MASTER'S THESIS

Recovery of primal solutions from dual subgradient methods for mixed binary linear programming; a branch-and-bound approach

PAULINE ALDENVIK
MIRJAM SCHIERSCHER

Department of Mathematical Sciences
Division of Mathematics
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2015

Thesis for the Degree of Master of Science

Recovery of primal solutions from dual subgradient methods for mixed binary linear programming; a branch-and-bound approach

Pauline Aldenvik
Mirjam Schierscher

Department of Mathematical Sciences
Division of Mathematics
Chalmers University of Technology and University of Gothenburg
SE – 412 96 Gothenburg, Sweden
Gothenburg, September 2015

Matematiska vetenskaper
Göteborg 2015

Abstract

The main objective of this thesis is to implement and evaluate a Lagrangian heuristic and a branch-and-bound algorithm for solving a class of mathematical optimization problems called mixed binary linear programs. The tests are performed on two different types of mixed binary linear programs: the set covering problem and the (uncapacitated as well as capacitated) facility location problem.

The purpose is to investigate a concept rather than trying to achieve good runtime performance. The concept involves ergodic iterates, which are convex combinations of all Lagrangian subproblem solutions found so far. The ergodic iterates are constructed from the Lagrangian subproblem solutions with different convexity weight rules.

In the Lagrangian heuristic, Lagrangian relaxation is utilized to obtain lower bounds on the optimal objective value and the ergodic iterates are used to create feasible solutions, and whence, to obtain upper bounds on the optimal objective value. The branch-and-bound algorithm uses the Lagrangian heuristic in each node and the ergodic iterates for branching decisions.

The investigated concept of this thesis is ergodic iterates constructed by different convexity weight rules, where the different rules are to weigh the Lagrangian subproblem solutions as follows: put all the weight on the last one (the traditional Lagrangian heuristic), use equal weights on all, and put a successively higher weight on the later ones.

The result obtained shows that a convexity weight rule that puts more weight on later Lagrangian subproblem solutions, without putting all the weight on the last one, is preferable.

Keywords: Branch-and-bound method, subgradient method, Lagrangian dual, recovery of primal solutions, ergodic sequence, mixed binary linear programming, set covering, facility location

Acknowledgements

We would like to thank our supervisor Emil Gustavsson at the Department of Mathematical Sciences at the University of Gothenburg for his help and support.

Pauline Aldenvik and Mirjam Schierscher
Gothenburg, June 2015

Contents

1 Introduction

In this section the subject of this thesis is introduced and briefly explained. Furthermore, the aims and limitations of the thesis are described and the outline of this report are presented.

This thesis deals with *mixed binary linear programs* (MBLP) which are a class of problems in *mathematical optimization*. Mathematical optimization is about finding an *optimal* solution, i.e., an in some well-defined sense best solution, to a given problem. An optimization problem consists of an *objective function*, for which the maximum or minimum *objective value* is wanted, and constraints on the simultaneous choices of values of these variables. Such a problem could be to minimize the cost or time for manufacturing certain objects and at the same time fulfill the demands of the clients.

The optimal objective value is the minimum of $f(\mathbf{x})$ for $\mathbf{x} \in X$, where $f : X \mapsto \mathbb{R}$ is a function from the set X to the set of the real numbers. The set of vectors satisfying the constraints of the problem defines the set X. A (suggested) solution $\mathbf{x}$ to the optimization problem is said to be *feasible* when $\mathbf{x} \in X$. Hence, X is referred to as the *feasible set*. For a solution to be *optimal*, it has to be feasible. An optimization problem can be defined as follows:

$$\text{minimize} \quad f(\mathbf{x}), \tag{1.1a}$$

$$\text{subject to} \quad \mathbf{x} \in X, \tag{1.1b}$$

where (1.1a) declares the objective: to minimize the objective function value, and (1.1b) describes the feasible set, i.e., the constraints. An optimal solution to the problem and its corresponding objective value is denoted $\mathbf{x}^*$ and f^*, respectively.

There exist different classes of optimization problems as mentioned above. If the objective function and the constraints are linear the problem is called a linear program (LP). If the objective function and the feasible set are convex it is a convex optimization problem. Then there are integer programs (IP) or, as mentioned, MBLPs. In an integer optimization problem, the variables are restricted to be integer or binary decision variables. In a mixed binary linear problem some variables are restricted to be binary while others are not. These problems can sometimes be very large and hard to solve,

but by applying different methods and algorithms they are made easier and solvable.

If the problem is hard to solve because of one or several constraints, then these constraints can be relaxed by using *Lagrangian relaxation*. This is used to create a relaxed problem which is easier than the original one. The relaxed problem can be solved by a *subgradient method* and its solution provides valuable information, e.g., a *bound* on the optimal solution to the original problem.

One method for solving integer programming problems is the *branch-and-bound method*, where the original problem is divided into smaller and smaller subproblems by fixing integer variables one at a time. The idea is to perform an exhaustive search, i.e., examine all solutions, without actually having to generate all solutions.

In this work two algorithms to solve the optimization problems are implemented and evaluated. One algorithm is a Lagrangian heuristic. It performs a subgradient method and utilizes the information obtained together with ergodic iterates to create feasible solutions. The other algorithm builds a branch-and-bound tree. At each node in the branch-and-bound tree the Lagrangian heuristic is applied to calculate a lower bound and an upper bound, respectively. The problems used to evaluate the algorithms are mixed binary linear programming problems.

1.1 Background

Integer programming problems are well-studied in the literature; they appear in production planning, scheduling, network flow problems and more. Many MBLPs are hard to solve and have been studied a lot. For comprehensive analysis, see, e.g., Wolsey and Nemhauser [20], Wolsey [19] and Lodi [16].

Difficult problems, where some constraints are complicated, can be solved with Lagrangian relaxation. This has been studied and developed a lot over the years by, e.g., Rockafellar [18, Part 6], Everett [8] and Fisher [9].

A common approach to solve integer programming problems is the branch-and-bounds method, which was introduced by Land and Doig [14] and Dakin [7]. A branch-and-bound with Lagrangian heuristic in the nodes has been studied by, e.g., Borchers and Mitchell [4], Fisher [9] and Görtz and Klose [11].

The construction of ergodic iterates, a sequence of primal vectors obtained from the Lagrangian dual subgradient method, and their convergence as well as some implementation has been studied by Gustavsson, Patriksson and Strömberg in [12]. Using ergodic iterates in a subgradient method to create a feasible solution has been studied by Gustavsson, Larsson, Patriksson, and Strömberg [13] where they present a framework for a

branch-and-bound algorithm with ergodic iterates.

1.2 Aims and limitations

The purpose of this theses is to study a Lagrangian heuristic with ergodic iterates, where the ergodic iterates are weighted according to different convexity weight rules, and to investigate a branch-and-bound method in which ergodic iterates are utilized for branching decision and for finding primal feasible solutions, i.e., to implement and test the third procedure of primal recovery described in Gustavsson et al. [13].

This work is restricted to investigate a concept; trying to achieve good runtime performance is excluded. The algorithms are tested on no other problem types than Facility location problems and Set covering problems.

1.3 Outline

In Section 2 the theory and concepts needed to understand the algorithms and the following analysis are described. It includes, for instance, general descriptions of MBLPs and the branch-and-bound method, how to calculate lower bounds with Lagrangian relaxation and the subgradient method, and how to create ergodic iterates from Lagrangian subproblem solutions. A small example is provided to visualize some of the concepts.

The algorithms implemented are found in Section 3. They are, in this work, used on the different types of MBLPs presented in Section 4. The test results of the algorithms are described in Section 5. Finally, the result is discussed, and the advantages and drawbacks of the investigated algorithms are pointed out. This is, together with proposed future research, located in Section 6.

2 Theory

The intention of this thesis is to study a method for solving optimization problems. Each problem studied belongs to a problem class called mixed binary linear programs (MBLPs). If the objective function is linear, the constraints are affine, and there are only continuous variables, then it is a linear program. However, if there are binary restrictions on some of the variables, there is a mixture of both binary and continuous variables, and it is therefore called a mixed binary linear program. A general MBLP can be defined as the problem to find

$$\begin{aligned} z^* = \text{minimum} \quad & \mathbf{c}^\intercal \mathbf{x} && \text{(2.1a)} \\ \text{subject to} \quad & A\mathbf{x} \geq \mathbf{b}, && \text{(2.1b)} \\ & \mathbf{x} \in X, && \text{(2.1c)} \end{aligned}$$

where $\mathbf{c} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$. The set $X = \{D\mathbf{x} \geq \mathbf{e}$ and $x_i \in \{0,1\},\ i \in \mathcal{I}\}$, where $\mathcal{I} \subseteq \{1,\ldots,n\}$, and X is assumed to be compact. Furthermore $D \in \mathbb{R}^{k \times n}$ and $\mathbf{e} \in \mathbb{R}^k$. The following is a small example of a MBLP which we will utilize throughout this report:

$$\begin{aligned} z^* = \min \quad & x_1 + 2x_2 + 2x_3, && \text{(2.2a)} \\ \text{s.t.} \quad & 2x_1 + 2x_2 + 2x_3 \geq 3, && \text{(2.2b)} \\ & x_1, x_2, x_3 \in \{0,1\}, && \text{(2.2c)} \end{aligned}$$

where the objective is to minimize the function, $z(\mathbf{x}) = x_1 + 2x_2 + 2x_3$, subject to the linear constraint (2.2b) and the binary restrictions on x_1, x_2 and x_3 (2.2c). The optimal objective value z^* to this problem is 3 and a corresponding optimal solution $\mathbf{x}^*$ is $(1,0,1)$ or $(1,1,0)$.

The remainder of this section includes the theory concerning this work, namely, Lagrangian duality, a subgradient method for solving the Lagrangian dual problem, how to generate sequences of primal vectors, and the branch-and-bound method.

2.1 Lagrangian duality

Lagrangian relaxation can be used to relax complicating constraints, which generates an easier problem than the original one. The original problem is

called the *primal problem*. The easier problem is referred to as the *Lagrangian dual problem* or just the *dual problem*. Furthermore, the objective value of a solution to the dual problem is an optimistic estimate of the objective value corresponding to a primal optimal solution. Hence, it can be used to evaluate the quality of a primal feasible solution.

Hereinafter, let the primal problem be the minimization problem in (2.1). Introduce Lagrangian multipliers, $\mathbf{u} \in \mathbb{R}^m_+$. For each $i = 1, ..., m$ the Lagrangian multiplier, u_i, corresponds to the linear constraint $\mathbf{a}_i^\intercal \mathbf{x} \geq b_i$, where $\mathbf{a}_i$ is row vector i in A. Lagrangian relax by removing the constraints and add them to the objective function with the Lagrangian multipliers as penalty parameters. This yields the *Lagrange function*,

$$\mathcal{L}(\mathbf{x}, \mathbf{u}) = \mathbf{c}^\intercal \mathbf{x} + \sum_{i=1}^{m} (b_i - \mathbf{a}_i^\intercal \mathbf{x}) u_i = \mathbf{c}^\intercal \mathbf{x} + (\mathbf{b} - A\mathbf{x})^\intercal \mathbf{u},$$

which is used to formulate the *Lagrangian dual function*:

$$q(\mathbf{u}) = \min_{\mathbf{x} \in X} \mathcal{L}(\mathbf{x}, \mathbf{u}) = \mathbf{b}^\intercal \mathbf{u} + \min_{\mathbf{x} \in X} (\mathbf{c} - A^\intercal \mathbf{u})^\intercal \mathbf{x}, \quad \mathbf{u} \in \mathbb{R}^m. \tag{2.3}$$

The *Lagrangian subproblem* at $\mathbf{u}$ is identified as the problem

$$\min_{\mathbf{x} \in X} (\mathbf{c} - A^\intercal \mathbf{u})^\intercal \mathbf{x}, \tag{2.4}$$

with the solution set denoted $X(\mathbf{u})$.

As mentioned, the dual objective value $q(\mathbf{u})$, where $\mathbf{u} \in \mathbb{R}^m_+$, is an optimistic estimate to the primal objective value. For a minimization problem that is a lower bound. Since the best possible lower bound is wanted, the dual problem is to find

$$q^* = \text{supremum} \;\; q(\mathbf{u}), \tag{2.5a}$$
$$\text{subject to} \qquad \mathbf{u} \in \mathbb{R}^m_+. \tag{2.5b}$$

The dual function, q, is concave and the feasible set, $\mathbf{u} \in \mathbb{R}^m_+$, is convex. Hence, the problem (2.5) is a convex optimization problem.

Theorem 1 (Weak duality). *Assume that* $\mathbf{x}$ *and* $\mathbf{u}$ *are feasible in the problem (2.1) and (2.5), respectively. Then, it holds that*

$$q(\mathbf{u}) \leq \mathbf{c}^\intercal \mathbf{x},$$

and, in particular,

$$q^* \leq z^*.$$

Proof. For all $\mathbf{u} \geq \mathbf{0}^m$ and $\mathbf{x} \in X$ with $A\mathbf{x} \geq \mathbf{b}$,

$$q(\mathbf{u}) = \min_{\mathbf{y} \in X} \mathcal{L}(\mathbf{y}, \mathbf{u}) \leq \mathcal{L}(\mathbf{x}, \mathbf{u}) = \mathbf{c}^\intercal \mathbf{x} + (\mathbf{b} - A\mathbf{x})^\intercal \mathbf{u} \leq \mathbf{c}^\intercal \mathbf{x},$$

so

$$q^* = \max_{\mathbf{u} \geq \mathbf{0}^m} q(\mathbf{u}) \leq \min_{\mathbf{x} \in X : A\mathbf{x} \geq \mathbf{b}} \mathbf{c}^\intercal \mathbf{x} = z^*.$$

□

Weak duality holds, but strong duality ($q^* = z^*$) does not hold for the general case since X is non-convex in general. A convex version of the primal problem is the one in which X is replaced by its convex hull, conv X, i.e., the problem to find

$$z^*_{\text{conv}} = \text{minimum } \mathbf{c}^\intercal \mathbf{x}, \tag{2.6a}$$

$$\text{subject to } A\mathbf{x} \geq \mathbf{b}, \tag{2.6b}$$

$$\mathbf{x} \in \text{ conv } X. \tag{2.6c}$$

with the solution set X^*_{conv}. One can show (for a proof, see, e.g., [13]) that it holds that $q^* = z^*_{\text{conv}}$, i.e., the best dual bound equals the optimal objective value to (2.6).

To illustrate Lagrangian relaxation consider the following: For the small example problem (2.2), the Lagrange function is

$$\mathcal{L}(\mathbf{x}, u) = x_1 + 2x_2 + 2x_3 + (3 - 2x_1 - 2x_2 - 2x_3)u,$$

and the Lagrangian dual function is then defined by

$$q(u) = 3u + \min_{\mathbf{x} \in \{0,1\}} \Big((1 - 2u)x_1 + (2 - 2u)x_2 + (2 - 2u)x_3 \Big),$$

which implies that the Lagrangian subproblem is the problem

$$\min_{\mathbf{x} \in \{0,1\}} \Big((1 - 2u)x_1 + (2 - 2u)x_2 + (2 - 2u)x_3 \Big).$$

In this small example a feasible solution to the dual problem is any solution where $u \geq 0$. Remember that the optimal objective value of the problem (2.2) is $z^* = 3$. Weak duality states that a objective value of any dual solution is a lower bound to z^*. Let us check this for $u = 1$:

$$q(1) = 3 + \min_{\mathbf{x} \in \{0,1\}} \Big((1 - 2)x_1 + (2 - 2)x_2 + (2 - 2)x_3 \Big) = 3 + \min_{\mathbf{x} \in \{0,1\}} (-x_1)$$

The Lagrangian subproblem is then

$$\min_{\mathbf{x} \in \{0,1\}} -x_1$$

which clearly has $x_1 = 1$ in an optimal solution. Since neither x_2 or x_3 affect the objective value, they can be either 0 or 1. Consequently, the dual objective value in this example is

$$q(1) = 3 - 1 = 2,$$

which is ≤ 3. Hence, Theorem 1 (Weak duality) is fulfilled for $u = 1$.

A more extended and detailed description of Lagrange duality can be found in, e.g., [1, Ch.6].

2.2 Algorithm for the Lagrangian dual problem

Since the Lagrangian dual problem (2.5) is a convex optimization problem, a subgradient method may be applied to solve it. The method works as follows. Let $\mathbf{u}^0 \in \mathbb{R}^m_+$ and compute iterates $\mathbf{u}^{t+1}$ according to

$$\mathbf{u}^{t+1} = [\mathbf{u}^t + \alpha_t(\mathbf{b} - A\mathbf{x}^t)]_+, \qquad t = 0, 1, ..., \tag{2.7}$$

where $\mathbf{x}^t \in X(\mathbf{u}^t)$ is the Lagrangian subproblem solution in (2.4) at $\mathbf{u}^t$, $[\cdot]_+$ denotes the Euclidean projection onto the nonnegative orthant, and $\alpha_t > 0$ is the step length chosen in iteration t. Since $\mathbf{x}^t \in X(\mathbf{u}^t)$, this implies that the vector $\mathbf{b} - A\mathbf{x}^t$ is a subgradient to q at $\mathbf{u}^t$.

Theorem 2 (convergence of the subgradient method). *Assume that, when applied to the problem (2.5), the following conditions for the step length are fulfilled:*

$$\alpha_t > 0, \;\; t = 0, 1, \ldots, \;\; \lim_{t\to\infty} \sum_{s=0}^{t-1} \alpha_s = \infty, \text{ and } \lim_{t\to\infty} \sum_{s=0}^{t-1} \alpha_s^2 < \infty,$$

then the subgradient method will converge, i.e., $u^t \to u^*$ and $q(u^t) \to q^*$.

A proof can be found in [2].

The subgradient method is further described in, e.g., [1, Ch.6].

2.3 Generating a sequence of primal vectors

The dual sequence $\{\mathbf{u}^t\}$ from the Lagrangian subgradient method converges to a dual optimal solution, but since the corresponding primal sequence $\{\mathbf{x}^t\}$ can not be guaranteed to converge, ergodic iterates are introduced to obtain a solution to the primal problem.

2.3.1 Ergodic iterates

The ergodic iterates are constructed by using the Lagrangian subproblem solutions obtained from each iteration of the subgradient method. The ergodic iterates are convex combinations of all subproblem solutions found so far. When the Lagrangian dual problem (2.5) is solved by the subgradient optimization method (2.7), then at each iteration t an ergodic iterate is composed as

$$\overline{\mathbf{x}}^t = \sum_{s=0}^{t-1} \mu_s^t \mathbf{x}^s, \quad \sum_{s=0}^{t-1} \mu_s^t = 1, \quad \mu_s^t \geq 0, \quad s = 0, \ldots, t-1. \tag{2.8}$$

Here $\mathbf{x}^s$ is the solution to the Lagrangian subproblem in iteration s and μ_s^t are the *convexity weights* of the ergodic iterate. The ergodic sequence

converges to the optimal solution set of the convexified version (2.6), if the step lengths and convexity weights are chosen appropriately (see [12]). Let

$$\gamma_s^t = \mu_s^t/\alpha_s, \;\; s = 0, \ldots, t-1, \;\; t = 1, 2, \ldots \;\; \text{and} \tag{2.9a}$$

$$\Delta\gamma_{\max}^t = \max_{s \in \{0,\ldots,t-2\}} \{\gamma_{s+1}^t - \gamma_s^t\}, \;\; t = 1, 2, \ldots. \tag{2.9b}$$

Assumption 1 (relation between convexity weights and step lengths)
The step length α_t and the convexity weights μ_s^t are chosen such that the following conditions are satisfied:

$$\begin{aligned} \gamma_s^t &\geq \gamma_{s-1}^t, \;\; s = 1, \ldots, t-1, \;\; t = 2, 3, \ldots, \\ \Delta\gamma_{\max}^t &\to 0, \qquad \text{as } t \to \infty, \\ \gamma_0^t &\to 0, \qquad \text{as } t \to \infty, \;\; \text{and} \;\; \gamma_{t-1}^t \leq \Gamma \;\; \text{for some} \;\; \Gamma > 0, \;\; \forall t. \end{aligned}$$

For example, if each ergodic iterate is chosen such that it equals the average of all previous subproblem solutions and the step length is chosen according to the harmonic series, Assumption 1 is fulfilled, i.e., if $\mu_s^t = 1/t$, $s = 0, \ldots, t-1$, $t = 1, 2, \ldots$, and $\alpha_s = a/(b+cs)$, $t = 0, 1, \ldots$, where $a, b, c > 0$ then with (2.9a) it follows that $\gamma_s^t = (b+cs)/at$ for $s = 0, \ldots, t-1$ for all t. Hence, $\gamma_s^t - \gamma_{s-1}^t = c/at > 0$ for $s = 0, \ldots, t-1$ and $\Delta\gamma_{\max}^t = c/at \to 0$ as $t \to \infty$. Moreover, $\gamma_t^1 \to 0$ and $\gamma_t^t \to c/a$ as $t \to \infty$.

Theorem 3 (convergence of the ergodic iterates). *Assume that the subgradient method (2.7) operated with a suitable step length rule attains dual convergence, i.e.,* $\mathbf{u}^t \to \mathbf{u}^\infty \in \mathbb{R}_+$*, and let the sequence* $\overline{\mathbf{x}}^t$ *be generated as in (2.8). If the step length* α_t *and the convexity weights* μ_s^t *fulfill Assumption 1, then*

$$\mathbf{u}^\infty \in U^* \;\; and \;\; \overline{\mathbf{x}}^t \to X_{conv}^*$$

The proof of the theorem can be found in [12]. Conclusively, the theorem states that if choosing step length and convexity weights correctly the ergodic sequence converges to the optimal solution set of the convexified problem (2.6).

2.3.2 Choosing convexity weights

Gustavsson et al. [12] introduces a set of rules (the s^k-rules) for choosing the convexity weights defining the ergodic sequences.

For $k = 0$ the rule is called the $1/t$-rule, where all previous Lagrangian subproblem solutions are weighted equally. (This has been studied and analysed by Larsson and Liu [15].) When $k > 0$, later subproblem solutions get more weight than the previous ones. This might give a better result as the later subproblem solutions are expected to be closer to the optimal solution of the original problem.

Definition 1 . *Let $k > 0$. The s^k-rule creates the ergodic sequences with the following convexity weights:*

$$\mu_s^t = \frac{(s+1)^k}{\sum_{l=0}^{t-1}(l+1)^k}, \quad for \ \ s = 0, \ldots t-1, \ \ t \leq 1. \tag{2.10}$$

An illustration of the convexity weight μ_s^t with $k = 0$, 1, 4 and 10, and $t = 10$ can be found in Figure 2.1.

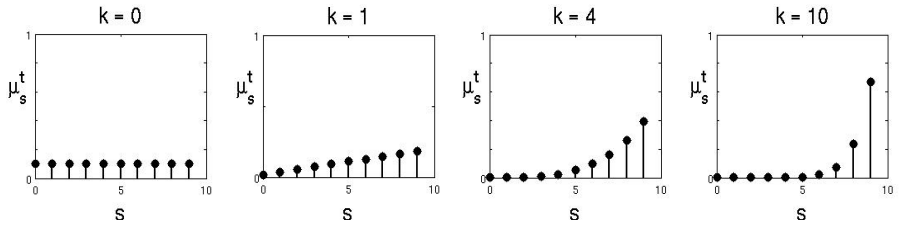

Figure 2.1: The convexity weight μ_s^t with $k = 0$, 1, 4 and 10, and with $t = 10$.

When constructing the ergodic iterates, only the previous ergodic iterate $\overline{\mathbf{x}}^{t-1}$ and the previous subproblem solution $\mathbf{x}^{t-1}$ is needed, as each ergodic iterate can be computed the following way

$$\overline{\mathbf{x}}^0 = \mathbf{x}^0, \quad \overline{\mathbf{x}}^t = \frac{\sum_{s=0}^{t-2}(s+1)^k}{\sum_{s=0}^{t-1}(s+1)^k}\overline{\mathbf{x}}^{t-1} + \frac{t^k}{\sum_{s=0}^{t-1}(s+1)^k}\mathbf{x}^{t-2}, \quad t = 1, 2, \ldots. \tag{2.11}$$

Hence, in each iteration the ergodic iterate can by updated easily.

2.4 Branch-and-bound algorithms

Branch-and-bound algorithms are methods used to solve integer programming problems. A branch-and-bound algorithm produces easier problems by relaxing the original problem and adding restrictions on variables to create subproblems. New subproblems correspond to new nodes in the branch-and-bound tree. The method find exact solutions to the problems and each feasible solution to a problem can be found in at least one subproblem. If the problem consists of n variables, one could solve at most 2^n subproblems, but by pruning nodes the amount is often reduced. For a more detailed explanation, see, e.g., Lodi [16] and Lundgren, Rönnqvist and Värbrand [17, Ch.15].

Branch-and-bound algorithms are composed of *relaxation*, *branching*, *pruning* and *searching*. The relaxation utilized is often LP-relaxation or Lagrangian relaxation. The solution of the relaxed problem is an optimistic estimate to the original problem.

Assuming now a minimization problem, the upper bound is a feasible solution to the original problem and the lower bound is given by the solution to the relaxed problem, which can be infeasible in the original problem.

The branching is done on the solution obtained from the relaxed problem. By restricting one or several variables possessing non-integer values in the solution of the relaxed problem, subproblems are created. For example, a relaxed binary variable is set to one in the first child node and to zero in the other.

In each node, the upper bound is compared with the global upper bound and the global upper bound is updated whenever there is a better upper bound. Furthermore, depending on the obtained solution from a subproblem a node is pruned or not. Nodes are pruned if

- there exists no feasible solution to the subproblem,
- the lower bound is higher than or equal to the global upper bound,
- the global lower bound equals the upper bound, or
- the solution is integer.

If the subproblem, in a node, has no feasible solution, then there is no feasible solution for the primal problem in this branch of the tree, and the branch can therefore be pruned. If the subproblem solution obtained in a node is worse or only as good as the best feasible solution found so far, the branching is stopped in that node of the tree, as the subproblem solution value can not be improved further in that branch.

The search through the branch-and-bound tree for the solutions, is done according to certain strategies. There are different ones for the branch-and-bound method. *Depth-first* and *breadth-first* are two of them. The depth-first strategy finds a feasible solution quickly, it searches through one branch at the time and goes on to the next branching level immediately, see Figure 2.2. The breadth-first strategy searches through all nodes in the same level first before going to the next level of branching as illustrated in Figure 2.3.

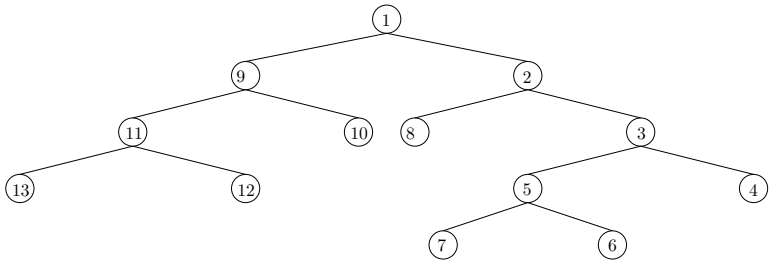

Figure 2.2: Depth-first branch-and-bound tree, where the node numbers illustrate the order in which the nodes are investigated.

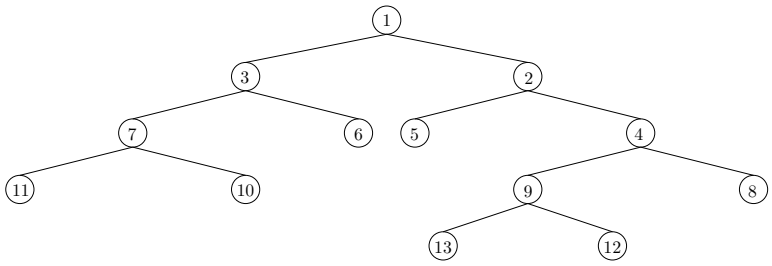

Figure 2.3: Breath-first branch-and-bound tree, where the node numbers illustrate the order in which the nodes are investigated

The variable to branch on can as well be chosen differently, i.e., one can choose to branch on the variables close to 0 or 1, or the ones close to 0.5.

Let's continue with the example in (2.2). The branch-and-bound method applied to that problem, with LP-relaxation and depth-first search strategy, is illustrated in Figure 2.4 where z_i is the objective function value of the relaxed problem in node i and $\mathbf{x}$ is the solution vector.

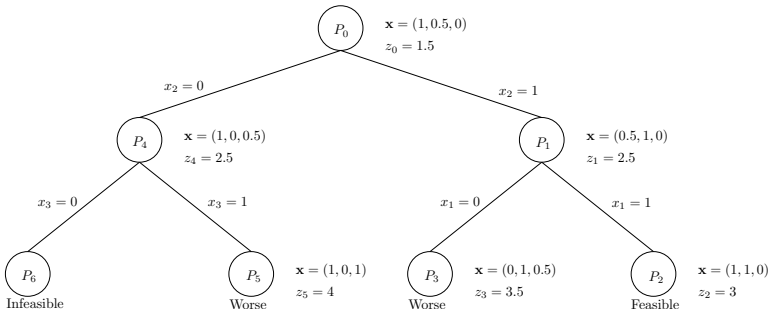

Figure 2.4: Branch-and-bound tree for example 1. P_0 is the root node which is solved by LP-relaxation and the solution obtained is $\mathbf{x} = (1, 0.5, 0)$. Then P_1 is solved where $x_2 = 0$ and so on.

First the LP-relaxation to the original problem is solved in the root node P_0, the objective function value z_0 obtained is a lower bound. Next the first branching is done on x_2, as this is the only fractional variable obtained from solving the LP-relaxed problem. x_2 is set to be 1 in one branch and 0 in the other. Then an LP relaxed problem is solved again now in the child-node P_1 where x_2 is fixed to be 1. The objective function value obtained in this node is z_1. In the solution vector of this node x_1 is the only fractional value, so this is the new variable to branch on by setting x_1 to 1 and 0. The branching then goes on and on until all variables are branched or the obtained solution vector $\mathbf{x}$ in a node contains no fractional values.

3 Evaluated algorithms

In this section the algorithms that are implemented and tested are presented. The first algorithm is the Lagrangian heuristic that uses the ergodic iterates to obtain upper bounds. The second algorithm is a branch-and-bound method where the Lagrangian heuristic is included and the branching decisions are based on the ergodic iterates.

3.1 Lagrangian heuristic

A Lagrangian heuristic is a method utilized in order to achieve a feasible solution to the primal problem by using the Lagrangian subproblem solutions generated in the iterations of the subgradient method.

It is possible to just take the Lagrangian subproblem solution from the latest iteration and construct, by making adjustments, a primal feasible solution. Unfortunately, there is no guarantee that the Lagrangian subproblem solution is close to the solution set of the primal problem. Thus, great adjustments might be required. The greater adjustments needed, the more uncertain is it that the recovered primal feasible solution is a good solution, i.e., close to optimum. The sequence of Lagrangian subproblem solutions, $\{\mathbf{x}^t\}$, is expected to get closer to the optimal solution, but does not converge. Consequently, how great adjustments that is needed is unknown.

The sequence of ergodic iterates, $\{\overline{\mathbf{x}}^t\}$, converges to the optimal solution set (X^*_{conv}) of the convexified version of the primal problem (2.6). This solution set is expected to be fairly close to the solution set of the primal problem. Thus, if the ergodic iterates are used to construct a primal feasible solution instead of the Lagrangian subproblem solutions, only small adjustments are needed. This implies that a Lagrangian heuristic that makes use of ergodic iterates may be preferable.

A Lagrangian heuristic based on the subgradient method can be described as below. This algorithm is also described and used by Gustavsson et al. [13].

Algorithm 1: Lagrangian heuristic

1. Choose a suitable step length and convexity weight rule. Decide on the maximum number of iterations, $\tau > 0$. Let $t := 0$ and choose $\mathbf{u}^0 \in \mathbb{R}^m_+$.

2. Solve the Lagrangian subproblem (2.4) at $\mathbf{u}^t$ and acquire the solution $\mathbf{x}^t \in X(\mathbf{u}^t)$. Calculate the dual function value $q(\mathbf{u}^t)$ [defined in (2.3)], which is the lower bound in iteration t. If possible, update the best lower bound found so far.

3. Update the ergodic iterate $\overline{\mathbf{x}}^t$ according to (2.11) and construct a feasible solution to the primal problem by making adjustments to $\overline{\mathbf{x}}^t$. Calculate the objective function value, which is the upper bound in iteration t. If possible, update the best upper bound found so far and the corresponding solution vector.

4. Terminate if $t = \tau$ or the difference between the upper and lower bound is within some tolerance.

5. Compute $\mathbf{u}^{t+1}$ according to (2.7), let $t := t + 1$ and repeat from 2.

3.2 Branch-and-bound with Lagrangian heuristic

The aim of this project is to incorporate the ideas of ergodic sequences in a branch-and-bound method as described in [13].

At each node in the branch-and-bound tree, the subgradient method (2.7) is applied to the problem (2.5) which yields a lower bound on the optimal value of (2.1) from an approximation of the optimal value of (2.5). The upper bound is obtained by applying the Lagrangian heuristic with the ergodic sequences which gives a feasible solution to the primal problem (2.1). The branching is performed with the help of the ergodic sequence $\overline{\mathbf{x}}^t$ obtained in each node from the subgradient method, where variable j is chosen such that $\overline{x}^t_j$ is either close to 0 or 1, or such that $\overline{x}^t_j$ is close to 0.5. The use of a Lagrangian heuristic with a dual subgradient method for a lower bound in a branch-and-bound tree has been studied by Görtz and Klose in [11].

The optimization problem in each node of the branch-and-bound tree is then the problem (2.6) with the additional constraints

$$x_j = \begin{cases} 1, & j \in \mathcal{I}^1_n, \\ 0, & j \in \mathcal{I}^0_n, \end{cases} \qquad j = 1, \dots n_x. \tag{3.1}$$

where the index sets $\mathcal{I}^1_n$ and $\mathcal{I}^0_n$ denotes the variables that have been fixed to 1 and 0 during the method.

The following algorithm creates a branch-and-bound tree, where the Lagrangian heuristic is applied in each node.

Algorithm 2: Branch-and-bound with Lagrangian heuristic

1. Initialize the Lagrangian multipliers $\mathbf{u}^0 \in \mathbb{R}^m_+$ and the iteration variable $\tau > 0$ for Algorithm 1.
2. For the optimization problem (2.6), (3.1): Let $t := 0$ and apply τ iterations of Algorithm 1, the Lagrangian heuristic, which gives a lower and an upper bound.
3. Check if pruning is needed. Prune, if possible.
4. Update the upper bound, if possible.
5. Choose a variable to branch on, based on the ergodic iterate $\overline{\mathbf{x}}^t$.
6. Branch on the chosen variable and repeat from 2.

The method terminates when all interesting nodes have been generated and investigated. The Lagrangian multipliers $\mathbf{u}^0$ in step 1 are often chosen as the final point ($\mathbf{u}^t$) obtained from the subgradient method of the parent node.

4 Problem types for algorithm evaluation

In this section the different problem types that are used for evaluating the algorithms are presented. The problem types are the set covering problem, the uncapacitated facility location problem, and the capacitated facility location problem. All of these problem types are well-studied mixed binary linear programs.

4.1 Set covering problem

The *set covering problem* (SCP) is the problem to minimize the total cost of chosen sets, such that all elements are included at least once.

The elements and the sets correspond to the rows and the columns, respectively, of a matrix A. Let $A = (a_{ij})$ be a $\mathcal{M} \times \mathcal{N}$ matrix with zeros and ones. Let $\mathbf{c} \in \mathbb{R}^{\mathcal{N}}$ be the cost vector. The value $c_j > 0$ is the cost of column $j \in \mathcal{N}$. If $a_{ij} = 1$, column $j \in \mathcal{N}$ covers row $i \in \mathcal{M}$. This problem has been studied by, for example, Caprara, Fischetti and Toth [5, 6]. The binary linear programming model can be formulated as the problem to

$$\text{minimize} \quad \sum_{j \in \mathcal{N}} c_j x_j, \tag{4.1a}$$

$$\text{subject to} \quad \sum_{j \in \mathcal{N}} a_{ij} x_j \geq 1, \qquad i \in \mathcal{M}, \tag{4.1b}$$

$$x_j \in \{0, 1\}, \qquad j \in \mathcal{N}. \tag{4.1c}$$

The objective function is to minimize the cost. The constraints (4.1b) ensure that each row $i \in \mathcal{M}$ of the matrix A is covered by at least one column.

Lagrangian relaxation of the SCP problem

The constraints (4.1b) are the ones that are Lagrangian relaxed and u_i, $i \in \mathcal{M}$, are the dual variables. The Lagrangian dual function $q : \mathbb{R}^{|\mathcal{M}|} \mapsto \mathbb{R}$ is

then defined as

$$q(u) := \sum_{i \in \mathcal{M}} u_i + \min \sum_{j \in \mathcal{N}} \bar{c}_j x_j, \tag{4.2a}$$

$$\text{s.t. } x_j \in \{0, 1\}, \qquad i \in \mathcal{N}, \tag{4.2b}$$

where $\bar{c}_j = c_j - \sum_{i \in \mathcal{M}} a_{ij} u_i$, $j \in \mathcal{N}$.

The subproblem in (4.2) can be separated into independent subproblems, one for each $j \in \mathcal{N}$. These subproblems can then be solved analytically in the following way. If $\bar{c}_j \leq 0$ then $x_j := 1$, otherwise $x_j := 0$, for $j \in \mathcal{N}$.

4.2 Uncapacitated facility location problem

The *uncapacitated facility location problem* (UFLP) deals with facility locations and clients. More precisely, the problem is to choose a set of facilities and from those serve all clients at a minimum cost, i.e., the objective is to minimize the sum of the fixed setup costs and the costs for serving the clients. The problem has been studied by, e.g., Barahona et al. in [3].

Let $\mathcal{F}$ be the set of facility locations and $\mathcal{D}$ the set of all clients. Then the UFLP can be formulated as the problem to

$$\text{minimize} \quad \sum_{i \in \mathcal{F}} f_i y_i + \sum_{j \in \mathcal{D}} \sum_{i \in \mathcal{F}} c_{ij} x_{ij}, \tag{4.3a}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{F}} x_{ij} \geq 1, \qquad j \in \mathcal{D}, \tag{4.3b}$$

$$0 \leq x_{ij} \leq y_i, \qquad j \in \mathcal{D}, \ \ i \in \mathcal{F}, \tag{4.3c}$$

$$y_i \in \{0, 1\}, \qquad i \in \mathcal{F}, \tag{4.3d}$$

where f_i is the opening cost of facility i and c_{ij} is the cost for serving client j from facility i. The binary variables y_i represents if a facility at location $i \in \mathcal{F}$ is open or not. The variable x_{ij} is the fraction of the demand from facility location $i \in \mathcal{F}$ to client $j \in \mathcal{D}$. The constraints (4.3b) ensure that the demand of each client $j \in \mathcal{D}$ is fulfilled. The constraints (4.3c) allow only the demand of a client from a certain facility to be greater than zero if that facility is open.

Lagrangian relaxation of the UFLP problem

The constraints (4.3b) can be Lagrangian relaxed. Consequently, the Lagrangian subproblem contains $|\mathcal{F}|$ easily solvable optimization problems. When the constraints (4.3b) are Lagrangian relaxed and u_j for $j \in \mathcal{D}$ are the dual variables, the Lagrangian dual function $q : \mathbb{R}^{|\mathcal{D}|} \mapsto \mathbb{R}$ is the following:

$$q(u) := \sum_{j\in\mathcal{D}} u_j + \min \sum_{j\in\mathcal{D}}\sum_{i\in\mathcal{F}} \overline{c}_{ij}x_{ij} + \sum_{i\in\mathcal{F}} f_i y_i, \tag{4.4a}$$

$$\text{s.t.} \quad 0 \le x_{ij} \le y_i, \qquad j \in \mathcal{D},\ \ i \in \mathcal{F}, \tag{4.4b}$$

$$y_i \in \{0,1\}, \qquad i \in \mathcal{F}, \tag{4.4c}$$

where $\overline{c}_{ij} = c_{ij} - u_j$ for $i \in \mathcal{F}$, $j \in \mathcal{D}$. The problem (4.4) can then be separated into independent subproblems, one for each $i \in \mathcal{F}$:

$$\min \quad \sum_{j\in\mathcal{D}} \overline{c}_{ij}x_{ij} + f_i y_i, \tag{4.5a}$$

$$\text{s.t.} \quad 0 \le x_{ij} \le y_i, \qquad j \in \mathcal{D}, \tag{4.5b}$$

$$y_i \in \{0,1\}, \tag{4.5c}$$

These problems (4.5) can then be solved as follows: If $\overline{c}_{ij} > 0$, then $x_{ij} := 0$ for $j \in \mathcal{D}$. Define $\mu_i = \sum_{j:\overline{c}_{ij}\le 0} \overline{c}_{ij}$. If $f_i + \mu_i < 0$, then $y_i := 1$ and $x_{ij} := 1$ if $\overline{c}_{ij} \le 0$. If $f_i + \mu_i \ge 0$, then $y_i := 0$ and $x_{ij} := 0$ for all $j \in \mathcal{D}$. In this way, the subproblems can be efficiently solved.

4.3 Capacitated facility location problem

The *capacitated facility location problem* (CFLP) involves facility locations and clients. The problem is to choose a set of facilities and from those serve all clients at a minimum cost, i.e., the objective is to minimize the sum of the fixed setup costs and the costs for serving the clients. At the same time each facility has a certain capacity s_i and the clients have a demand d_j that needs to be fulfilled. This problem has been studied among others by Barahona, et al. [3] and Geoffrion and Bride [10]. Let $\mathcal{F}$ be the set of facility locations and $\mathcal{D}$ the set of all clients. Then the CFLP can be formulated as the problem to

$$\text{minimize} \quad \sum_{i\in\mathcal{F}} f_i y_i + \sum_{j\in\mathcal{D}}\sum_{i\in\mathcal{F}} d_j c_{ij} x_{ij}, \tag{4.6a}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{F}} x_{ij} \ge 1, \qquad j \in \mathcal{D}, \tag{4.6b}$$

$$\sum_{j\in\mathcal{D}} d_j x_{ij} \le s_i y_i, \qquad i \in \mathcal{F}, \tag{4.6c}$$

$$0 \le x_{ij} \le y_i, \qquad j \in \mathcal{D},\ \ i \in \mathcal{F}, \tag{4.6d}$$

$$y_i \in \{0,1\}, \qquad i \in \mathcal{F}. \tag{4.6e}$$

where f_i is the opening cost of facility i, c_{ij} is the cost for serving client j from facility i, d_j is the demand of client $j \in \mathcal{D}$, and s_i is the capacity of the facility at location $i \in \mathcal{F}$. The binary variable y_i represents if a facility at location i $\in \mathcal{F}$ is open or not. The variable x_{ij} is the fraction of the demand from facility location $i \in \mathcal{F}$ to client $j \in \mathcal{D}$. The constraints (4.6b) ensure that the demand of each client $j \in \mathcal{D}$ is fulfilled. The constraints (4.6c) prohibit the demand part from a certain facility to a client to exceed the capacity of the facility. The constraints (4.6d) allow only the demand of a client from a certain facility to be greater than zero if that facility is open.

Lagrangian relaxation of the CFLP problem

The constraints (4.6b) can be Lagrangian relaxed. Consequently, the Lagrangian subproblem contains $|\mathcal{F}|$ easily solvable optimization problems. When the constraints (4.6b) are Lagrangian relaxed and u_j for $j \in \mathcal{D}$ are the dual variables, the Lagrangian dual function $q : \mathbb{R}^{|\mathcal{D}|} \mapsto \mathbb{R}$ is the following:

$$q(u) := \sum_{j\in\mathcal{D}} u_j + \min \sum_{j\in\mathcal{D}}\sum_{i\in\mathcal{F}} \overline{c}_{ij}x_{ij} + \sum_{i\in\mathcal{F}} f_i y_i, \tag{4.7a}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{D}} d_j x_{ij} \le s_i y_i \qquad i \in \mathcal{F}, \tag{4.7b}$$

$$0 \le x_{ij} \le y_i, \qquad j \in \mathcal{D},\ i \in \mathcal{F}, \tag{4.7c}$$

$$y_i \in \{0,1\}, \qquad i \in \mathcal{F}, \tag{4.7d}$$

where $\overline{c}_{ij} = d_j c_{ij} - u_j$ for $i \in \mathcal{F}$, $j \in \mathcal{D}$. The problem (4.7) can the be separated into independent subproblems, one for each $i \in \mathcal{F}$:

$$\min \quad \sum_{j\in\mathcal{D}} \overline{c}_{ij}x_{ij} + f_i y_i, \tag{4.8a}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{D}} d_j x_j \le sy, \tag{4.8b}$$

$$0 \le x_{ij} \le y_i, \qquad j \in \mathcal{D}, \tag{4.8c}$$

$$y_i \in \{0,1\}, \tag{4.8d}$$

These problems (4.8) can then be solved as follows: First if $\overline{c}_j > 0$ the x_j is set to 0. Then one orders

$$\frac{\overline{c}_1}{d_1} \le \frac{\overline{c}_2}{d_2} \le \frac{\overline{c}_3}{d_3} \cdots \le \frac{\overline{c}_n}{d_n}.$$

Let $b(k) = \sum_{j=1}^{j=k} d_j$, where k is the largest index such that $\sum_{j=1}^{j=k} d_j \le s$, and let $r = (s - b(k))/d_{k+1}$. If $f + \sum_{j=1}^{j=k} \overline{c}_j + \overline{c}_{k+1} r \ge 0$, then set $y = 0$ and $x_j = 0$ for all j, otherwise set $y = 1$ and $x_j = 1$ for $1 \le j \le k$, and $x_{k+1} = r$, if x_j is not already set to 0.

5 Numerical results

The Lagrangian heuristic, Algorithm 1 in 3.1, and the Branch-and-bound with Lagrangian heuristic, Algorithm 2 in 3.2, are implemented in MATLAB. In Algorithm 1 the Lagrangian relaxation is implemented as described in Section 4 for each problem type. Algorithm 2 is a depth-first branch-and-bound algorithm which works recursive. The global upper bound is in each node compared with the local upper bound and updated if possible. A global lower bound is not taken care of. In step 2, a slightly modified version of Algorithm 1 is performed: step 1 is disregarded and instead of constructing a primal feasible solution in each iteration, this is merely done after the last iteration.

This section contains numerical results from using the algorithms to solve test instances of UFLPs, SCPs and CFLPs. Algorithm 1 is utilized in case of UFLPs and Algorithm 2 is used for the SCPs and CFLPs.

5.1 UFLP

Algorithm 1 is applied to the UFLP defined in (4.3). The test instances of the problem are from Beasley's OR library [1].

In the subgradient method (2.7) the step lengths are chosen to be $\alpha_t = \frac{10^5}{1+t}$, and the dual variables are initialized to $u_j^0 = 0$ for $j \in \mathcal{D}$. In each iteration t, the subproblem (4.5), for each $i \in \mathcal{F}$, is solved for $\mathbf{u} = \mathbf{u}^t$, and an ergodic iterate $\overline{\mathbf{y}}^t$ is computed according to (2.11). Randomized rounding (see [3]) is used for the construction of feasible solutions in step 3 in each iteration. The procedure for this is as follows:

Open facility $i \in \mathcal{F}$ with probability $\overline{y}_i^t$. If none of the facilities are opened, just open the one with the highest probability. Assign all clients $j \in \mathcal{D}$ to their closest open facility. Generate 10 solutions by 10 tries of randomized rounding and use the best one, i.e. a feasible solution with the lowest objective value, as an upper bound.

The number of iterations to obtain an optimal solution is investigated for different convexity weight rules; the s^k-rules (2.10), which are different

[1] Available at: http://people.brunel.ac.uk/ mastjjb/jeb/orlib/uncapinfo.html (accessed 2015-03-13)

in regard to their k-value. The different convexity weight rules affect how the ergodic iterate $\overline{\mathbf{y}}^t$ is constructed according to (2.11). The algorithm is tested for the s^k-rules with $k = 0$, $k = 1$, $k = 4$, $k = 10$, $k = 20$ and $k = \infty$, where $k = 0$ generates the average of all Lagrangian subproblem solutions and $k = \infty$ represents the traditional Lagrangian heuristic that only utilizes the last Lagrangian subproblem solution.

In Table 5.1 and Figure 5.1 the results from running Algorithm 1 on 12 test instances of UFLP for the six different convexity weight rules are illustrated. The result are averages over 100 runs. In Table 5.1 the entries represent the number of iterations until an optimal solution was found. In Figure 5.1 the graphs show the performance profiles of Algorithm 1 when using the different convexity weight rules. The performance is the percentage of the problem instances solved within τ times the number of iterations needed by the method that used the least amount of iterations.

ID	Size	$k = 0$	$k = 1$	$k = 4$	$k = 10$	$k = 20$	$k = \infty$
cap71	16×50	51.9	40.3	**34.2**	34.5	35.3	74.0
cap72	16×50	87.3	63.6	56.1	55.7	**53.8**	92.0
cap73	16×50	104.6	82.0	69.2	58.8	**57.6**	144.0
cap74	16×50	62.9	50.9	38.9	33.1	**24.0**	105.0
cap101	25×50	152.8	110.0	85.6	77.1	**74.4**	598.0
cap102	25×50	179.9	137.8	109.4	103.1	**99.1**	121.0
cap103	25×50	158.7	111.9	86.4	**75.8**	78.3	337.0
cap104	25×50	98.9	67.7	52.2	45.9	**43.2**	61.0
cap131	50×50	331.4	206.7	150.7	136.8	**133.5**	470.0
cap132	50×50	300.0	173.4	130.0	116.3	**112.3**	466.0
cap133	50×50	376.8	231.1	187.0	168.9	**164.8**	1193.0
cap134	50×50	165.5	92.5	63.7	56.3	**52.4**	91.0

Table 5.1: The average number of iterations of Algorithm 1 over 100 runs for finding an optimal solution to each of the 12 test instances using different convexity weight rules. The best result, i.e., the rule that required the least number of iterations to find an optimal solution, for each test instance is marked with a bold entry.

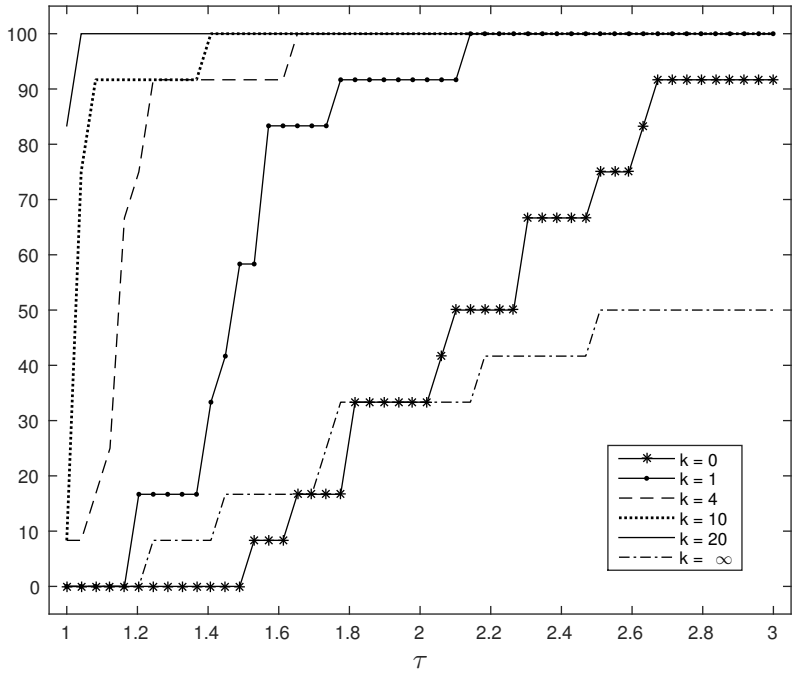

Figure 5.1: Performance profiles for Algorithm 1 applied on the 12 test instances from the OR-library. The graphs correspond to the six convexity weight rules and shows the percentage of the problem instances solved within τ times the number of iterations needed by the method that used the least amount of iterations, for $\tau \in [1.0, 3.0]$.

5.2 SCP

The SCPs, defined in (4.1), are solved by employing Algorithm 2. The step lengths are set to $\alpha_t = \frac{10}{1+t}$ for each iteration t and the initial values of the dual variables are set to $u_i^0 = \min_{j \in N: a_{ij}=1}\{c_j/|I_j|\}$, where $I_j = \{i : a_{ij} = 1\}$, $i \in \mathcal{M}$. In each iteration of Algorithm 1 the problem (4.2) is solved and the ergodic iterates $\overline{\mathbf{x}}^t$ are constructed according to (2.11) with the s^k-rule (2.10), where the k-value set to 4. The Lagrangian subproblem solution obtained in each iteration is investigated for feasibility in the primal problem and saved if feasible. Lastly, a feasible solution is constructed by randomized rounding with the ergodic iterates and compared with the saved Lagrangian subproblem solution (if there is one). The best one is then used as an upper bound.

The randomized rounding is performed by choosing set $j \in \mathcal{N}$ with probability $\overline{x}_j^t$. 10 solutions are generated by randomized rounding and the best feasible one is then used as an upper bound. Then the ergodic iterate $\overline{\mathbf{x}}^t$ is used to choose the next variable to branch on. The branching is done on the variable closest to 0.5.

For comparison, a branch-and-bound method with LP-relaxation is implemented and tested on the same problems as Algorithm 2. In each node of the branch-and-bound tree the LP-relaxation is solved by MATLABs own function `linprog` and a feasible solution is constructed by randomized rounding.

In Table 5.2, test instances from Beasley's OR library [2] are solved either with Algorithm 2 or a branch-and-bound method with LP-relaxation. The number of nodes in the branch-and-bound tree is listed for each problem instance. The maximum number of iterations of Algorithm 1 is set individually for each problem instance.

In Table 5.3 the the number of nodes of the branch-and-bound tree, when running Algorithm 2 for different sizes of the SCPs, is presented. The problem instances are created by setting the cost c_j to 1 for all columns and the elements in the matrix A are randomly set to 0 or 1. In each node either a LP-relaxation and randomized rounding or Algorithm 1 are applied to solve the problem. In case of Algorithm 1, the number of iterations is 10, 50, 100 and 1000 in all nodes except in the root node, in which 10 times as many iterations are done. Step length, dual variables and k-value are initialized as above.

In Figure 5.2 the performance profiles for Algorithm 2 and a branch-and-bound with LP-relaxation on 35 test instances are illustrated. For Algorithm 2, four different maximum number of iterations of Algorithm 1 are tested. The graphs shows the percentage of the problems solved within τ

[2] Available at: http://people.brunel.ac.uk/ mastjjb/jeb/orlib/scpinfo.html (accessed 2015-03-13)

times the number of nodes needed by the method the used the least amount of nodes, for $\tau \in [1.0, 3.0]$.

			Algorithm 2	
ID	Size	B&B-LP	Iterations	Nodes
4.1	200×1000	2.44	2000/200	**1.96**
4.2	200×1000	1	1000/100	1
4.3	200×1000	1	500/200	1
4.4	200×1000	5	5000/500	5
4.5	200×1000	1	500/100	1
4.6	200×1000	**8**	5000/500	13
4.7	200×1000	1	2000/500	1
4.8	200×1000	15	2000/500	**13**
4.9	200×1000	**7**	5000/1000	9
4.10	200×1000	3.4	2000/200	**3**
5.1	200×2000	72	2000/500	**25**
5.2	200×2000	49	5000/500	**5**
5.3	200×2000	2.28	2000/500	**1**
5.4	200×2000	17	5000/1000	**15**
5.5	200×2000	3	2000/200	3
5.6	200×2000	1	2000/200	1
5.7	200×2000	16.52	4000/400	**13**
5.8	200×2000	23	5000/200	**15**
5.9	200×2000	**2.44**	5000/1000	3
5.10	200×2000	1.24	1000/100	**1**
6.1	200×2000	211	1000/500	**151**
6.2	200×2000	361	1000/500	**197**
6.3	200×2000	83	1000/500	**33**
6.4	200×2000	38.8	1000/500	**19**
6.5	200×2000	83.2	5000/1000	**73**

Table 5.2: The amount of branch-and-bound nodes for the SCPs, over 25 runs, solved with Algorithm 2 and a branch-and-bound with LP-relaxation (B&B-LP), respectively. Algorithm 2 is run for different maximum number of iterations of Algorithm 1 for each test instance. The maximum number of iterations is stated for the root node and the remaining nodes, respectively, e.g., 2000/200 stands for 2000 iterations in the root node and 200 iterations in the remaining nodes of the tree.

		Algorithm 2			
Size	B&B-LP	100/10	500/50	1000/100	10000/1000
10×10	1.02	**1**	**1**	**1**	**1**
10×10	**1**	27.32	**1**	**1**	**1**
10×10	2.32	6.56	**1**	**1**	**1**
10×10	1.58	2.8	**1**	**1**	**1**
10×10	**2.42**	4.76	2.72	2.74	2.66
10×15	2.1	30.02	2.2	1.98	**1.96**
10×15	1.54	77	**1**	**1**	**1**
10×15	1.58	147	**1**	**1**	**1**
10×15	1.66	28.12	**1**	**1**	**1**
10×15	2.28	**1**	**1**	**1**	**1**
10×20	3.04	305	7	5	**3**
10×20	1.66	105.4	**1**	**1**	**1**
10×20	2	35.54	**1**	**1**	**1**
10×20	4.02	**1**	**1**	**1**	**1**
10×20	1.62	199	**1**	**1**	**1**
10×25	2.66	739.52	**1**	**1**	**1**
10×25	1.22	264.26	**1**	**1**	**1**
10×25	2.04	288.84	2.08	**2.02**	2.06
10×25	1.34	**1**	**1**	**1**	**1**
10×25	1.7	1105	**1**	**1**	**1**
20×20	2.52	276.1	5.96	2.52	**1**
20×20	4.9	770.62	12.68	**2.86**	2.9
20×20	3.22	1228.4	**1**	**1**	**1**
20×20	3.72	534.76	3.48	6.18	**3.40**
20×20	3.54	1256.3	41	11	**3**
20×30	3.56	2691.1	5.42	5.18	**3.32**
20×30	3.48	12165	191	27	**3**
20×30	4.86	3213.1	5	**4.62**	4.74
20×30	9.46	3502.4	199.94	**1**	**1**
20×30	2.70	7820.3	**1**	**1**	**1**
20×40	2.98	7762.8	**1**	**1**	**1**
20×40	2.36	9883.7	423	**1**	**1**
20×40	2.02	6625.2	**1**	**1**	**1**
20×40	2.78	8040.6	305	**1**	**1**
20×40	8.9	11767	242.16	12.46	**6.7**

Table 5.3: The average number of branch-and-bound nodes over 100 runs for different sizes of the SCPs, solved with Algorithm 2 and a branch-and-bound with LP-relaxation (B&B-LP). Algorithm 2 is run for different maximum numbers of iterations of Algorithm 1 for each test instance. The maximum number of iterations is stated for the root node and the remaining nodes, respectively, e.g., 100/10 stands for 100 iterations in the root node and 10 iterations in the remaining nodes of the tree.

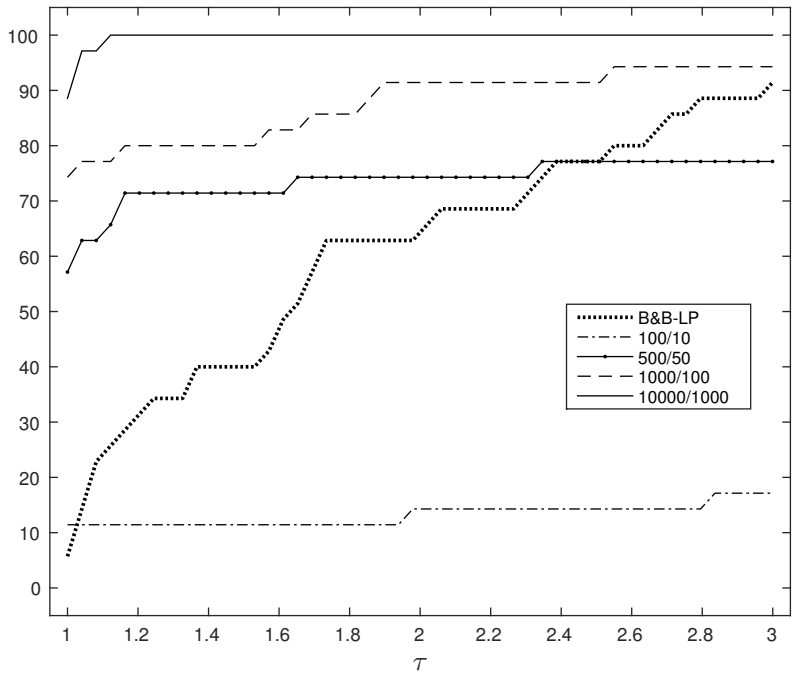

Figure 5.2: Performance profiles for Algorithm 2 and a branch-and-bound with LP-relaxation (B&B-LP) on 35 test instances. Algorithm 2 is run for four different maximum numbers of iterations of Algorithm 1. The name, e.g., 100/10, is the maximum number of iterations in the root node and the remaining nodes, respectively. The graphs shows the percentage of the problems solved within τ times the number of nodes needed by the method the used the least amount of nodes, for $\tau \in [1.0, 3.0]$.

5.3 CFLP

Running Algorithm 2 on the CFLPs as defined in (4.6), gives the result presented in Table 5.4 and Figure 5.3.

The CFLPs are created according to Klose and Görtz [11]. Customer and facility locations are generated as uniformly distributed points in a unit square $[a, b)$. The demands d_j are generated in the interval $[5, 35)$ and the capacities s_i in the interval $[10, 160)$. The transportation cost c_{ij} are obtained as the Euclidean distance multiplied by $10d_j$. The fixed setup cost for the facilities is $f_i = [0, 90) + [100, 110)\sqrt{s_i}$. The capacities are then rescaled such that $\sum_i s_i = 5\sum_j d_j$.

In each node Algorithm 1 is applied to the problem. The number of iterations is 500 in the root node and 100 in all other nodes. The step length is set to $\alpha_t = \frac{10^3}{1+t}$ for each iteration t. The initial values of the dual variables are set to $u_i^0 = \min_{j \in N: a_{ij}=1}\{c_j/|I_j|\}$ where $I_j = \{i : a_{ij} = 1\}, i \in \mathcal{M}$. The result is a comparison between different convexity weights (2.10). The k-value is set to $k = 0$, $k = 4$, $k = 20$, $k = \infty$, where $k = 0$ generate the average of all Lagrangian subproblem solutions and $k = \infty$ represent the traditional Lagrangian heuristic that only utilizes the last Lagrangian subproblem solution.

In each iteration of Algorithm 1 the problem (4.8) is solved and the ergodic iterates $\overline{\mathbf{y}}^t$ are constructed according to (2.11). The Lagrangian subproblem solution obtained in each iteration is investigated for feasibility in the primal problem and saved if feasible. Lastly, a feasible solution is constructed by randomized rounding with the ergodic iterates and compared with the saved Lagrangian subproblem solution (if there is one). The best one is then used as a upper bound.

The randomized rounding is done by opening facility $i \in \mathcal{F}$ with probability $\overline{y}_i^t$. Then a linear optimization problem is solved for all x_{ij}s with MATLABs function `linprog` and the obtained solution is checked for feasibility. If the solution obtained is feasible it is saved. This is done 10 times and the best solution is then used as a upper bound to the problem. Then the ergodic iterate $\overline{\mathbf{y}}^t$ is used to choose the next variable to branch on. The branching is done on the variable closest to 0.5.

In Figure 5.3 the performance profiles are illustrated for the 35 test instances created with different size, where the graphs show the percentage of the problem instances solved by each method depending on τ. The variable τ is the number describing how many times the number of nodes are needed to solve the problem instance with the method that used the least amount of nodes.

Size	k = 0	k = 4	k = 20	k = ∞
10×10	**11**	**11**	13	43
10×10	25	11	**5**	61
10×10	7	**5**	**5**	83
10×10	39	**25**	**25**	31
10×10	5	**3**	**3**	11
10×15	41	**7**	**7**	11
10×15	21	**7**	**7**	21
10×15	15	**13**	**13**	21
10×15	19	**11**	13	13
10×15	9	**5**	**5**	9
10×20	17	**13**	**13**	21
10×20	**21**	**21**	**21**	63
10×20	**1**	**1**	**1**	**1**
10×20	**7**	**7**	**7**	21
10×20	13	**9**	15	21
15×15	45	**37**	**37**	39
15×15	19	**3**	17	23
15×15	17	**9**	17	21
15×15	13	9	**5**	7
15×15	7	**5**	**5**	11
15×20	67	**43**	**43**	67
15×20	33	**29**	33	31
15×20	33	**31**	33	33
15×20	35	**33**	**33**	37
15×20	53	**39**	**39**	99
20×20	**3**	**3**	**3**	57
20×20	163	**69**	83	387
20×20	9	**3**	**3**	**3**
20×20	79	45	**43**	47
20×20	13	**7**	21	113
30×30	169	**73**	107	167
30×30	77	**71**	99	99
30×30	287	**163**	247	1823
30×30	147	**87**	151	127
30×30	67	**63**	69	**63**

Table 5.4: The average number of branch-and-bound nodes and the depth of the tree for different size of the CFLPs and different k-values, $k = 0, 4, 20$ and ∞. The number of iterations is 500 in the root node and 100 in all other nodes.

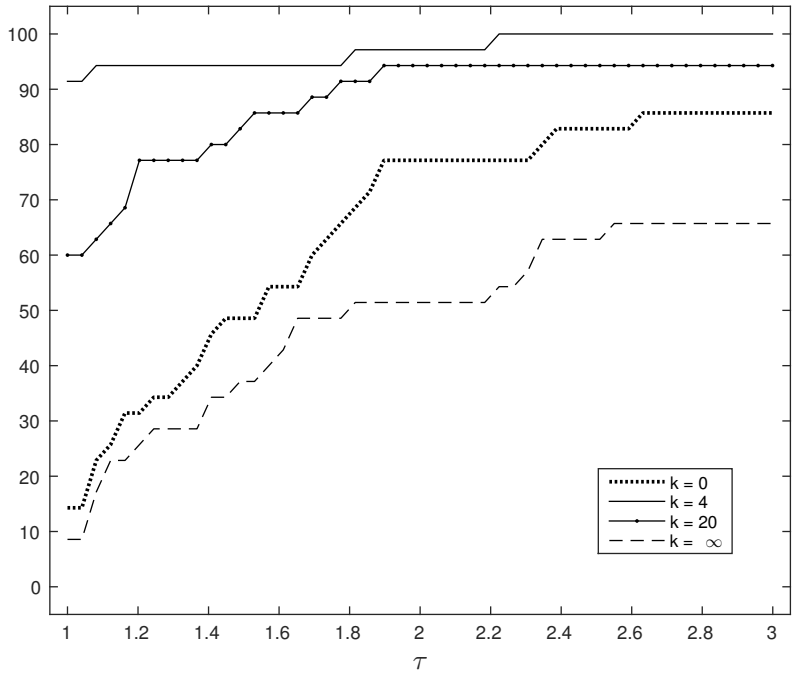

Figure 5.3: Performance profiles for Algorithm 2 on 35 test instances for each of the four convexity weight rules. The graphs shows the percentage of the problem instances solved within τ times the number of nodes needed by the method that used the least amount of nodes, for $\tau \in [1.0, 3.0]$.

6 Discussion and conclusions

In this section the results are discussed, conclusions are presented, and future work is proposed.

6.1 Discussion

The results from testing the algorithms are commented on for each problem type, namely UFLP, SCP and CFLP.

6.1.1 UFLP

In Table 5.1 the results from solving test instances of the UFLP with Algorithm 1 using different convexity weight rules are presented. For $k = 0$, the constructed ergodic iterate is an average of all Lagrangian subproblem solutions. When $k = \infty$ no ergodic iterate is created. This is the traditional Lagrangian heuristic where only the last Lagrangian subproblem solution is used.

One can conclude that k = 4, 10, and 20 performs much better than $k = 0$ and $k = \infty$. The entries with bold font represent the number of the least iterations to obtain an optimal solution in each problem instance, and most of them belong to the same convexity weight rule. Hence, it is clear that $k = 20$ is the best, but only slightly better than $k = 10$. The good performances of k = 4, 10, and 20 are also visualised in Figure 5.1 with the performance profiles of all the convexity weight rules. Similar results for the same test instances can be found in Gustavsson et al. [13].

Note that for $k = 20$, the last Lagrangian subproblem solution has a high weight in comparison with weights of the previous Lagrangian subproblem solutions and this yields a very good result. However, for $k = \infty$, where all the weight is on the last Lagrangian subproblem solution, the result varies but is always worse. This shows the importance of letting the previous Lagrangian subproblem solutions have an impact.

6.1.2 SCP

When solving test instances of the SCP with Algorithm 2, a convexity weight rule with $k = 4$ is used because it is one of the rules that yields good results. Another value for k could also be appropriate if it is not too small or too large.

In Table 5.2 the results from solving 25 test instances of two different sizes of SCPs with Algorithm 2 and branch-and-bound with LP-relaxation (B&B-LP) are presented.

In most cases, Algorithm 2 performs better than the B&B-LP. For 22 of the 25 test instances the number of nodes in the completed branch-and-bound tree produced by Algorithm 2 is less than or equal to the number of nodes generated by the B&B-LP.

One drawback with Algorithm 2 is that the maximum number of iterations performed in each node has a great affect on the performance of the algorithm. The best settings varies from case to case and, unfortunately, the amount of iterations does not merely depend on the problem size.

Table 5.3 and Figure 5.2 display the results from solving small sized SCPs with Algorithm 2 and B&B-LP. The number of test instances is 35 in total and there are five test instances of each size. For Algorithm 2, different maximum numbers of iterations in the nodes are tested. The maximum number of iterations are chosen to be 10, 50, 100 and 1000 in all nodes except in the root node, in which 10 times as many iterations are allowed to be performed. This is, e.g., denoted 100/10 for the method with, at most, 100 and 10 iterations in the root node and the other nodes, respectively.

Depending on the number of iterations, Algorithm 2 performs better than the B&B-LP. For the method 100/10 the number of nodes becomes very large, especially for the larger problems. Therefore, more iterations are needed. If five times as many iterations are applied, i.e., the method 500/50, it is already enough to be better than B&B-LP for the small problem instances. For all problem sizes, the methods 1000/100 and 10000/1000 performs the best. This can easily be seen in Table 5.3 and the performance profiles in Figure 5.2.

6.1.3 CFLP

In Table 5.4 and Figure 5.3 the results from solving test instances of the CFLP with Algorithm 2 using different convexity weight rules are presented. The number of iterations is 500 in the root node and 100 in all other nodes, which was considered appropriate according to the problem sizes in this experiment. For $k = 0$, the constructed ergodic iterate is an average of all Lagrangian subproblem solutions. When $k = \infty$ no ergodic iterate is created. This is the traditional Lagrangian heuristic in which the last Lagrangian subproblem solution is used.

Since the least number of nodes for each problem instance is marked with a bold entry in Table 5.4, it is obvious that the algorithm with the convexity weight rule using $k = 4$ performs the best. Furthermore, to get an overview of the performance of all four methods one can study the performance profiles in Figure 5.3. In this figure it is clear that both $k = 4$ and $k = 20$ are superior to the other two methods. The traditional Lagrangian heuristic, where $k = \infty$, performs doubtlessly worst.

6.2 Conclusions

For Algorithm 1 and 2, i.e. the Lagrangian heuristic and the branch-and-bound with Lagrangian heuristic, when solving SCPs, UFLPs and CFLPs, the choice of the convexity weight rule has a great impact on their performance. The traditional Lagrangian heuristic, where all the weight is put on the last Lagrangian subproblem solution ($k = \infty$), performs the worst. This shows the importance of including the previous Lagrangian subproblem solutions. If all Lagrangian subproblem solutions are equally weighted ($k = 0$) the performance is only slightly better. However, by using a convexity weight rule which puts more weight on later solutions the performance is significantly improved. Conclusively, the later solutions are more important than the earlier ones. Although, too much weight on the last ones does not always give the best result. For example, in the case of solving CFLPs, the algorithm performs better with $k = 4$ compared to when $k = 20$.

Algorithm 2, with an appropriate convexity weight rule, can be better than a branch-and-bound with LP-relaxation. Though, it depends on the number of iterations of Algorithm 1, which can be hard to choose. The better performance can be explained with the fact that $\overline{\mathbf{x}}^t \to X^*_{conv}$ (Theorem 3) in Algorithm 2, while $\mathbf{x}^t \to X^*_{LP}$ in branch-and-bound with LP-relaxation and the solution set $X^*_{conv} \subseteq X^*_{LP}$.

6.3 Future work

A further development of this thesis could be to implement Algorithm 2 in a more computational efficient programming language. This would give the possibility to test the algorithm on larger problem instances and its runtime performance. The branch-and-bound method itself could also be improved by implementing other features, such as different search strategies and to make use of a global lower bound. Another advanced version of a branch-and-bound method could be the combination of Algorithm 2 and the construction of a core problem as described in Gustavsson et al. [13], where one would fixate more the one variable at the same time in each branch-and-bound node. Furthermore, another expansion of this project would be to test the two algorithms on other MBLPs.

Bibliography

[1] ANDRÉASON, N., EVGRAFOV, A., AND PATRIKSSON, M. *An Introduction to Continous Optimization*. Studentlitteratur, Lund, Sweden, 2005.

[2] ANSTREICHER, K. M., AND WOLSEY, L. A. Two "well-known" properties of subgradient optimization. *Mathematical Programming 120*, 1 (2009), 213–220.

[3] BARAHONA, F., AND CHUDAK, F. A. Near-optimal solutions to large-scale facility location problems. *Discrete Optimization 2*, 1 (2005), 35–50.

[4] BORCHERS, B., AND MITCHELL, J. E. An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers & Operations Research 21*, 4 (1994), 359–367.

[5] CAPRARA, A., FISCHETTI, M., AND TOTH, P. A heuristic method for the set covering problem. *Operations Research 47*, 5 (1999), 730–743.

[6] CAPRARA, A., TOTH, P., AND FISCHETTI, M. Algorithms for the set covering problem. *Annals of Operations Research 98*, 1–4 (2000), 353–371.

[7] DAKIN, R. J. A tree-search algorithm for mixed integer programming problems. *The Computer Journal 8*, 3 (1965), 250–255.

[8] EVERETT III, H. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations research 11*, 3 (1963), 399–417.

[9] FISHER, M. L. The Lagrangian relaxation method for solving integer programming problems. *Management Science 27*, 1 (1981), 1–18.

[10] GEOFFRION, A., AND MCBRIDE, R. Lagrangean relaxation applied to capacitated facility location problems. *AIIE Transactions 10*, 1 (1978), 40–47.

[11] GÖRTZ, S., AND KLOSE, A. A simple but usually fast branch-and-bound algorithm for the capacitated facility location problem. *INFORMS Journal on Computing 24*, 4 (2012), 597–610.

[12] GUSTAVSSON, E., PATRIKSSON, M., AND STRÖMBERG, A.-B. Primal convergence from dual subgradient methods for convex optimization. *Mathematical Programming 150*, 2 (2015), 365–390.

[13] GUSTAVSSON, E., PATRIKSSON, M., AND STRÖMBERG, A.-B. Recovery of primal solutions from dual subgradient methods for mixed binary linear programming. Preprint. Chalmers University of technology and University of Gothenburg, (2015).

[14] LAND, A. H., AND DOIG, A. G. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society* (1960), 497–520.

[15] LARSSON, T., PATRIKSSON, M., AND STRÖMBERG, A.-B. Ergodic, primal convergence in dual subgradient schemes for convex programming. *Mathematical programming 86*, 2 (1999), 283–312.

[16] LODI, A. Mixed integer programming computation. In *50 Years of Integer Programming 1958–2008*. Springer, Berlin, Germany, 2010, pp. 619–645.

[17] LUNDGREN, J., RÖNNQVIST, M., AND VÄRBRAND, P. *Optimization*. Studentlitteratur, Lund, Sweden, 2010.

[18] ROCKAFELLAR, R. T. *Convex Analysis*. No. 28. Princeton University Press, Princeton, NJ, USA, 1970.

[19] WOLSEY, L. A. *Integer Programming*, vol. 42. John Wiley & Sons, New York, NY, USA, 1998.

[20] WOLSEY, L. A., AND NEMHAUSER, G. L. *Integer and Combinatorial optimization*. John Wiley & Sons, New York, NY, USA, 2014.

Printed and Bound at
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg
2015