2}{f_0} I \right) \right]^{-1} (AD^{-2}A^T)^{-1} \times \\
&\quad \left(-\frac{1}{f_0} w^k + \frac{1}{n} AD^{-1} e \right) \\
&= \gamma \left[I + \gamma (AD^{-2}A^T)^{-1} \left(\frac{4}{f_0^2} w^k {w^k}^T + \frac{2}{f_0} I \right) \right]^{-1} (AD^{-2}A^T)^{-1} (-h)
\end{aligned}
$$

Let $\gamma = \epsilon > 0$ and consider $\lim_{\epsilon \to 0^+} h^T \Delta w^*$. We have

$$\lim_{\epsilon \to 0^+} \Delta w^* = \epsilon \, (AD^{-2}A^T)^{-1}(-h)$$

and therefore

$$\lim_{\epsilon \to 0^+} h^T \Delta w^* = -\epsilon \, h^T (AD^{-2}A^T)^{-1} h.$$

Since, by assumption, $\epsilon > 0$ and $h^T (AD^{-2}A^T)^{-1} h > 0$ then

$$\lim_{\epsilon \to 0^+} h^T \Delta w^* < 0. \ \Box$$

We now propose an algorithm to find a solution of (13-14) satisfying condition (15-17) and show how to incorporate this algorithm into an algorithm to solve INTEGER PROGRAMMING. Each iteration of this algorithm is comprised of two tasks. To simplify notation, let

$$H_c = AD^{-2}A^T$$

$$H_o = -\frac{4}{f_0^2} w^k {w^k}^T - \frac{2}{f_0} I$$

Given the current iterate w^k we first seek a value of γ such that

$$M = H_c + \gamma H_o$$

is nonsingular. This can be done by binary search, as we will see shortly. Once such a parameter γ is found, the linear system

$$M \Delta w^* = \gamma h \tag{19}$$

is solved for $\Delta w^* \equiv \Delta w^*(\gamma)$. Lemma 2.3 guarantees that the length $l(\Delta w^*(\gamma))$ is a monotonically increasing function of γ in the interval $0 \le \gamma \le \gamma_{max}$.

Optimality condition (16) implies that $r = \sqrt{l(\Delta w^*(\gamma))}$ if $\mu > 0$. Small lengths result in small changes in the potential function, since r is small and the optimal solution lies on the surface of the ellipsoid. A length that is too large may not correspond to an optimal solution of (13-14), since this may require $r > 1$. We maintain an interval $(\underline{l}, \overline{l})$ that we call the *acceptable length region* and accept a length $l(\Delta w^*(\gamma))$ if $\underline{l} \leq l(\Delta w^*(\gamma)) \leq \overline{l}$. If $l(\Delta w^*(\gamma)) < \underline{l}$, γ is increased and (19) is resolved with the new M matrix and h vector. On the other hand, if $l(\Delta w^*(\gamma)) > \overline{l}$, γ is reduced and (19) is resolved. Once an acceptable length is produced the new iterate w^{k+1} is computed by moving in direction $\Delta w^*(\gamma)$ from w^k with a step size $\alpha < 1$,

$$w^{k+1} = w^k + \alpha \Delta w^*(\gamma).$$

Pseudo code 3.1 details **procedure** `ip`, the integer programming algorithm that makes of **procedure** `descent_direction` to optimize (13-14) producing the descent direction.

```
procedure ip($A, c, \gamma_0, \underline{l}_0, \overline{l}_0$)
1   $k := 0$;  $\gamma := \gamma_0$;  $\underline{l} := \underline{l}_0$;  $\overline{l} := \overline{l}_0$;  $K := 0$;
2   $w^k :=$ get_start_point($A, c$);
3   $\tilde{w}^k :=$ round_off($w^k$);
4   do $A^T \tilde{w}^k \not< c \rightarrow$
5       $\Delta w^* :=$ descent_direction($\gamma, w^k, \underline{l}, \overline{l}$);
6       do $\varphi(w^k + \alpha \Delta w^*) \geq \varphi(w^k)$ and $\overline{l} > \epsilon \rightarrow$
7           $\overline{l} := \overline{l}/l_r$;
8           $\Delta w^* :=$ descent_direction($\gamma, w^k, \underline{l}, \overline{l}$)
9       od;
10      if $\varphi(w^k + \alpha \Delta w^*) < \varphi(w^k) \rightarrow$
11          $w^{k+1} := w^k + \alpha \Delta w^*$;
12          $\tilde{w}^{k+1} :=$ round_off($w^{k+1}$);
13          $k := k + 1$
14      fi;
15      if $\overline{l} \leq \epsilon \rightarrow$
16          $A :=$ new_matrix($A$);  $c :=$ new_rhs($c$);
17          $k := 0$;  $\gamma := \gamma_0$;  $\underline{l} := \underline{l}_0$;  $\overline{l} := \overline{l}_0$;  $K := K + 1$;
18          $w^k :=$ get_start_point($A, c$);
19          $\tilde{w}^k :=$ round_off($w^k$)
20      fi
21  od
end ip;
```

Pseudo-Code 3.1 - The `ip` Algorithm

Procedure `ip` takes as input the A matrix, the c right hand side vector, an initial guess γ_0 of parameter γ and initial lower and upper bounds on the acceptable length, $\underline{l}_0$ and $\bar{l}_0$, respectively. In the first line of `ip` the minor iteration counter (k), lower and upper bounds on the acceptable length region $(\underline{l}, \bar{l})$ and major iteration counter (K) are initialized. In line 2, `get_start_point` returns a strict interior point of the polytope under consideration, i.e. $w^k \in \mathcal{L}_s$. In many situations this is a trivial task. In others, a phase I interior point linear programming algorithm may be required. In line 3, the array w^k is rounded off to the nearest ± 1 vertex by procedure `round_off` and the result is placed in array $\tilde{w}^k$.

The algorithm iterates in the loop between lines 4 and 21, terminating only when a feasible (-1,1) integer solution $\tilde{w}^k$ is found. The algorithm can be easily modified to also terminate if a local minimum is found by deleting lines 15-20 and changing the termination criterion in line 4 to include the negation of the condition in line 15. At each iteration, a descent direction of the potential function $\varphi(w)$ is produced in lines 5 through 9. In line 5 the optimization (13-14) is realized. Because of higher order terms the direction returned by `descent_direction` may not be a descent direction for $\varphi(w)$. Loop 6-9 is repeated, reducing the upper bound of the acceptable length region $\bar{l}$ by l_r, until an improving direction for the potential function is produced or the largest acceptable length falls below a given tolerance ϵ. These two cases are treated in lines 10-14 and 15-20, respectively.

In the case that the direction produced is a descent for $\varphi(w)$, a new point w^{k+1} is defined (in line 11) by moving from the current iterate w^k in the direction Δw^* by a step length $\alpha < 1$. In line 12 this new point is rounded off and set to $\tilde{w}^{k+1}$.

If in loop 6-9 the largest acceptable length has fallen below ϵ we say the algorithm has converged to a local (not global) minimum. A new problem is defined in line 16 and the algorithm is restarted in lines 17-19. We later discuss how to define a new problem, in the context of independent k-set of dense random graphs.

Pseudo-code 3.2 details procedure `descent_direction`, where (13-14) is optimized.

```
procedure descent_direction($\gamma, w^k, \underline{l}, \overline{l}$)
    $l := \infty$;  $LD_{key} :=$ false;  $\overline{\gamma}_{key} :=$ false;  $\underline{\gamma}_{key} :=$ false;
    do $l > \overline{l}$ or ($l < \underline{l}$ and $LD_{key} =$ false) $\rightarrow$
        $M := H_c + \gamma H_o$;
        do $M$ is singular $\rightarrow$
            $\gamma := \gamma / \gamma_r$;  $LD_{key} :=$ true;
            $M := H_c + \gamma H_o$;  $b := \gamma h$
        od;
        $\Delta w^* := M^{-1} b$;  $l := (\Delta w^*)^T A D^{-2} A^T \Delta w^*$;
        if $l < \underline{l}$ and $LD_{key} =$ false $\rightarrow$
            $\underline{\gamma} := \gamma$;  $\underline{\gamma}_{key} :=$ true;
            if $\overline{\gamma}_{key} =$ true $\rightarrow \gamma := \sqrt{\underline{\gamma}\overline{\gamma}}$ fi;
            if $\overline{\gamma}_{key} =$ false $\rightarrow \gamma := \gamma \cdot \gamma_r$ fi
        fi;
        if $l > \overline{l} \rightarrow$
            $\overline{\gamma} := \gamma$;  $\overline{\gamma}_{key} :=$ true;
            if $\underline{\gamma}_{key} =$ true $\rightarrow \gamma := \sqrt{\underline{\gamma}\overline{\gamma}}$ fi;
            if $\underline{\gamma}_{key} =$ false $\rightarrow \gamma := \gamma / \gamma_r$ fi
        fi
    od;
    do $l < \underline{l}$ and $LD_{key} =$ true $\rightarrow \underline{l} := \underline{l} / l_r$ od;
    return($\Delta w^*$)
end descent_direction;
```

Pseudo-Code 3.2 - The `descent_direction` Algorithm

As input, procedure `descent_direction` is given a guess for parameter γ, the current iterate w^k around which the inscribing ellipsoid is to be constructed and the current acceptable length region defined by $\underline{l}$ and $\overline{l}$. The value of γ passed to `descent_direction` at minor iteration k of `ip` is the value returned by `descent_direction` at minor iteration $k-1$. It returns a descent direction Δw^* of the quadratic approximation of the potential function $Q(w)$ from w^k, the next guess for parameter γ and the current lower bound of the acceptable length region, $\underline{l}$.

In line 1 the length l is set to a large number and several logical keys are initialized: LD_{key} is **true** if a linear dependency in the rows of M is found during the solution of the linear system (19) and is **false** otherwise; $\overline{\gamma}_{key}$ ($\underline{\gamma}_{key}$) is **true** if an upper (lower) bound for an acceptable γ has been found and **false** otherwise.

The nonconvex quadratic optimization on the ellipsoid is carried out in the loop going from line 2 to 19. The loop is repeated until either a length l is found such that $\underline{l} \leq l \leq \overline{l}$ or $l \leq \underline{l}$ due to a linear dependency found during the solution of (19) (i.e. $LD_{key} = \textbf{true}$). Lines 3 to 8 produce a descent direction which may not necessarily have an acceptable length. In line 3 the matrix M is formed. The linear system (19) is tentatively solved in line 4. The solution procedure may not be successful (i.e. M may be singular). This implies that the parameter γ is too large. If this occurs, the parameter γ is reduced in line 5 of loop 4-7, which is repeated until a nonsingular matrix M is produced.

Once a nonsingular M matrix is available, a descent direction Δw^* is computed in line 8 along with its corresponding length l. Three cases can occur: (i) - the length is too small even though no linear dependency was detected in the factorization; (ii) - the length is too large; or (iii) - the length is acceptable. Case (iii) is the termination condition for the main loop 2-19. In lines 9-13 the first case is considered. The value of γ is a lower bound on an acceptable value of γ and is recorded in line 10 and the corresponding logical key is set. If an upper bound $\overline{\gamma}$ for an acceptable value of γ has been found the new estimate for γ is set to the geometric mean of $\underline{\gamma}$ and $\overline{\gamma}$ in line 11. Otherwise γ is increased by a fixed factor in line 12.

Similar to the treatment of case (i), case (ii) is handled in lines 14-18. The value of γ is an upper bound on an acceptable value of γ and is recorded in line 15 and the corresponding logical key is set. If a lower bound $\underline{\gamma}$ for an acceptable value of γ has been found the new estimate for γ is set to the geometric mean of $\underline{\gamma}$ and $\overline{\gamma}$ in line 16. Otherwise γ is decreased by a fixed factor in line 17.

In line 20, the lower bound $\underline{l}$ may have to be adjusted if $l < \underline{l}$ and $LD_{key} = \textbf{true}$. Finally, the search direction Δw^* is returned in line 21.

4. Computational Results

A variant of **procedure** `ip` has been implemented and tested on independent k-set problems in dense random graphs. In this section we present these computational results. These results are preliminary, since further implementation is ongoing. We have selected this class of problems because it is easy to generate hard instances using some results from the theory of random graphs [4].

Consider a graph $G = (V, E)$ with vertex set V and edge set E and its complement, $\bar{G} = (V, \bar{E})$, where $\bar{E} = \{(i,j) \notin E;\ i,j \in V, i \neq j\}$. An *independent set* of vertices (or vertex packing or stable set) is a vertex set whose elements are pairwise nonadjacent, i.e. a subset $S \subset V$ is independent if $\forall\, i,j$ such that $i,j \in S$, $(i,j) \notin E$. We call an *independent k-set* of vertices an independent set made up of k vertices. A *clique* is a maximal complete subgraph of G. A k-clique is a clique with vertex set of size k. A *vertex cover* is a subset $S \subset V$ such that all edges of G have at least one endpoint in S. In a k-vertex cover $|S| = k$. It is well known that S is a maximum independent set of G if and only if S is a maximum clique of $\bar{G}$ if and only if $V \setminus S$ is a minimum vertex cover of $\bar{G}$. Finding a maximum independent set, a maximum clique or a minimum vertex cover of a graph are equivalent and are known to be NP-complete [7]. Practical applications of these models are abundant, e.g. information retrieval, signal transmission analysis, classification theory, economics, scheduling, experimental design and computer vision (See [1], [2], [3], [5], [6], [15], [9] and [16] for details).

A simple integer programming formulation for vertex packing is given next. Let $S \subset V$ be an independent k-set of the graph $G = (V, E)$ and define

$$w_j = \begin{cases} 1 & \text{if vertex } j \in S \\ -1 & \text{otherwise} \end{cases}$$

A (-1,1) integer programming formulation is: Find $w \in \Re^{|V|}$ such that

$$-\sum_{j \in V} w_j \leq |V| - 2 \cdot k$$

$$w_i + w_j \leq 0, \ \ \forall (i,j) \in E$$

$$w_j = \{-1, 1\} \ \ \forall j \in V$$

Before we describe the computational experiment, we need to describe some implementation details, e.g. describe how the initial solution is generated, how rounding off is carried out, how the linear system is solved, how local minima are treated and what parameter settings are used. We use as the initial solution

$$w_j = -\frac{|V| - 2 \cdot k}{|V| + 1}, \ \ \forall j \in V.$$

One simple way to round off the components is as follows:

$$\tilde{w}_j = \begin{cases} 1 & \text{if } w_j > 0 \\ -1 & \text{if } w_j \leq 0 \end{cases} \tag{20}$$

This scheme will always round off to an integer solution that corresponds to an independent set. However, components that are slightly negative are assigned value -1 with this procedure, even if they could be be assigned a +1 and thus increase the size of the independent set found. In our implementation, at each iteration the vertices of the graph are considered in decreasing order of w and are placed into the k-set if this is feasible. This will always produce a k-set of size at least as large as the one produced by scheme (20).

One approach for dealing with local minima is the following. Consider all pairs of vertices in $\mathcal{K}$ (in decreasing order of w) and for each pair fix those vertices in the independent set. If the two vertices and all vertices adjacent to them are removed from the graph, the resulting graph is very small. By enumeration, a maximum independent set of the reduced graph is readily available. For enumeration we use the semi-exhaustive greedy independent set algorithm [8] implemented by D.S. Johnson with parameter settings to do exhaustive enumeration.

At each call of **procedure** `descent_direction` a linear system $M\Delta w^* = b$ must be solved at least once (lines 4-8). We use the conjugate gradient algorithm with preconditioning to solve these linear systems. At the first iteration of `ip` a full factorization of M is carried out and those factors are used as preconditioners for that and subsequent iterations. The conjugate gradient algorithm terminates when the residue $\|M\Delta w^* - b\|_2 < 10^{-10}$ or when 20 conjugate gradient iterations are completed. A refactorization is called for when either the conjugate gradient algorithm required 20 iterations or when $|1 - \cos\theta| > 10^{-6}$, where θ is the angle between $\hat{b} = M\Delta w^*$, produced by the conjugate gradient algorithm, and the right hand side b.

The other parameters were set as follows: In line 1 of `ip` $\gamma_0 = 32$, $\underline{l}_0 = 0.5$ and $\bar{l}_0 = 1.0$. In line 6 of `ip` $\epsilon = 10^{-4}$. In line 11 of `ip` $\alpha = 0.5$. In lines 5, 12 and 17 of `descent_direction` $\gamma_r = \sqrt{2}$. In line 20 of `descent_direction` $l_r = .25$.

Most of the code was written in FORTRAN with some code in C. We ran our experiment on a VAX* 8700 running Ultrix*. FORTRAN code was compiled with the `f77` compiler and C code was compiled with the `cc` compiler. The compiler optimization flag `-O` was set. For the largest instances memory requirements exceeded 240 Mbytes. This memory requirement will be reduced considerably with the version of the linear system solver under development.

* VAX and Ultrix are trademarks of the Digital Equipment Corporation.

We have generated 187 dense random graphs to test our implementation of `ip`. We used $p = 0.5$ for all instances. We generated 100 graphs with 100 vertices, 50 graphs with 200 vertices, 25 graphs with 300 vertices, 10 graphs with 500 vertices and 2 graphs with 1000 vertices. Random graphs with vertex set V and where edges from the complete graph on $|V|$ vertices are placed in the edge set E independently with probability p, have maximum independent sets with cardinalities that are distributed in a highly concentrated manner, i.e. almost all of the graphs have maximum independent sets of the same size [4]. The expected number $E_k = E(X_k)$ of independent sets of size k is given by

$$E_k = \begin{pmatrix} |V| \\ k \end{pmatrix} p^{\begin{pmatrix} k \\ 2 \end{pmatrix}}. \tag{21}$$

The variance $\sigma^2(X_k)$ of independent sets of size k is

$$\sigma^2(X_k) = \begin{pmatrix} |V| \\ k \end{pmatrix} \sum_{r=0}^{k} \begin{pmatrix} k \\ r \end{pmatrix} \begin{pmatrix} |V|-k \\ k-r \end{pmatrix} (1-p)^{k(k-1)-\begin{pmatrix} r \\ 2 \end{pmatrix}} - E(X_k)^2. \tag{22}$$

An upper bound U_k on $\text{Prob}(X_k = 0)$ is given by

$$U_k = \sigma^2(X_k)/E(X_k)^2. \tag{23}$$

Using (21-23), one can generate graphs of size, for example, $|V| = 1000$ and $p = 0.5$ such that with high probability there exists an independent k-set of size 15 and there exists no independent k-set of size 16. Tables 4.1-4.5 give E_k and U_k for the graphs generated in this experiment.

k	E_k	U_k
7	7.63×10^3	1.38×10^{-1}
8	6.93×10^2	3.46×10^{-1}
9	2.77×10^1	1.21×10^0
10	4.92×10^{-1}	1.28×10^1
11	3.93×10^{-3}	6.58×10^2

Table 4.1 – $|V| = 100$ and $p = 0.5$

k	E_k	U_k
9	1.71×10^4	1.05×10^{-1}
10	6.38×10^2	2.36×10^{-1}
11	1.08×10^1	1.10×10^0
12	8.28×10^{-2}	3.73×10^1
13	2.92×10^{-4}	6.17×10^3

Table 4.2 – $|V| = 200$ and $p = 0.5$

k	E_k	U_k
10	3.97×10^{4}	9.99×10^{-2}
11	1.02×10^{3}	9.99×10^{-1}
12	1.20×10^{1}	9.99×10^{-1}
13	6.51×10^{-2}	9.99×10^{1}
14	1.63×10^{-4}	9.99×10^{3}

Table 4.3 – $|V| = 300$ and $p = 0.5$

k	E_k	U_k
12	6.05×10^{3}	5.45×10^{-2}
13	5.54×10^{1}	1.74×10^{-1}
14	2.35×10^{-1}	9.93×10^{0}
15	4.65×10^{-4}	3.34×10^{3}
16	4.31×10^{-7}	2.94×10^{6}

Table 4.4 – $|V| = 500$ and $p = 0.5$

k	E_k	U_k
14	4.23×10^{3}	2.29×10^{-2}
15	1.70×10^{1}	1.78×10^{-1}
16	3.19×10^{-2}	5.07×10^{1}
17	2.81×10^{-5}	4.58×10^{4}

Table 4.5 – $|V| = 1000$ and $p = 0.5$

Instances of size $|V| = 100$, 200, 300 and 500 were run on the enumeration algorithm `enum` and `ip` was asked to find a set as good as the one found by `enum`. The enumeration algorithm required over 20 hours of cpu for each $|V| = 500$ instance. We attempted to run `enum` on a 1000 vertex instance but gave up, after one week of real time produced no solution of size 15. We asked `ip` to look for solutions of size 15 on all $|V| = 1000$ instances.

Table 4.6 summarizes the results of our runs. The table shows several algorithm statistics, averaged over all problems tested. These statistics are given by problem size classes and include number of `ip` iterations, number of refactorization calls, refactorization time, number of conjugate gradient calls, number of conjugate gradient iterations, conjugate gradient time, number of enumeration calls and size of enumeration graph. The `ip` itrs, refact calls and refact secs statistics for problem class of size $|V| = 1000$ is shown to be lower than what in it is, since problem 2 of that class was stopped at iteration 100, well before reaching a local minimum.

size of G	$\|V\| = 100$	$\|V\| = 200$	$\|V\| = 300$	$\|V\| = 500$	$\|V\| = 1000$
`ip` itrs	169.8	208.4	246.0	438.8	340.5
refact calls	3.1	2.8	3.0	3.5	2.5
refact secs	1.6	27.6	49.1	209.1	1761.6
`cg` calls	216.3	266.5	294.6	491.6	387.5
`cg` itrs	13.3	15.4	13.7	14.5	12.8
`cg` secs	1.1	11.0	9.7	24.3	87.1
`enum` calls	4.2	11.5	16.0	4.5	16.5
size `enum` graph	32.4	60.9	90.4	153.1	272.8

Table 4.6 – Computational Results (Averages)

We make the following observations regarding the computational experiment:

- The algorithm found the solution it was seeking in 86.1% of the cases. The breakdown per problem class is

$\|V\|$	problems	successes	%
100	100	94	94.0
200	50	38	76.0
300	25	17	68.0
500	10	10	100.0
1000	2	2	100.0

Table 4.7 – Algorithm effectiveness

- Procedure `ip` converged to a local minimum in most problems and in those cases required a local search to find the global minimum. It converged to a global minimum in 9 problems of size $|V| = 100$ and 2 of size $|V| = 200$.

- In all the cases where the algorithm failed, the solution it found was never off by more than 1 from the value sought.

- As indicated by the success of enumeration as a backend to `ip`, there is a substantial intersection between k-set defined by the rounded solution found by `ip` and the k-set corresponding to a global minimum.

- The conjugate gradient algorithm with preconditioning works surprisingly well. Refactorizations were rare and the number of conjugate gradient iterations averaged in the teens for all problem classes. By using time balancing, i.e. monitoring the duration of the conjugate gradient procedure and factorizing when conjugate gradient time reaches a given threshold, we should be able to keep the average number of conjugate gradient iterations below 5 and still reduce total CPU time.

5. Concluding Remarks

This paper described preliminary results of ongoing research. More computational experience is called for. Our future work will include implementation of improved data structures, a rank-1 update scheme for the factorization and specialized variants for different problem classes. We plan to test the procedure on other NP-complete problems, such as graph partitioning, graph coloring, the traveling salesman problem, inductive inference and linear ordering. This approach has been shown to work well for small global VLSI routing problems [13]. With the improved data structures we plan to evaluate its applicability to real-world VLSI routing.

References

[1] G. Avondo-Bodeno. *Economic applications of the theory of graphs.* Gordon & Breach Science Publishers, 1962.

[2] Egon Balas and Chang Sung Yu. Finding a maximum clique in an arbitrary graph. *SIAM Journal of Computing*, 15:1054–1068, 1986.

[3] C. Berge. *The theory of graphs and its applications.* Methuen, 1962.

[4] Béla Bollobás. *Random Graphs.* Academic Press, 1985.

[5] V. Degot and J.M. Hualde. De l'utilisation de la notion de clique en matière de typologie des populations. *R.A.I.R.O.*, 1, 1975.

[6] N. Deo. *Graph theory with applications to engineering and computer science.* Prentice-Hall, 1974.

[7] Michael R. Garey and David S. Johnson. *Computers and intractability - A guide to the theory of NP-completeness.* W.H. Freeman and Company, 1979.

[8] A. Johri and D.W. Matula. Probabilistic bounds and heuristic algorithms for coloring large random graphs. unpublished manuscipt, 1982.

[9] L.E. Trotter Jr. Solution characteristics and algorithms for the vertex packing problem. Technical Report 168, Dept. of Operations Research, Cornell University, Ithaca, NY, 1973.

[10] N. Karmarkar. An interior-point approach to NP-complete problems – extended abstract. In *Mathematical developments arising from linear programming algorithms.* Summer Research Conference sponsored jointly by AMS, IMS and SIAM. Bowdoin College, Brunswick, Maine, June 1988.

[11] K. Levenberg. A method for the solution of certain problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[12] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11:431–441, 1963.

[13] R. Pai, N. Karmarkar, and S.S.S.P. Rao. A global router based on Karmarkar's interior point method. Technical report, Indian Institute of Technology, Bombay, April 1988.

[14] S. Sahni. Computationally related problems. *SIAM Journal of Computing*, 3:262–279, 1974.

[15] C.E. Shannon. The zero-error capacity of a noisy channel. *I.R.E. Transactions*, 3, 1956.

[16] J. Turner and W.H. Kautz. A survey of progress in graph theory in the Soviet Union. *SIAM*, 12, 1970.

[17] Yinyu Ye. On the interior algorithms for nonconvex quadratic programming. Technical report, Integrated Systems Inc., Santa Clara, CA, 1988.