CHALMERS | GÖTEBORG UNIVERSITY

MASTER'S THESIS

Large scale 0-1 minimization problems solved using dual subgradient schemes, ergodic sequences and core problems

Carolin Juneberg

Department of Mathematics
CHALMERS UNIVERSITY OF TECHNOLOGY
GÖTEBORG UNIVERSITY
Göteborg Sweden 2001

Thesis for the Degree of Master of Science

Large scale 0-1 minimization problems solved using dual subgradient schemes, ergodic sequences and core problems

Carolin Juneberg

CHALMERS | GÖTEBORG UNIVERSITY

Department of Mathematics
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
Göteborg, March 2001

Matematiskt centrum
Göteborg 2001

Abstract

We present a heuristic algorithm for solving large scale 0-1 minimization problems. The algorithm is based on Lagrangean relaxation of complicating side constraints together with subgradient optimization and an ergodic, i.e. averaged, sequence of subproblem solutions. This sequence tends towards a solution to the continuous relaxation of the original program. This solution predicts optimal values for a (large) subset of the variables. The remaining, unpredictable variables are included in a core problem which is considerably smaller than the original one, and which is solved to optimality.

The algorithm was implemented in the C programming language together with Cplex callable library. It is tested on large scale 0-1 minimization problems arising in airline crew scheduling. Our strategy has shown to work well on many problems. For 50% of the test problems we found the optimal solution. The size of the core problems vary between 0.25% and 7.5% of the size of the original problem. We have not been able to solve unicost problems.

Sammanfattning

Vi presenterar en heuristisk algoritm för att lösa storskaliga 0-1 minimiserings problem. Algoritmen baseras på Lagrange relaxering av de besvärliga bivillkoren tillsammans med subgradient optimering och en ergodisk sekvens av subproblem lösningar. Sekvensen är en approximativ lösning till det ursprungliga problemets kontinuerliga relaxering. Denna lösning förutsäger optimala värden för en stor delmängd av variablerna. De kvarvarande, oförutsägbara variablerna inkluderas i ett kärnproblem som är mycket mindre än ursprungsproblemet, och som löses till optimalitet.

Algoritmen implementerades i programmeringsspråket C tillsammans med Cplex och vi har löst storskaliga 0-1 minimiserings problem som uppkommer inom flygbesättningsoptimering. Vår strategi har visat sig fungera väl på många problem. Till 50% av testproblemen fann vi optimallösningen. Storleken på kärnproblemen varierar mellan 0.25% till 7.5% av ursprungsproblemets storlek. Vi har inte lyckats lösa problem med lika kostnad.

Preface

The work presented in this report completes my studies in mathematics at Göteborg University, leading to the degree Master of Science.

I would like to thank my supervisor Ann-Brith Strömberg at Chalmers University of Technology and Göteborg University for making this possible, and for all the input and support through out the project.

Contents

1 Background

Over the last thirty years there have been much research done in the field of planning and scheduling crews. One reason may be that next to fuel costs the crew costs are the largest direct operating costs of airlines [AHKW]. The problem involves many different parts and is therefore solved in several steps; production of the timetable, allocation of aircraft to the flight legs (a flight leg is a non-stop flight between two airports), construction of pairings (a pairing is a sequence of flight legs for a crew) and the assignment of pairings to crew members. One constructs legal pairings (often more than a million) that satisfies (all) existing rules, such as governmental rules and union agreements. A subset of all these pairings is chosen such that every flight leg gets a crew and so that the total cost is minimized. Here, the optimization problem is modelled under the assumption that the set of legal pairings is available.

A typical airline crew scheduling problem deals with about 50 to 2000 flight legs and 10,000 to 1,000,000 possible pairings [Wed95]. Of course, this becomes a very large-scale integer programming problem and therefore is very difficult to solve. Normally this complex problem is solved with advanced software engineering. Companies like Carmen Systems AB, Resource Management AB and AD OPT Technologies Inc. develop and market systems to airlines for planning of aircrafts and flight crews. Our strategy is to use a heuristic method based on Lagrangean relaxation of the set covering constraints, together with subgradient optimization for the resulting Lagrangean dual problem and an ergodic, i.e. averaged, sequence of subproblem solutions. Through the ergodic sequence we will predict whether a specific pairing is included in the optimal set of pairings or not. We then get a smaller problem consisting of the pairings that we were not able to make any predictions about. This smaller problem is called a *core problem* and may be solved either exactly or approximately. In this work, we will examine whether or not our choice of method is applicable in practice and perform tests to compare the solution obtained with our method with the optimal solution.

In Section 2 we present the method and the theory behind it. The method is specialised for solving set covering problems arising in airline crew scheduling in Section 3 and in Section 4 we describe the tests we have performed and present the results. In the final section, Section 5, we draw conclusions.

2 Methodology

In Section 2.1, we model the crew scheduling problem as an integer optimization program and in Sections 2.2–2.4 we present the tools used for finding solutions to this program.

2.1 Modelling the problem as an IP-problem

The crew scheduling problem consists of pairings. Pairings are work plans, that is, possible sequences of flight legs for an unspecified crew member starting and ending at the home base, satisfying all existing rules, such as governmental rules and union agreements, for example the length of a duty period. A subset of all these possible pairings is chosen such that every flight gets a crew and such that the total cost is minimized. The optimization problem is modelled under the assumption that the set of *all* legal pairings is available. This gives the following mathematical formulation of the crew scheduling problem.

Let n be the number of possible pairings and m the number of flight legs that have to be covered. The variables x_j are defined as:

$$x_j = \begin{cases} 1, & \text{if pairing } j \text{ is in the solution,} \\ 0, & \text{otherwise,} \end{cases} \qquad j = 1, \ldots, n.$$

We let c_j denote the cost associated with the choice of pairing j. One set of constraints, the covering constraints, expresses that each flight leg must be covered by at least b_i crew members. For cockpit crew, typically, $b_i = 1$. The reason for the inequality relation in the covering constraints are the possibility of deadheading, that is, that a crew member is transported passively as a passenger to the home base or to some other base for another duty period. The parameters a_{ij} are defined as:

$$a_{ij} = \begin{cases} 1, & \text{if pairing } j \text{ is operating leg } i, \\ 0, & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

With the above defined parameters we have the following mathematical model of the crew scheduling problem, called a *set covering problem*:

$$\begin{aligned}
\min \quad & \sum_{j=1}^{n} c_j x_j && \text{(SCP)} \\
\text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad i = 1, \dots, m \\
& x_j \in \{0, 1\} \quad j = 1, \dots, n.
\end{aligned}$$

This is a NP-complete optimization problem [Kar72] (a class of problems for which no polynomial algorithm has been found), i.e., solving a NP-complete problem with an optimal algorithm requires a considerable computational effort. Therefore, a computationally effective or cheap heuristic algorithm capable of producing high quality (that is, near-optimal) solutions is preferable. The difficulties are due to the combination of covering constraints, integrality requirements, and the size of the problem that is, m and n.

2.2 Lagrangean relaxation

We may view this linear integer optimization problem as a fairly easily solvable problem with a number of complicating side constraints. In order to make simplifications we use the Lagrangean relaxation technique, but first some definitions. A general model for an integer program may be defined as:

$$\begin{aligned}
z^* = \min \quad & c^T x && \text{(IP)} \\
\text{s.t.} \quad & Ax \geq b \\
& x \in X = Z_+^n \cap X_P,
\end{aligned}$$

where $c \in R^n$, $b \in R^m$, $A \in R^{m \times n}$ and $X_P \subset R^n$ is a bounded polyhedron. We assume that the set of feasible solutions to (IP) is nonempty; then the set of optimal solutions, X^*, is nonempty and bounded. We consider that $Ax \geq b$ are the complicating constraints. For the integer program (IP) the linear programming relaxation is given by

$$\begin{aligned}
z_{LP}^* = \min \quad & c^T x && \text{(LP)} \\
\text{s.t.} \quad & Ax \geq b \\
& x \in X_{LP} = R_+^n \cap X_P.
\end{aligned}$$

That is, we have replaced the integrality constraint in (IP) by its continuous relaxation. A well known relation [Geo74] between z^* and z^*_{LP} is,

$$z^*_{LP} \leq z^*. \tag{1}$$

The Lagrangean relaxation technique associates a vector, $\lambda \in R^m$, to the complicating constraints and bring them into the objective function. This vector λ can be interpreted as a penalty for not fulfilling the associated constraints. We define $\theta : R^m \mapsto R$ as the concave (and continuous) function,

$$\theta(\lambda) = \min_{x \in X}[c^T x + \lambda^T(b - Ax)] = b^T\lambda + \min_{x \in X}[c - A^T\lambda]^T x. \tag{2}$$

Let $x(\lambda)$ denote a solution to the inner minimization problem (2) at λ. For the special case (SCP) of (IP), (2) is easily solved by inspection, and $x(\lambda)$ is given by

$$x_j(\lambda) = \begin{cases} 1, & \text{if} \quad c_j - \sum_{i=1}^m \lambda_i a_{ij} \leq 0, \\ 0, & \text{otherwise,} \end{cases} \qquad j = 1, \dots, n,$$

where $\lambda = (\lambda_1, \dots, \lambda_m)^T$. For all $\lambda \geq 0$, the solution, $x(\lambda)$, yields a lower bound, $\theta(\lambda)$, on the optimal value z^* of the original problem (IP) [Geo74]:

$$\theta(\lambda) = c^T x(\lambda) + \lambda^T(b - Ax(\lambda)) \leq z^*, \quad \lambda \geq 0. \tag{3}$$

Typically, a solution $x(\lambda)$ to the Lagrangean subproblem (2) is infeasible in the original problem.

A heuristic algorithm is an approximative method for solving problems. Such a method may be designed in different ways for a problem, depending on the solution desired. One purpose may be to quickly generate a feasible solution (using a simple heuristic method) and another to generate a high-quality (i.e., near-optimal) solution (using a more complicated heuristic procedure). We will use a simple heuristic procedure to convert a relaxed solution $x(\lambda)$ into a feasible solution for the original problem, see Section 3.2. This modified primal feasible solution, x_H, yields an upper bound, $\bar{z}$, on the optimal solution to the original problem (IP):

$$z^* \leq c^T x_H \stackrel{\text{def.}}{=} \bar{z}, \quad \text{where } x_H \text{ is a feasible solution to (IP).}$$

2.3 Lagrangean duality and subgradient optimization

We are interested in finding the values of $\lambda_i, i = 1, \dots, m$, that give the maximum lower bound, θ^*. This maximization problem, the Lagrangean dual program, is defined as

$$\theta^* = \max_{\lambda \in R^m_+} \theta(\lambda). \tag{LD}$$

It follows from (3) that $\theta^* \leq z^*$ [Geo74]. Ideally, the optimal value of the Lagrangean dual program is equal to the optimal value of the original problem, i.e., $\theta^* = z^*$, that is when (IP) is continuous. If the two programs, (IP) and (LP), do not have equal optimal values then a duality gap, $z^* - \theta^* > 0$, is present. In our case a duality gap is present.

The relationship between θ^* and z^*_{LP}, is explained next.

Definition 1 *The convex hull of a bounded polyhedron $Y \subseteq R^n$, denoted conv(Y), is defined as*

$$conv(Y) = \left\{ y : y = \sum_{i=1}^{k} \alpha_i y^i, \sum_{i=1}^{k} \alpha_i = 1, \alpha_i \geq 0, i = 1, \ldots, k \right\},$$

where $y^i, i = 1, \ldots, k$, are the extreme points of Y.

We replace X in (IP) by its closed convex hull, conv(X). This gives the convex program

$$\begin{aligned} z^*_{CP} = \min \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \\ & x \in \text{conv}(X) \end{aligned} \tag{CP}$$

Since $conv(X) \supseteq X$, (CP) is a relaxation of (IP), and we have that

$$z^*_{CP} \leq z^*.$$

Since $X \subseteq X_{LP}$, we have that $conv(X) \subseteq conv(X_{LP}) = X_{LP}$, implying that (LP) is a relaxation of (CP), and we have that

$$z^*_{LP} \leq z^*_{CP}. \tag{4}$$

According to [Geo74], we also have that

$$\theta^* = z^*_{CP},$$

The above results give us the following relations between z^*, θ^*, z^*_{CP} and z^*_{LP}:

$$z^* \geq \theta^* = z^*_{CP} \geq z^*_{LP}. \tag{5}$$

In the case when $conv(X) = X_{LP}$, the program (2) is said to have the *integrality property* [Geo74], implying that the value of the optimal solution for the Lagrangean dual program equals that of the linear programming relaxation, i.e., $\theta^* = z^*_{LP}$ ((4) and (5)). We expect to have integrality property.

The dual objective function, $\theta : R^m \mapsto R$, is continuous, concave and piecewise linear [Fis81]. Thus, we have to search for the maximum of a concave but not everywhere differentiable function over R^m_+. The optimality conditions for the Lagrangean dual program may be formulated in terms of subgradients.

Definition 2 *A vector $\xi \in R^m$ is a subgradient of the concave function θ at $\overline{\lambda} \in R^m$ if it satisfies [Fis81]*

$$\theta(\lambda) \leq \theta(\overline{\lambda}) + \xi^T(\lambda - \overline{\lambda}), \quad \forall \lambda \in R^m.$$

Following from (2) and Definition 2, a subgradient $\xi(\bar{\lambda})$ to the dual objective function θ at $\bar{\lambda}$ is given by

$$\xi(\bar{\lambda}) = b - Ax(\bar{\lambda}).$$

The Lagrangean dual program (LD) is solved by $\lambda^* \geq 0$ if and only if there is a subgradient, ξ^*, of θ at λ^* such that [LL97]

$$\xi^* \leq 0 \text{ and } (\lambda^*)^T \xi^* = 0.$$

For solving the Lagrangean dual program, i.e., to find optimal values λ^* of the multipliers λ we use *subgradient optimization*. The subgradient optimization technique is an iterative procedure. Given an initial value, $\lambda^1 \geq 0$, a sequence $\{\lambda^s\}$ of iterates is generated by the method

$$\lambda_i^{s+1} = \max\{\lambda_i^s + \gamma_s \underbrace{(b_i - A_i x^s)}_{\xi_i^s}, 0\}, \quad i = 1, \dots, m, \quad s = 1, 2, \dots, \tag{6}$$

where A_i denotes row i of A, $x^s = x(\lambda^s)$ is a solution to the Lagrangean subproblem (2) at λ^s and $\gamma_s > 0$ is the step length, all at iteration $s = 1, 2, \dots$. If $x^s = x(\lambda^s)$ then the step direction $\xi^s = b - Ax^s$ is a subgradient of the function θ at λ^s. By

$$\bar{z}^s = \min_{r=1,\dots,s} z_r^H \quad \text{and} \quad \underline{z}^s = \max_{r=1,\dots,s} \theta(\lambda^r), \quad s = 1, 2, \dots,$$

we will denote the best upper and lower bounds, respectively, on z^* found at iteration s.

It is useful to adjust the subgradients, by projection onto the binding constraints, if they do not contribute to the update of the multipliers λ in an effective manner, [Bea90], with theoretical convergence according to [LPS96]. The formula for the projection when the feasible set for (LD) is R_+^m is given by

$$\bar{\xi}_j^s = \begin{cases} 0, & \text{if } \lambda_j^s = 0 \text{ and } \xi_j^s \leq 0, \\ \xi_j^s, & \text{otherwise,} \end{cases} \qquad j = 1, \ldots, n. \tag{7}$$

The step length formula most successfully used in practice is, [Bea93], given by

$$\gamma_s = \frac{\pi^s(\bar{z}^s - \theta(\lambda^s))}{\| \xi^s \|^2}, \quad s = 1, 2, \ldots, \tag{8}$$

where $\pi^s \in [\sigma_1, 2 - \sigma_2], 0 < \sigma_1 \leq 2 - \sigma_2 < 2, s = 1, 2, \ldots$, is a parameter for guaranteeing theoretical convergence of the sequence $\{\lambda^s\}$ to an optimal solution to (LD) [Pol69]. The classical approach is to choose, $\pi^1 = 2$ and update π^s according to

$$\pi^{s+1} = \begin{cases} \beta \cdot \pi^s, & \text{if } \underline{z}^{s-p} = \underline{z}^s \\ \pi^s, & \text{otherwise,} \end{cases} \qquad s = q + 1, \ldots, \tag{9}$$

where $\beta \in (0, 1)$ and $p \in \{1, \ldots, q\}$. We have employed the projection formula (7), resulting in the step length formula,

$$\bar{\gamma}_s = \frac{\pi^s(\bar{z}^s - \theta(\lambda^s))}{\|\bar{\xi}^s\|^2}, \quad s = 1, 2, \ldots. \tag{10}$$

To choose the values of π^s, we have followed the recommendation in [CFT99], in order to obtain fast convergence to an optimal solution to (LD). The algorithm [CFT99] uses an adaptive step length parameter with $\pi^1 = 0.1$ and the following criterion for updating π^s:

$$\pi^{s+1} = \begin{cases} \frac{1}{2}\pi^s, & \text{if } \bar{\theta} - \underline{\theta} > 0.01 \cdot \underline{\theta}, \\ \frac{3}{2}\pi^s, & \text{if } \bar{\theta} - \underline{\theta} < 0.001 \cdot \underline{\theta}, \\ \pi^s, & \text{otherwise,} \end{cases} \qquad s = 1, 2, \ldots, \tag{11}$$

where

$$\bar{\theta} = \min_{r=s-p+1,\ldots,s} \theta(\lambda^r) \quad \text{and} \quad \underline{\theta} = \max_{r=s-p+1,\ldots,s} \theta(\lambda^r).$$

For every $p = 20$ subgradient iterations they compare the best and worst lower bounds computed during the last p iterations. If these two values differ more than 1%, the current value of π^s is halved. If the two values are less than 0.1% from each other, they multiply the current value of π^s by 1.5.

The difference in convergence speed between the updating formulas (9) and (11) is illustrated in Figure 1 for test problem 5.1, see Table 1.

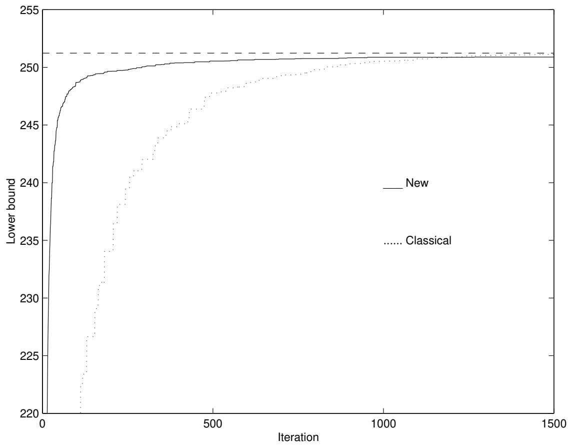

Figure 1: *Illustration of the iterative process in the subgradient method, for the classical approach (9) with $\pi^1 = 2$ and the new approach (11) with $\pi^1 = 0.1$. In both cases the subgradients are adjusted by projection (7). The lower bound progress towards the optimal value for (LD). The dashed line represents z^*_{LP}.*

One obvious way in which the subgradient procedure could be terminated (but rarely the case, and we don't expect that to be the case) is if we find that the maximum lower bound equals the upper bound (obtained via a heuristic method, Section 3.2). Then we have found an optimal solution. Another, and more useful, way to terminate the procedure is after a prespecified number of iterations. The iteration (6), (7), (10) solves the dual program (LD) in the limit, but an optimal solution to (IP) cannot be guaranteed to be obtained from the Lagrangean subproblem; this has been referred to as the non-coordinability phenomenon of linear programming [DJ79].

2.4 An ergodic sequence of primal subproblem solutions

We will utilize an ergodic sequence to find an optimal solution to (LP). This sequence is easily constructed in the subgradient optimization procedure.

As mentioned above, there is no guarantee for finding an optimal solution to (IP) when solving the Lagrangean dual program using the subgradient optimization technique combined with some heuristic method for finding feasible solutions. To get an optimal solution to (LP), an ergodic sequence, consisting of weighted averages of the Lagrangean subproblem solutions, may be utilized, see [LPS]. An ergodic sequence of the Lagrangean subproblem solutions is defined by

$$\bar{x}^t = \Gamma_t^{-1} \sum_{s=0}^{t-1} \gamma_s x^s, \quad \Gamma_t = \sum_{s=0}^{t-1} \gamma_s, \quad t = 1, 2, \ldots . \tag{12}$$

Here, $x^s = x(\lambda^s)$ is a solution to the Lagrangean subproblem (2) at λ^s and γ_s is the step length. The value $\bar{x}_j^t$ may be interpreted as the relative frequency by which the optimal value of the variable x_j is one in the sequence of subproblem solved. The sequence $\{\bar{x}^t\}$ converges to the set of optimal solutions to (LP) under the following conditions [LL97].

Theorem 3 *If the step lengths in the dual subgradient procedure (6) applied to the Lagrangean dual program (LD) satisfies the following conditions*

$$\gamma_s > 0, \quad \lim_{s\to\infty} \gamma_s = 0, \quad \sum_{s=0}^{\infty} \gamma_s = \infty, \quad \textit{and} \quad \sum_{s=0}^{\infty} \gamma_s^2 < \infty, \tag{13}$$

and the sequence $\{\bar{x}^t\}$ is defined by formula (12), then

$$\lim_{t\to\infty} \min_{x \in X_{LP}^*} \|x - \bar{x}^t\| = 0.$$

An alternative ergodic sequence, with equal weights on all subproblem solutions, is given by

$$\hat{x}^t = \frac{1}{t} \sum_{s=0}^{t-1} x^s, \quad t = 1, 2, \ldots . \tag{14}$$

The sequence (14) converges in the same manner as the sequence (12), given that the step lengths is chosen as a harmonic series

$$\gamma_s = \frac{a}{b + c \cdot s},$$

where $a, c > 0, b \geq 0$.

Experience from numerical experiments conducted in [LL97] and [LPS99] shows that a method with ergodic sequence $\{\hat{x}^t\}$ with equal weights performs better than the ergodic sequence $\{\bar{x}^t\}$ with weights proportional to the step lengths. The disadvantage of using (12) is that the initial poor solutions are more sensitive for the weights chosen in that sequence, compared to those of the sequence (14). Hence, we will use the ergodic sequence $\{\hat{x}^t\}$. A preferable procedure is to delay the start of computing the ergodic sequence $\{\hat{x}^t\}$ until the subproblem solutions x^s have reached an acceptable quality. Since the algorithm (6) is memoryless and we utilize the properties in the limit of the sequence $\{\hat{x}^t\}$, the delayed start of the averaging yields no problems with the theoretical convergence.

3 Application to airline crew scheduling

We have used the algorithm described above for solving set covering problems arising in airline crew scheduling.

3.1 Solution procedure

Our strategy is to use a heuristic method for predicting optimal values for a relatively large subset of the variables, x_j, and then solve a smaller problem constituted by the remaining variables. This smaller problem is called the core problem, and may be solved either exactly or approximately. In the construction of the core problem feasibility has to be maintained. Our hypothesis is that if the prediction is based on a solution to (LP) that is good enough to produce a feasible and near-optimal solution to the original problem (SCP). Our prediction is based on the primal ergodic iterates $\hat{x}^t$ as follows. Assume that the subgradient procedure has terminated, at iteration t, say. For appropriate values of ϵ_0 and ϵ_1, if $\hat{x}_j^t \geq 1 - \epsilon_1 (\leq \epsilon_0)$, then fix x_j to 1(0). The remaining variables (those x_j for which $\epsilon_0 < \hat{x}_j^t < 1 - \epsilon_1$) are included in the core problem. If the core problem is feasible, then we solve it, otherwise, we decrease the tolerances ϵ_0 and ϵ_1 and repeat. As an alternative, the prediction could be based on near-optimal reduced costs.

3.2 The algorithm

By applying Lagrangean relaxation of the covering constraints, with multipliers $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T$, to (SCP), the Lagrangean subproblem (2) is obtained. The Lagrangean dual program is solved by the subgradient optimization algorithm, (6) and (7), and an ergodic sequence $\{\hat{x}^t\}$ of subproblem solutions is simultaneously computed, using (14).

Some of the solutions will, typically, violate some of the relaxed covering constraints. Therefore, in the subgradient optimization procedure, a *heuristic algorithm* is included that will produce feasible solutions from infeasible ones, as follows. For each row i which is uncovered (i.e., $\sum_{j=1}^{n} a_{ij} x_j^s = 0$) set $x_k^s = 1$ for column k corresponding to $c_k = \min_j \{c_j | a_{ij} = 1\}$.

For the step lengths in the subgradient procedure (6) there are some alternatives. One option is to choose them according to (8), in which case theoretical convergence, $\{\theta(\lambda)\} \to \theta^*$, is given in [Pol69], and the practical convergence is according to [HWC74]. Another option is to choose them according to a modified harmonic series (a special case of (13)),

$$\bar{\gamma}_s = \frac{M}{b+s}, \quad M > 0, \quad b > 0, \tag{15}$$

with theoretical convergence, $\{\theta(\lambda^s)\} \to \theta^*$ and $\{\lambda^s\} \to \lambda^*$, according to Theorem 2.7 in [LPS96]. A final option, a special case of (13) and a generalization of (15), is to choose

$$\frac{\mu}{b+s} \leq \bar{\gamma}_s \leq \frac{M}{b+s}, \quad M \geq \mu > 0, \quad b > 0, \tag{16}$$

with the same convergence properties as (15). In practice, we will choose step lengths according to (10) until we start computing the ergodic sequence, and then as a modified harmonic series (15), with $b = 0$ and, letting r be the start iteration for computing the ergodic sequence, choosing M according to

$$M = \frac{r}{10} \sum_{s=r-9}^{r} \bar{\gamma}_s.$$

Our solution scheme is summarized as follows.

1. Initialize $s = 0$, $\bar{z}^0 = \infty$, $\underline{z}^0 = -\infty$ and $\lambda_i^0 = \min_{j=1,\dots,n}\{c_j | a_{ij} = 1\}$ $(i = 1, \dots, m)$. Choose $\epsilon > 0$.

2. Choose the number, t, of subgradient iterations to be performed, the start iteration r for computation of the ergodic sequence, the lower and upper tolerances, $\epsilon_0, \epsilon_1 \in R_+, 0 < \epsilon_0 + \epsilon_1 < 1$, and the maximal relative difference $\epsilon \geq 0$ between $z(\epsilon_0, \epsilon_1)$ and $\underline{z}^s$, where $z(\epsilon_0, \epsilon_1)$ is the value of the solution. Perform t subgradient iterations applied to the Lagrangean dual program (LD), and apply the heuristic method in each iteration in order to compute $\bar{z}^s$ and $\underline{z}^s$.

3. Fix the binary variables x_j with values $\hat{x}_j^t \geq 1 - \epsilon_1$ to 1 and those x_j with values $\hat{x}_j^t \leq \epsilon_0$ to 0. If the number of variables fixed to 1 is larger than the number of rows, then decrease the upper tolerance ϵ_1 and repeat step 3.

4. When creating the core, rows and columns are deleted from the original problem. The remaining problem is the core. The core is constructed by deleting every column j that corresponds to a fixed variable x_j and for every variable x_j fixed to 1 delete those rows i for which $a_{ij} = 1$. If the core problem is feasible, then solve it. Otherwise, decrease the lower tolerance ϵ_0 and repeat from step 3.

5. Construct a solution to the original problem from the values of the fixed variables and those of the core problem.

6. Examine the solution to the original problem. If

   a. $z(\epsilon_0, \epsilon_1) > \bar{z}^s$, then decrease the upper tolerance ϵ_1 and repeat from step 3.

   b. the solution is satisfactory, that is,

$$\frac{z(\epsilon_0, \epsilon_1) - \underline{z}^s}{\underline{z}^s} \leq \epsilon, \tag{17}$$

   then terminate the algorithm.

   c. $\underline{z}^s(1 + \epsilon) < z(\epsilon_0, \epsilon_1) \leq \bar{z}^s$ decrease the upper and lower tolerances, ϵ_0 and ϵ_1, and repeat from step 3.

4 Computational results

The computer implementation of the algorithm was made in the C programming language together with the Cplex callable library, which provides an integer programming solver, for solving the core problem. The existing C-function `scpreader()` reads the data of a set covering problem from a file and stores it in appropriate data structures. We have used test problems of different sizes and structures from Beasley's OR-Library (denoted x.x; where x denotes integer numbers, and they may be found on the Internet, http://mscmga.ms.ic.ac.uk/jeb/jeb.html) and real life problems from the Swedish Railway Company (denoted sjxx).

4.1 Problem reduction

By inspection of the problems one might find unnecessary information that increases the size of the problems. Such information can be removed without changing the solutions, in order to reduce the size of the problems. For example there can be identical columns or rows dominating each other, i.e, whenever a certain constraint is fulfilled some other constraints are fulfilled too. We used the Cplex function Presolve to examine whether the problems were as compact as possible, and made reductions where possible. For our tests we then used the presolved problems.

4.2 Parameter settings

In the algorithm there are parameters to calibrate. We are aiming for a small, easily solved core problem yielding a high quality solution for (SCP), and for this we utilize an ergodic sequence with calibrated parameters, ϵ_0 and ϵ_1.

The size of the core problem and the quality of the solution is dependent on the ergodic sequence and due to the fact that we can not make up for a bad sequence afterwards, we started by calibrating the parameters concerning the computation of the sequence, which involves the start iteration, r, for computation of the the ergodic sequence and the total number of iterations, t, to be made in the subgradient optimization scheme.

An optimal solution to the continuous relaxation (LP) corresponds to a "perfect" ergodic sequence, since the ergodic sequence converges towards an optimal solution to (LP) in the limit. Hence, we also solve (LP) to optimality, both with the simplex and the barrier algorithm (within Cplex).

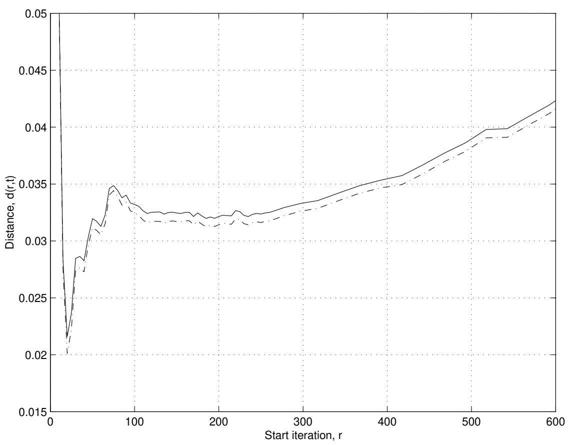

Figure 2: *Illustration of the average dependence, for a.1-a.5 from Beasley's OR-library with the same size and density, between the start iteration r and the distance, $d_S(r,t)$ (solid line) and $d_B(r,t)$ (dashdotted line). Here, $t = 2000$.*

Then, we use the euclidean distances between $\hat{x}^t(r)$, the ergodic iterate when the averaging is delayed r iterations, and an optimal solution to (LP) from the Simplex algorithm, given by

$$d_S(r,t) = \frac{\|\hat{x}^t(r) - x_S^*\|_2}{\sqrt{n}} = \frac{\sqrt{\sum_{j=1}^n |\hat{x}_j^t(r) - (x_S^*)_j|^2}}{\sqrt{n}}$$

and the corresponding distance between $\hat{x}^t(r)$ and an optimal solution x_B^* to (LP) from the Barrier method, given by

$$d_B(r,t) = \frac{\|\hat{x}^t(r) - x_B^*\|_2}{\sqrt{n}} = \frac{\sqrt{\sum_{j=1}^n |\hat{x}_j^t(r) - (x_B^*)_j|^2}}{\sqrt{n}},$$

as measures of the quality of the ergodic sequence solution $\hat{x}^t(r)$. Using these definitions, the maximum distance between the ergodic iterate and an optimal solution to (LP) that may occur is 1, that is, $0 \leq d_S(r,t) \leq 1$ and $0 \leq d_B(r,t) \leq 1$. We are searching for parameter values t and r yielding the minimum distance and then compute the best ergodic solution, $\hat{x}^t(r)$. We supposed that r and t depend on the size and density of the problem and therefore, we decided to run totally $t = 2000$ subgradient iterations and then compare the distances $d_S(r, 2000)$ and $d_B(r, 2000)$ for different start iterations r.

In Figure 2 we can see that the very first subproblem solutions obtained in

Table 1: *The start iterations, r, yielding the minimum distance $d_S(r,t)$ and $d_B(r,t)$ on the test instances 4.1-4.10, 5.1-5.10 and 6.1-6.5 from Beasley's OR-Library. Here, $t = 2000$. Size indicates rows $\times$ columns and Dens. means the density of the problem, i.e. the number of nonzero elements in the matrix divided by the size of the matrix ($m \cdot n$) times 100 (%).*

Prob.	Size	Dens. (%)	r	$d_S(r,t)$	$d_B(r,t)$
4.1	200x1000	2.00	33	0.083403	0.063239
4.2	200x1000	1.99	20	0.063829	0.063829
4.3	200x1000	1.99	15	0.051939	0.051939
4.4	200x1000	2.00	13	0.055333	0.031789
4.5	200x1000	1.97	11	0.060543	0.060543
4.6	200x1000	2.04	20	0.038648	0.029223
4.7	200x1000	1.96	13	0.078570	0.076400
4.8	200x1000	2.01	20	0.044870	0.044870
4.9	200x1000	1.98	19	0.024459	0.024459
4.10	200x1000	1.95	19	0.035153	0.026685
5.1	200x2000	2.00	42	0.026398	0.023990
5.2	200x2000	2.00	20	0.023075	0.020384
5.3	200x2000	2.00	20	0.029754	0.007230
5.4	200x2000	1.98	19	0.029289	0.022167
5.5	200x2000	1.96	17	0.012139	0.007406
5.6	200x2000	2.00	19	0.015556	0.015556
5.7	200x2000	2.01	20	0.017941	0.014610
5.8	200x2000	1.98	18	0.038800	0.024619
5.9	200x2000	1.97	22	0.049028	0.038823
5.10	200x2000	2.00	20	0.025838	0.009678
6.1	200x1000	4.92	22	0.044056	0.041091
6.2	200x1000	5.00	25	0.037343	0.035206
6.3	200x1000	4.96	25	0.028064	0.028064
6.4	200x1000	4.93	22	0.043020	0.041831
6.5	200x1000	4.97	25	0.019369	0.019369

the subgradient optimization scheme have a negative impact (given that x_S^* and x_B^* always are considered as “the best optimal solutions”, if there are more than one optimal solution, for our purpose) on the weighted averages, giving a larger distance compared to when the computation of the ergodic sequence is delayed, so that the initial poor subproblem solutions are not included in the averaging sequence. On the other hand, if the computation of the ergodic sequence is started too late, then there are too few iterations left to produce enough many different relevant subproblem solutions. We notice, in Figure 2 and Table 1, that $\hat{x}^t(r)$ is equally close or closer to x_B^* than to x_S^*. Therefore, in the following we use only $d_B(r,t)$ as a measure of the quality of the ergodic sequence.

Table 2: *An average value of the start iteration, $\tilde{r}$, and a safe interval for r, for the groups of problems arising in Table 1.*

Prob.	Size	Dens. (%)	$\tilde{r}$	Safe interval
4.1-10	200x1000	2	18	$50 \leq$ r ≤ 350
5.1-10	200x2000	2	22	$70 \leq$ r ≤ 300
6.1-5	200x1000	5	24	$80 \leq$ r ≤ 300

Table 3: *The start iteration r yielding the minimum distance $d_B(r,t)$ on the test instances sj01-05. Here, t $=2000$. We consider 200 $\leq$ r $\leq$ 450 to be a safe interval for sj01-05.*

Prob.	Size	Dens. (%)	r	$d_B(r,t)$
sj01	434x45397	1.71	57	0.018120
sj02	429x38391	1.66	53	0.017460
sj03	434x44837	1.71	59	0.016881
sj04	429x38148	1.66	61	0.014299
sj05	434x43968	1.70	51	0.017500

We can find a “best” start iteration r, or at least an interval for r giving satisfactory ergodic iterates. This interval represent a robust choice of the start iteration r and it will include the plateau which can be seen in Figure 2, since it is too risky to predict the initial dip. The “best” start iteration r for respective problems are stated in Table 1. Since some problems have similar properties, such as size and density, we group them together; as for example problems 4.1-4.10. For such a group of problems we will use the same parameter settings and get similar results. For these groups of problems, we present both a start iteration $\tilde{r}$, obtained as an average value of each problems individual "best" r, and a safe interval for r; see Table 2.

For the test instances a.1-a.5, b.1-b.5, c.1-c.5 and d.1-d.5 from Beasley's OR-Library we received results similar to those in Table 2, and believe that for these seven groups of problems a robust interval for the start iteration, r, is $100 \leq r \leq 300$.

Hence, our recommendation is to choose the start iteration r in the interval

$$200 \leq r \leq 300. \tag{18}$$

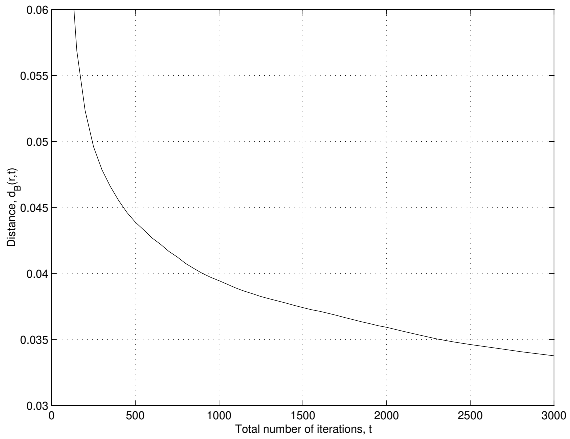

Figure 3: *Illustration of the dependence between the total number of iterations, t, for the computation of the ergodic sequence and the distance $d_B(r,t)$, for test problems 6.1-6.5 (see Table 1). Here r $= 24$.*

Regarding the total number of subgradient iterations to run we decided, that if the distance has not decreased more than a predefined tolerance, δ, in a prespecified number of iterations, τ, we stop the subgradient procedure. The stop iteration $T(\tau, \delta)$ is defined as

$$T(\tau, \delta) = \min\{t \mid d_B(r, t-\tau) - d_B(r,t) \leq \delta, t > \tau\}, \quad \tau, \delta \geq 0. \tag{19}$$

Table 4: *The stop iteration, $T(\tau, \delta)$, for each group of problems computed according to (19). Here, $\tau = 200$ and $\delta = 0.0005$.*

Prob.	Size	Dens. (%)	r	$T(\tau, \delta)$	$d_B(r, T)$
4.1-10	200x1000	2	18	1600	0.0533
5.1-10	200x2000	2	22	2000	0.0184
6.1-5	200x1000	5	24	1600	0.0353
a.1-5	300x3000	2	21	1850	0.0202
b.1-5	300x3000	5	33	1550	0.0176
c.1-5	400x4000	2	23	1150	0.0225
d.1-5	400x4000	5	32	1100	0.0122
sj01-05	429x42148	1.69	56	2100	0.0172

We choose $\tau = 200$ and $\delta = 0.0005$.

For each group of problems we computed the dependence between the stop iteration, t, and the distance, $d_B(r, t)$, and by applying the criterion (19) we found the total number of iterations, $T(\tau, \delta)$, to run in the subgradient procedure (see Table 4).

Table 4 gives us an over-all interval for the total number of iterations to run in the subgradient scheme, specifically

$$1600 \leq T \leq 2000. \tag{20}$$

4.3 Discussion concerning tolerances

The size of the core problem, and the quality of the resulting solution, is dependent of the choice of the upper and lower tolerance, ϵ_0 and ϵ_1. Different choices of tolerances will lead to different core problems and, thereby, different solutions. Therefore, we have to face the trade off between the size of the core problem and the quality of its solution. The quality of the solution is measured according to (17). There are different advantages for the respective choices, a small core problem will generate a feasible, probably non-optimal, solution in a small amount of time (preferable when a solution is needed quickly) and a larger core problem produces a solution of higher quality with a considerable computational effort.

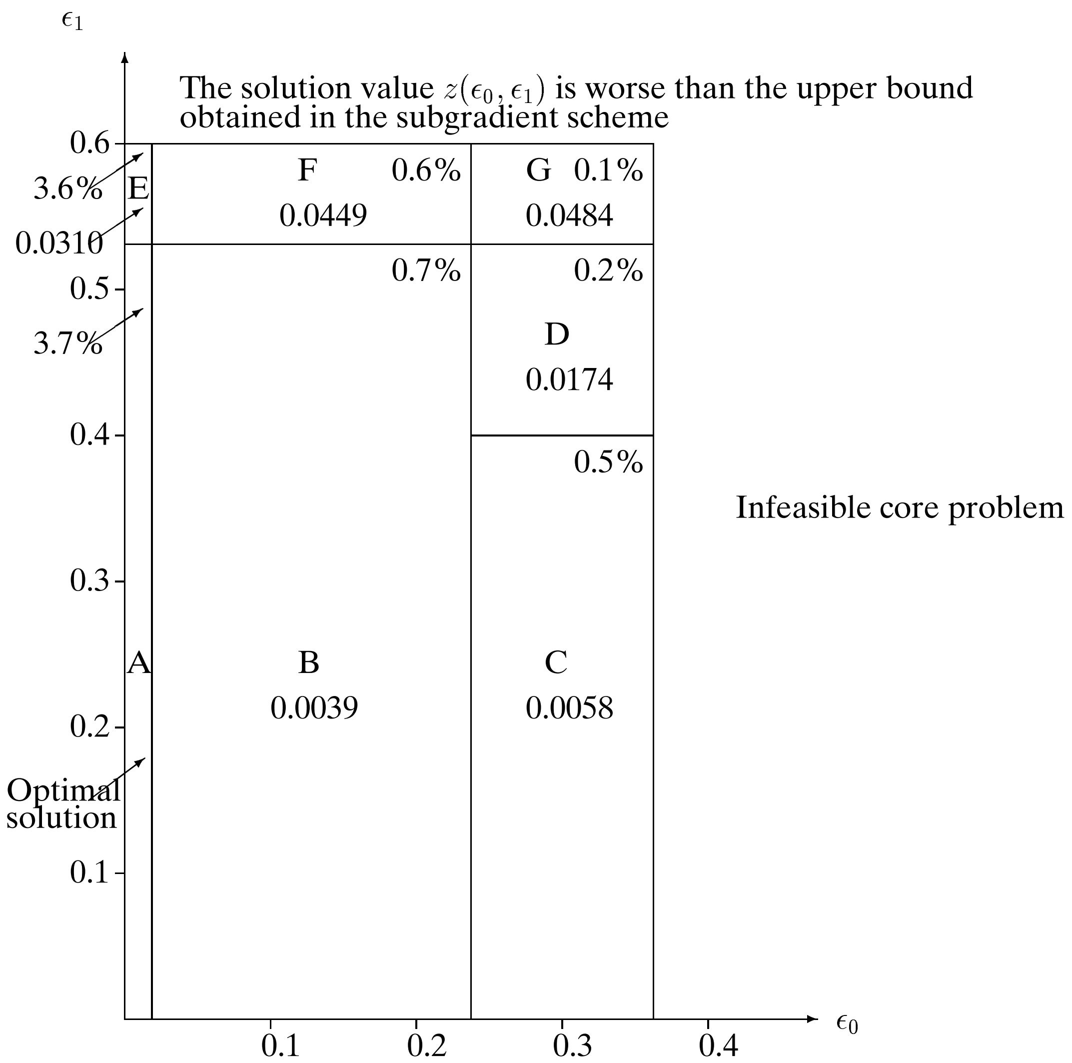

Figure 4: *Illustration of the trade off between the size of the core problem and the quality of the solution, depending on the choice of the upper and lower tolerance, for test problem 4.3.*

Figure 4 is an illustration of the trade off between the size of the core problem and the quality of the solution, depending on the choice of the upper and lower tolerances. Each box represents a solution value. The quality of the solution, $(z(\epsilon_0, \epsilon_1) - \underline{z}^s)/\underline{z}^s$, is written in the middle of each box. The number in the upper right corner of each box is the size of the smallest core problem, in percentage of the original problem size, yielding the corresponding solution value $z(\epsilon_0, \epsilon_1)$. We can see that there are maximum "feasible" values for both the upper and lower tolerances. If we fix too many variables to zero, the core problem will become infeasible (for example, if all variables in a row are put to zero it is impossible to cover that row). If we fix too many variables to one, the solution value will become worse than the upper bound obtained in the subgradient procedure. Box A is the "optimal" box since we will obtain the optimal solution for all choices of tolerances within this box. If we like to obtain the optimal solution we have to choose ϵ_0 and ϵ_1 somewhere in the intervals defining the box A. The smallest core problem that is possible to receive within the "optimal" box is when the tolerances are corresponding to the upper right corner, the "optimal corner", the upper/rightmost endpoints of the interval for box A. For the problem in Figure 4, the size of the smallest core problem giving the optimal solution is 3.7% of the size of the original problem. Every problem has it's own "optimal" box. These boxes have different sizes and shapes. The smallest core problem possible to construct still yielding a feasible solution corresponds to the upper right corner in box G. For this case the size of the smallest possible core problem is 0.1% of the size of the original problem.

For example, in Figure 4, more variables are fixed to zero in box B than in box A. Since the solution value in box B is greater than in box A, at least one variable that received the value one in the core problem solution in box A, is fixed to zero in box B, and by that it is not included in the core problem corresponding to box B. The solution in box B then has to select a "second best" variable in the core (problem) to obtain a feasible solution, leading to a higher solution value.

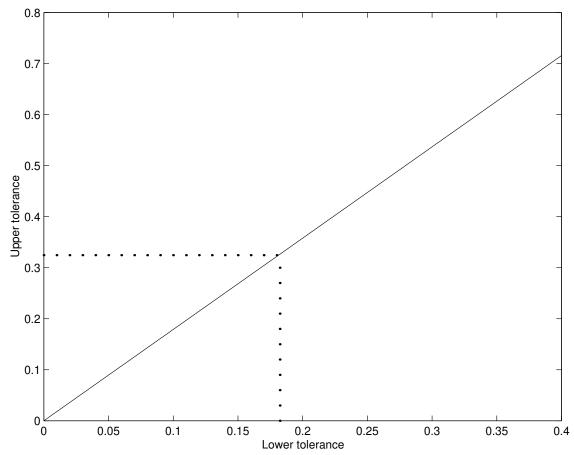

Figure 5: *A linear dependence between the upper and lower tolerance, decided by an average "optimal corner" for 25 test problems,* $\tilde{\epsilon}_0 = 0.18$ *and* $\tilde{\epsilon}_1 = 0.33$.

In order to compare the quality of the solution and the size of the core problem for different choices of tolerances we decide an average "optimal corner" for 25 test problems. This average corresponds to $\epsilon_0 = 0.18$ and $\epsilon_1 = 0.33$. We assume a linear dependence between the upper and lower tolerance, see Figure 5, resulting in the function,

$$\epsilon_1 = \epsilon_1(\epsilon_0) = 1.8\epsilon_0. \tag{21}$$

Given a lower tolerance, the above relationship (21) gives the upper tolerance.

4.4 Core problems created from different LP solutions

We generate core problems with different solutions to (LP); $\hat{x}^t$, x_S^* and x_B^* and then compare the size of the core with the quality of the solution.

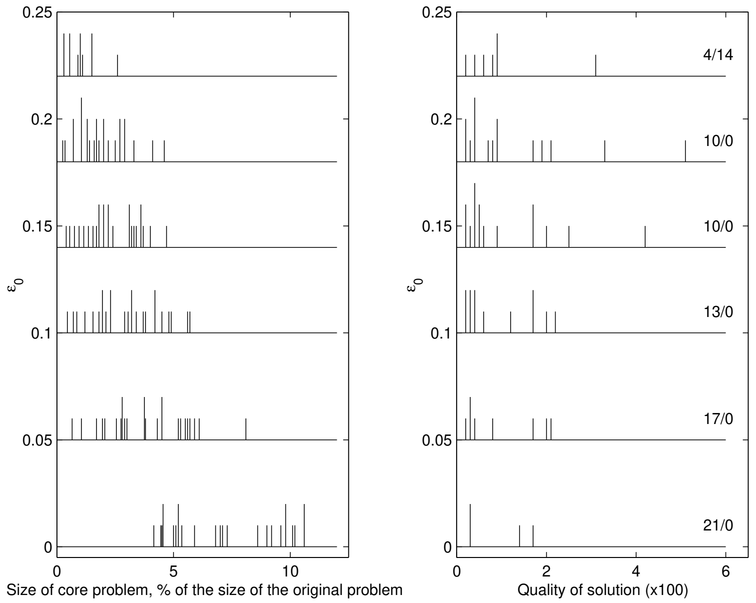

Figure 6: *The histograms illustrate the trade off between the size of the core problem and the quality of its solution when the core problem is created from the ergodic iterate. On the vertical axis is the lower tolerance. On the horizontal axis is the size of the core problem in percentage of the size of the original problem and the quality of the solution, respectively. Totally 25 test problems.*

Figure 6 illustrates how the size of the core problem and the quality of its solution vary with the tolerances when the core problem is created from the ergodic iterate. We solve the core problem exactly using the integer programming solver provided within Cplex.

For each choice of the lower tolerance in Figure 6, we have examined 25 test problems. A small column at respective lower tolerance represents a result for a test problem. For example, when the lower tolerance is $\epsilon_0 = 0.05$ one test problem generates a core problem with a size that is 8$\%$ of the original problem. When a column is twice as high as a small column, it means that two test problems generated the same result. The numbers to the right at the horizontal axis in the histograms denote the number of problems solved to optimality/number of infeasible core problems with the corresponding choices of tolerances. For example, when $\epsilon_0 = 0.22$ (and $\epsilon_1 = 0.39$), 14 out of 25 test problems have an infeasible core problem and the optimal solution is found for 3 of the 11 feasible problems. If choosing the upper and lower tolerance as $\epsilon_0 = 0.05$ and $\epsilon_1 = 0.09$, the optimal solution is found for 17 of the 25 test problems, no core problems are infeasible and the size of the core problems are between 0.7$\%$ - 8.1$\%$ of the size of the original problem.

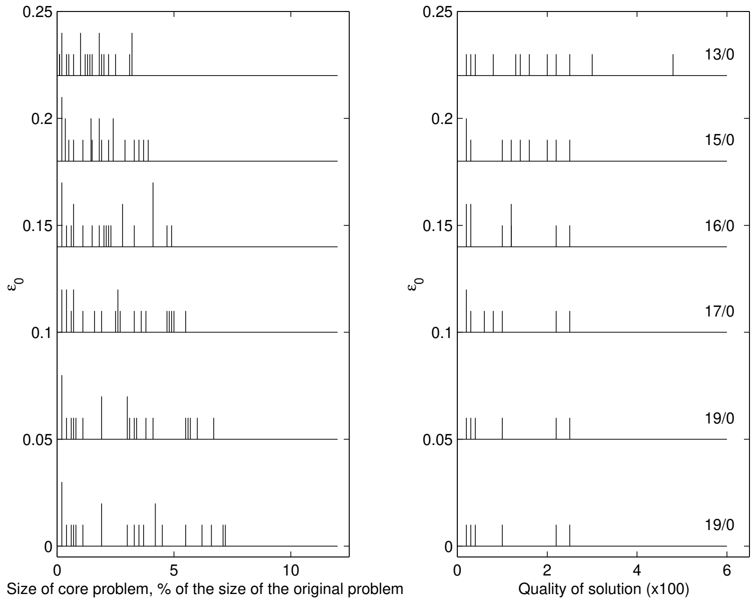

Figure 7: *The trade off between the size of the core problem and the quality of its solution when the core problem is created from x_B^*. Totally 25 test problems. See Figure 6 for notations.*

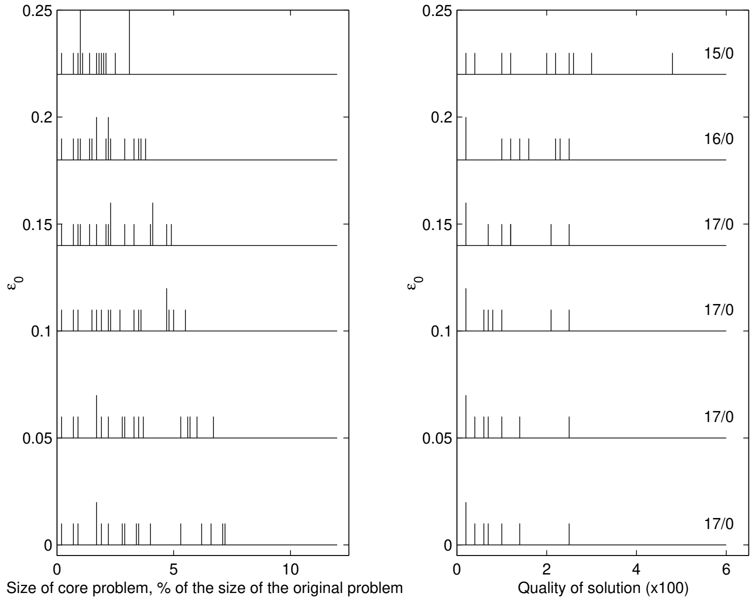

Figure 8: *The trade off between the size of the core problem and the quality of its solution when the core problem is created from x_S^*. Totally 25 test problems. See Figure 6 for notations.*

The corresponding results when creating the core problem from x_S^* and x_B^* are slightly better than those from a core problem constructed from the ergodic iterate; see Figure 6, 7 and 8. For example, when $\epsilon_0 = 0.18$ (and $\epsilon_1 = 0.33$) the ergodic iterate generates the optimal solution for only 10 problems while x_S^* and x_B^* generate the optimal solution for 16 respective 15 problems.

4.5 Final results

We have solved 65 problems with our algorithm and for 34 problems we found the optimal solution or the best solution known in the literature, see Table 6, 8 and 10. The test were made with $r = 200$, $t = 2000$, $\epsilon_0 = 0.08$ and $\epsilon_1 = 0.14$.

In [CFT99] they present the results for four different algorithms. These algorithms have been tested on the same problems as we have, that is, test instances from Beasley's OR Library. Two of these algorithms, a genetic algorithm by Beasley and Chu and the algorithm presented in [CFT99], find

Table 5: *Test data for problems 4.1-4.10, 5.1-5.10 and 6.1-6.5 from Beasley's OR-library.*

Prob.	Size	Dens. (%)	z^*	z^*_{LP}	$100 \cdot (z^* - z^*_{LP})/z^*$
4.1	200x1000	2	429	429.0	0.0
4.2	200x1000	2	512	512.0	0.0
4.3	200x1000	2	516	516.0	0.0
4.4	200x1000	2	494	494.0	0.0
4.5	200x1000	2	512	512.0	0.0
4.6	200x1000	2	560	557.3	0.5
4.7	200x1000	2	430	430.0	0.0
4.8	200x1000	2	492	488.7	0.7
4.9	200x1000	2	641	638.5	0.4
4.10	200x1000	2	514	513.5	0.1
5.1	200x2000	2	253	251.2	0.7
5.2	200x2000	2	302	299.8	0.7
5.3	200x2000	2	226	226.0	0.0
5.4	200x2000	2	242	240.5	0.6
5.5	200x2000	2	211	211.0	0.0
5.6	200x2000	2	213	212.5	0.2
5.7	200x2000	2	293	291.8	0.4
5.8	200x2000	2	288	287.0	0.3
5.9	200x2000	2	279	279.0	0.0
5.10	200x2000	2	265	265.0	0.0
6.1	200x1000	5	138	133.1	3.5
6.2	200x1000	5	146	140.5	3.8
6.3	200x1000	5	145	140.1	3.4
6.4	200x1000	5	131	129.0	1.5
6.5	200x1000	5	161	153.4	4.8

the optimal solution in almost every case. The other two algorithms, one by Balas and Carrera and one by Beasley, find the optimal solution for 50% and 30% of the problems, respectively.

The run times for respective problems are probably a bit unfair since our implementation is not optimized in any way; it could probably be speeded up a lot. Despite that, we may compare the run times for the test problems. In Table 10 we notice the differences between the size of the core problems and the impact that have on the run times; a small core problem yields a shorter run time.

Table 6: *The results when we solved 4.1-4.10, 5.1-5.10 and 6.1-6.5 from Beasley's OR-library with our algorithm (Size, the size of the core problem in percentage of the size of the original problem and Time, the total run time in CPU-seconds).*

Prob.	z(ϵ_0, ϵ_1)	$\underline{z}^t$	$100 \cdot (z^* - \underline{z}^t)/z^*$	Size (%)	Time (s)
4.1	429	429.0	0.0	3.9	3.7
4.2	512	512.0	0.0	7.5	1.3
4.3	519	516.0	0.0	6.4	0.7
4.4	495	493.8	0.04	4.8	3.7
4.5	512	512.0	0.0	6.0	3.2
4.6	561	557.2	0.5	3.7	3.8
4.7	430	429.9	0.02	2.6	3.7
4.8	492	488.4	0.7	4.5	3.8
4.9	641	638.3	0.4	5.8	3.7
4.10	514	513.5	0.1	1.1	3.7
5.1	254	251.0	0.8	2.6	7.1
5.2	305	299.0	1.0	3.7	7.1
5.3	226	226.0	0.0	0.4	7.1
5.4	247	240.5	0.6	2.3	7.0
5.5	211	211.0	0.0	0.8	6.9
5.6	213	212.5	0.2	0.6	7.0
5.7	294	290.8	0.8	2.6	7.0
5.8	289	286.7	0.5	3.5	7.1
5.9	279	278.1	0.3	1.9	7.0
5.10	265	265.0	0.0	0.3	7.1
6.1	143	130.3	5.6	5.6	7.2
6.2	149	138.8	4.9	5.6	7.3
6.3	145	138.8	4.3	5.2	7.3
6.4	131	128.6	1.8	3.5	7.1
6.5	161	151.8	5.7	5.3	7.4

We have tested the algorithm on unicost problems. A unicost problem is a problem where every pairing has equal cost, i.e. $c_j = c$ for all $j, j = 1, \dots, n$, where c is a constant value. Also, in the unicost problems we tested there

Table 7: *Test data for problems a.1-a.5, b.1-b.5, c.1-c.5, d.1-d.5, e.1-e.5, f.1-f.5 and g.1-g.5 from Beasley's OR-library. For the problems e.2 and g.1-g.5 z^* and z^*_{LP} denotes the best solution and the lower bound known in literature [CFT99], respectively.*

Prob.	Size	Dens. (%)	z^*	z^*_{LP}	$100 \cdot (z^* - z^*_{LP})/z^*$
a.1	300x3000	2	253	246.8	2.5
a.2	300x3000	2	252	247.5	1.8
a.3	300x3000	2	232	228.0	1.7
a.4	300x3000	2	234	231.4	1.1
a.5	300x3000	2	236	234.9	0.5
b.1	300x3000	5	69	64.5	6.5
b.2	300x3000	5	76	69.3	8.8
b.3	300x3000	5	80	74.2	7.3
b.4	300x3000	5	79	71.2	9.9
b.5	300x3000	5	72	67.7	6.0
c.1	400x4000	2	227	223.8	1.4
c.2	400x4000	2	219	212.9	2.8
c.3	400x4000	2	243	234.6	3.5
c.4	400x4000	2	219	213.9	2.3
c.5	400x4000	2	215	211.6	1.6
d.1	400x4000	5	60	55.3	7.8
d.2	400x4000	5	66	59.4	10.0
d.3	400x4000	5	72	65.1	9.6
d.4	400x4000	5	62	55.9	9.8
d.5	400x4000	5	61	58.6	3.9
e.1	500x5000	10	29	21.4	26.2
e.2	500x5000	10	30	28	6.7
e.3	500x5000	10	27	20.5	24.1
e.4	500x5000	10	28	21.4	23.6
e.5	500x5000	10	28	21.3	23.9
f.1	500x5000	20	14	8.8	37.1
f.2	500x5000	20	15	10.0	33.3
f.3	500x5000	20	14	9.5	32.1
f.4	500x5000	20	14	8.5	39.3
f.5	500x5000	20	13	7.8	40.0
g.1	1000x10000	2	176	165	6.3
g.2	1000x10000	2	154	147	4.5
g.3	1000x10000	2	166	153	7.8
g.4	1000x10000	2	168	154	8.3
g.5	1000x10000	2	168	153	8.9

Table 8: *The results when we solved problems a.1-a.5, b.1-b.5, c.1-c.5, d.1-d.5, e.1-e.5, f.1-f.5 and g.1-g.5 from Beasley's OR-library with our algorithm.*

Prob.	$z(\epsilon_0, \epsilon_1)$	$\underline{z}^t$	$100 \cdot (z^* - \underline{z}^t)/z^*$	Size (%)	Time (s)
a.1	255	244.9	3.2	3.2	15.2
a.2	252	246.8	2.1	3.1	14.7
a.3	233	227.5	1.9	3.1	14.6
a.4	236	229.7	1.8	2.0	14.3
a.5	239	233.4	1.1	1.9	14.4
b.1	70	62.3	9.7	2.1	30.5
b.2	76	68.1	10.4	2.3	31.2
b.3	81	72.4	9.5	2.0	30.2
b.4	81	68.9	12.8	2.3	31.4
b.5	74	66.0	8.3	2.1	30.5
c.1	228	222.6	1.9	2.6	24.5
c.2	221	211.8	3.3	2.8	25.3
c.3	244	233.5	3.9	2.9	25.7
c.4	220	213.0	2.7	2.6	24.3
c.5	216	210.8	2.0	2.6	24.0
d.1	61	53.8	10.3	2.0	54.4
d.2	66	57.5	12.9	2.0	54.9
d.3	73	61.2	15.0	2.0	53.5
d.4	65	53.5	13.7	2.0	61.8
d.5	61	57.6	5.6	1.7	52.4
e.1	29	18.4	36.6	1.1	161.5
e.2	30	18.7	37.7	2.8	18581.2
e.3	27	16.8	37.8	2.3	1124.5
e.4	28	17.5	37.5	2.3	936.6
e.5	28	18.7	33.2	1.1	159.9
f.1	14	6.7	52.1	2.3	1179.1
f.2	15	7.5	50.0	2.0	628.7
f.3	14	7.3	47.9	2.5	899.8
f.4	14	6.5	53.6	2.7	4418.7
f.5	13	6.1	53.1	2.4	1461.5
g.1	179	155.6	11.6	2.0	461.3
g.2	156	137.3	10.8	1.9	227.1
g.3	169	144.2	13.1	2.0	1989.4
g.4	172	144.5	14.0	1.9	1754.5
g.5	171	143.8	14.6	2.0	1309.4

where equally many ones in every column in A. This construction of the problem results in that all variables are equally frequent in $\hat{x}^t$; $\hat{x}^t_j = k$ for all j, $j = 1, \ldots, n$, where k is constant between zero and one. Therefore, we are not able to make any predictions about any variable x_j. Hence, the core problem is equal the original problem and no improvement in size have been made.

Table 9: *Test data for h.1-h.5 from Beasley's OR-library. Here, z^* and z^*_{LP} denotes the best solution and the lower bound known in literature [CFT99], respectively.*

Prob.	Size	Dens. (%)	z^*	z^*_{LP}	$100 \cdot (z^* - z^*_{LP})/z^*$
h.1	1000x10000	5	63	52	17.5
h.2	1000x10000	5	63	52	17.5
h.3	1000x10000	5	59	48	18.6
h.4	1000x10000	5	58	47	19.0
h.5	1000x10000	5	55	46	16.4

Table 10: *The results when we solved h.1-h.5 from Beasley's OR-library with our algorithm. Here, $\epsilon_0 = 0.18$ and $\epsilon_1 = 0.32$.*

Prob.	$z(\epsilon_0, \epsilon_1)$	$\underline{z}^t$	$100 \cdot (z^* - \underline{z}^t)/z^*$	Size (%)	Time (s)
h.1	63	40.6	35.6	1.4	27456.8
h.2	63	41.1	34.8	1.5	163120.0
h.3	59	37.4	36.6	1.5	114418.4
h.4	58	36.8	36.6	1.3	83662.8
h.5	55	34.6	37.1	1.3	9037.5

The slow run time for h.1-h.5 in Table 10 indicates that the resulting core problem is hard to solve with the optimizer even though the original problem is considerably reduced in size.

In some cases the lower bound generated with our algorithm, $\underline{z}^t$, is poor compared with z^*_{LP}. That indicates that the step length we have chosen in the subgradient procedure not are the most efficient.

5 Conclusions

The main conclusion is that this is a applicable method for solving large scale 0-1 minimization problems, excluded unicost problems. Since there are many parameters within the algorithm, for example the tolerances, the start and stop iteration for computing the ergodic iterate and the step length in the subgradient procedure, that need to be calibrated individually and together we believe that the algorithm needs som further testing and development to generate better results faster.

The ergodic iterate has shown to be almost as good to use as an optimal solution to (LP) from the Simplex or the Barrier algorithm when predicting whether a variable is included in the optimal solution or not.

Even though we seem to have reduced the original problem considerably in size the resulting core problem still can be hard to solve with the optimizer, and that have a impact on the run time.

References

[AHKW] E. Andersson, E. Housos, N. Kohl, and D. Wedelin. *Crew pairing optimization*, pages 1–31. OR in Airline Industry. Kluwer Academic Publishers, Boston.

[Bea90] J.E. Beasley. A Lagrangean heuristic for set covering problems. *Naval Research Logistics*, 37:151–164, 1990.

[Bea93] J. E. Beasley. *Lagrangean relaxation*, pages 243–303. Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publications, Oxford, 1993.

[CFT97] A. Caprara, M. Fischetti, and P. Toth. Algorithms for railway crew management. *Mathematical Programming*, 79:125–141, 1997.

[CFT99] A. Caprara, M. Fischetti, and P. Toth. A Heuristic method for the set covering problem. *Operations Research*, 47:730–743, 1999.

[CNS98] S. Ceria, P. Nobili, and A. Sassano. A Lagrangian-based heuristic for large-scale set covering problems. *Mathematical Programming*, 81:215–228, 1998.

[DJ79] Y. M. I. Dirickx and L. P. Jennergren. *Systems analysis by multilevel methods*. Wiley, Chichester, 1979.

[Fis81] M. L. Fisher. The Lagrangean relaxation method for solving integer programming problems. *Management Science*, 27:1–18, 1981.

[Geo74] A. M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[HWC74] M. Held, P. Wolfe, and H. P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 16:62–88, 1974.

[Kar72] R. M. Karp. *Reducibility among combinatorial problems*. Complexity of Computer Computations. Plenum Press, New York, 1972.

[LL97] T. Larsson and Z. Liu. A Lagrangean relaxation scheme for structured linear programs with application to multicommodity network flow. *Optimization*, 40:247–284, 1997.

[LPS] T. Larsson, M. Patriksson, and A.-B. Strömberg. Ergodic primal solutions from dual subgradient schemes in discrete optimization.

[LPS96] T. Larsson, M. Patriksson, and A.-B. Strömberg. Conditional subgradient optimization - Theory and applications. *European Journal of Operational Research*, 88:382–403, 1996.

[LPS99] T. Larsson, M. Patriksson, and A.-B. Strömberg. Ergodic, primal convergence in dual subgradient schemes for convex programming. *Mathematical Programming*, 86:283–312, 1999.

[Pol69] B. T. Polyak. Minimization of unsmooth functionals. *USSR Computational Mathematics and Mathematical Physics*, 9:14–29, 1969.

[Wed95] D. Wedelin. An Algorithm for large scale 0-1 integer programming with application to airline crew scheduling. *Annals of Operations Research*, 57:283–310, 1995.