
**Programa de Doctorado "Matemáticas"**

PhD Dissertation

# Mixed Integer Nonlinear Optimization. Applications to Competitive Location and Supervised Classification

***Author***
*Amaya Nogales Gómez*

***Supervisors***
Prof. Dr. *Rafael Blanquero Bravo*
Prof. Dr. *Emilio Carrizosa Priego*
Prof. Dr. *Dolores Romero Morales*

December 9, 2014

*A mis padres y a mi hermana*

# Agradecimientos

A mis directores de tesis, Rafael, Emilio y Dolores, por su apoyo incondicional. Gracias por haberme enseñado tanto y por todo vuestro esfuerzo.

To Andrea, for his great support and for making Bologna one of my favorite places to do research.

A Txema, por hacerme descubrir las matemáticas.

A mis compañeros y amigos de la L2, por crear un magnífico ambiente y amenizar las eternas tardes de trabajo.

A mis amigos, sois una pieza clave en mi vida, vuestro ánimo ha sido fundamental. A mis Maireneras (en el exilio o no), y a los viernes al sol. A mi familia Trianera. A los Viejos Lobos.

A mi familia de Pedrera, Barcelona y Chiclana. A mis padres y a mi hermana, por todo. A mi yaya, porque es la mejor del mundo entero, os lo prometo. A mis tíos y primos, porque saben hacerme sentir querida a mil kilómetros de distancia.

A Carmen Ana, por estar ahí todos los días, porque sabe escucharme, ayudarme y quererme.

# Abstract

This PhD dissertation focuses on the study of Mixed Integer Nonlinear Programming (MINLP) problems [34] for two important and current applications: competitive location on networks [59, 64] and Support Vector Machines (SVM) [56, 152, 153].

Location problems on a network in a competitive environment were first introduced in [82]. They have been deeply studied in operations research and applied in problems such as market area analysis [122], demand estimation [123], or location of retail centres [78].

The SVM has proved to be one of the state-of-the-art methods for Supervised Classification [2, 6, 85, 160]. Successful applications of the SVM are found, for instance, in health care [22, 46, 81], fraud detection [43], credit scoring [113] and cancellations forecasting [138].

In its general form, an MINLP problem can be represented as:

$$\min f(x_1, x_2)$$

s.t.

$$\begin{aligned} g_i(x_1, x_2) \leq 0 \qquad & \forall i = 1, \ldots, I \\ x_1 \in \mathbb{Z}^{n_1} & \\ x_2 \in \mathbb{R}^{n_2}, & \end{aligned}$$

where $n_1$ is the number of integer variables, $n_2$ is the number of continuous variables, $I$ is the number of constraints and $f, g_i$ are arbitrary functions such that $f, g_i : \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$. The general class of MINLP problems is composed by particular cases such as Mixed Integer Linear Programming (MILP) problems, when $f, g_i \ \forall i = 1, \ldots, I$ are linear functions, Mixed Integer Quadratic Programming (MIQP) problems, when $f$ is quadratic or Quadratically Constrained Quadratic Programming (QCQP) problems, when $f, g_i$ are quadratic functions.

There are two main lines of research to solve this kind of problems: to develop packages for general MINLP problems [32, 55] or to exploit the specific structure of the problem. In this PhD dissertation we focus on the latter.

Concerning the first application, we study the problem of locating one or several facilities on a competitive environment in order to maximize the market share. We study the single and $p$-facility Huff location model on a network and the single Huff origin-destination trip model [95]. Both models are formulated as MINLP problems and solved by a specialized branch and bound, where bounding rules are designed using DC (difference of convex) and Interval Analysis tools.

In relation to the second application, we present three different SVM-type classifiers, focused either on robustness or interpretability. In order to build the classifier, different approaches are proposed based on the solution of MINLP problems, or particular cases of it such as MILP, MIQP or QCQP problems, and globally optimized using a commercial branch and bound solver [55, 100, 101].

# Contents

# Part I

# Competitive Location Analysis on Networks

# Chapter 1

# Location Analysis on Networks and the Huff Problem

## 1.1 An introduction to location analysis

Facility location problems [64, 84] aim at finding the optimal location of one or more facilities inside a region satisfying the demand from users, usually represented as points of a certain space. Classical examples of location applications are found in the private and public sectors. For example, industry must locate warehouses, production plants and retail outlets, while governments need to determine the location of hospitals, ambulances or fire stations.

Location problems can be classified based on two main components: the location search space (continuous, discrete or on networks) and the objective that determines the location of the facility (attraction, repulsion or a combination of them).

Let us focus on the first component, the location search space. A number of different spaces have been employed in the literature. In the continuous case, users and facilities are located in the $d$-dimensional space $\mathbb{R}^d$, usually considering $d = 2$ [110, 129]. In the discrete case, the set of feasible locations for the facilities is a finite set [59, 118, 134], and in network location problems, users are usually identified with the nodes of the network, and facilities are located in any node or edge of the network [12, 59]. In this part of the dissertation, Part I, we focus on location problems on networks.

Concerning the second component, there are two main opposite objectives: attraction and repulsion. Users may either try to attract desirable facilities closer to them, or repel undesirable facilities from them. The class of attractive objectives are based on the assumption that the facility or facilities to be located are desirable. As a result, the users demanding a service obtain as larger benefit as closer the facility is located. Examples such as schools, shopping or leisure centers are considered as desirable facilities. Attractive location problems are often based on minimizing a function of the facility-user distance. Classic problems in this class are the minisum, minimax and covering problems [50, 54, 82, 87, 148, 157].

On the contrary, some facilities are considered undesirable when the lower the distance to the users, the more threat causes to them. Examples such as waste sites, nuclear reactors or water purification plants are considered obnoxious facilities, and therefore, the decision maker would try to locate the facility in order to maximize the distance from the users to it. When the location problem involves locating an

undesirable facility, the problem is considered a repulsive problem [49, 69, 68, 119].

In the case the new facility to be located has both desirable and undesirable effects, the problem is considered a semi-obnoxious or semi-repulsive problem [24, 41, 47, 116, 121, 137]. On one hand, the facilities are beneficial for a set of users, which want to have the facilities as close as possible, so that transportation cost minimization is the main objective. On the other hand, the installation of the new facilities has undesirable effects, such as the environmental impact, which makes users want to have the facilities as far as possible. When both transportation costs and environmental impact are present, the semi-obnoxious facilities should be located at points optimizing a certain compromise measure between these two conflicting criteria. The traditional approach in the literature for semi-obnoxious location problems has been the aggregation of both objectives into a single one [41, 47].

One branch of location theory deals with the location of retail and other commercial facilities which operate in a competitive environment [61, 62, 67, 83, 130]. These are a particular case of attractive problems, where the facilities compete for users and market share, with a profit maximization objective. A location problem is said to be competitive when it explicitly incorporates the fact that other facilities are already present in the market and that the new facility will have to compete with them for its market share. In this part of the dissertation, we focus on competitive location problems.

## 1.2 Location on networks

Many datasets in real systems from fields such as sociology, health care or computer science can be represented as a network. The Internet, the World Wide Web, highways and neural, protein, citation, airline or social networks are just a few examples [13, 31, 75].

Network optimization problems [21] are widely used in practice due to their methodological aspects and intuitive formulations. They arise naturally in the context of assignment, flow, transportation or location problems among others [3, 112]. For a comprehensive introduction to location problems on networks see [103, 146].

Formally, we denote as $N = (V, E)$ a network, where $V$ is a finite set whose elements are called vertices or nodes, and $E$ is a subset of $V \times V$, whose elements are called arcs or edges. In network optimization problems, the users are usually represented as the nodes of the network, and the communication between users takes place through the edges. The length of the edge $e \in E$ is given, and it is denoted by $l_e$. The distance between two nodes $v_i, v_j$ is calculated as the shortest path [103] from $v_i$ to $v_j$. For each $e \in E$, with end nodes $v_i, v_j$, we identify each $x \in [0, l_e]$ with the point in the edge $e$ at distance $x$ from $v_i$ and distance $l_e - x$ from $v_j$. This way, we obtain that, for any vertex $v_k \in V$, the distance $d(x, v_k)$ from $x$ to $v_k$ is, as a function of $x$, a concave piecewise linear function, given by:

$$d(x,\ v_k) = \min\{f_{ik}(x),\ f_{jk}(x)\}, \tag{1.1}$$

$$\text{with } f_{ik}(x) = x + d(v_i,\ v_k),\ \ f_{jk}(x) = (l_e - x) + d(v_j,\ v_k).$$

Facility location problems on networks deal with the problem of selecting one or more points of a network as facility locations in order to optimize some criterion. Users are taken to be at the nodes of the network as a rule, and the problem is then to locate the facilities in some optimal way over the network such that the demand from users is satisfied.

### 1.2.1 The Huff location problem on networks

Competitive location problems [67, 130] were originally introduced by Hotelling in [93], considering the location of two competing facilities on a linear market. In the seminal work of Hotelling, users patronize the facility closest to them. In contrast with this all·or·nothing assumption, it was introduced the Huff location problem [95], in which the probability that a user patronizes a facility is proportional to its attractiveness and inversely proportional to a power of the distance to it. The Huff location problem has been extensively studied in the field of continuous location [25, 63, 72, 95, 96] and successfully applied in the marketing field, in problems such as location of petrol stations, shopping centers or restaurants [78, 122, 124]. The problem was extended by [122] on networks with the shortest-path distance and has been also addressed in [18, 30]. In [127], a modified Huff problem is proposed, namely the Pareto-Huff problem, based on the assumption that a user will patronize a further facility only if that facility is more attractive. In [136], metaheuristics are proposed for the $p$-facility case.

In the Huff location problem on networks, the finite set $V$ of vertices of the network represents users, asking for a certain service. Each user $v \in V$ has a demand $\omega_v > 0$, that is patronized by different existing facilities, located at points $y_1, \ldots, y_r$ on the network. The demand captured by facility at $y_i$ from user $v$ is inversely proportional to a positive nondecreasing function of the distance $d(v, y_i)$, namely, $1/\varphi_v(d(v, y_i))$ is used as the utility or attraction function of $y_i$. Therefore, the demand captured by the facility at $y_i$ from the user at $v$ is given by

$$\omega_v \frac{1/\varphi_v(d(v, y_i))}{\sum_{j=1}^{r} 1/\varphi_v(d(v, y_j))}. \tag{1.2}$$

Here $\varphi_v$ is assumed to be nonnegative and nondecreasing. The usual choice for each $\varphi_v$ has the form $\varphi_v(d) = d^{\lambda_v}$. When $\lambda_v = 2$ for all $v \in V$, we have the so-called gravitational model.

Expression (1.2) must be carefully considered when $\varphi_v(0) = 0$. Indeed, in such a case, if some $y_j$ exists with $y_j = v \in V$, then the overall demand of $v$ will be captured by $y_j$. Hence, such $v$ will not be taken into account, and thus we assume in what follows without loss of generality that $y_j \notin V$, $j = 1, \ldots, r$.

A new firm is entering the market, by locating $p$ new facilities at some points $x_1, \ldots, x_p$ on the network. This perturbs how the market is shared, since the new

facilities will capture part of the demand from $v \in V$,

$$\omega_v \frac{\sum_{j=1}^{p} 1/\varphi_v(d(v,x_j))}{\sum_{j=1}^{p} 1/\varphi_v(d(v,x_j)) + \sum_{j=1}^{r} 1/\varphi_v(d(v,y_j))}. \tag{1.3}$$

Our goal is the maximization of the market share of the entering firm. Thus, the problem we need to solve can be formulated as

$$\max_{\substack{x_1\in[0,l_{e_1}],\ldots,x_p\in[0,l_{e_p}]\\ e_1,\ldots,e_p\in E}} \sum_{v\in V} \omega_v \frac{\sum_{j=1}^{p} 1/\varphi_v(d(v,x_j))}{\sum_{j=1}^{p} 1/\varphi_v(d(v,x_j)) + \sum_{j=1}^{r} 1/\varphi_v(d(v,y_j))}. \tag{1.4}$$

Let us denote the total attraction of the existing facilities for each $v \in V$ by the positive constant

$$\beta_v = \sum_{j=1}^{r} \frac{1}{\varphi_v(d(v,y_j))}. \tag{1.5}$$

It follows that Problem (1.4) can be rewritten as the following Mixed Integer Nonlinear Programming (MINLP) problem

$$\max_{\substack{x_1\in[0,l_{e_1}],\ldots,x_p\in[0,l_{e_p}]\\ e_1,\ldots,e_p\in E}} F(x_1,\ldots,x_p) \tag{1.6}$$

with $F$ defined as

$$F(x_1,\ldots,x_p) = \sum_{v\in V} \omega_v \frac{1}{1 + \dfrac{\beta_v}{\sum_{j=1}^{p} \dfrac{1}{\varphi_v(d(v,x_j))}}}. \tag{1.7}$$

In this part of the dissertation, we focus on the MINLP problem (1.6). This problem is formed by a combinatorial and a continuous part. First, we need to solve the combinatorial problem of choosing a set of $p$ edges to locate the facilities, and then solve a continuous location problem on the edges. Hence, (1.6) includes, as subproblems, a hard combinatorial problem and a continuous multimodal optimization problem, yielding a challenging MINLP.

### 1.2.2 The Huff origin-destination trip problem

Many location problems implicitly assume that users wishing to obtain a service are travelling to the facility to obtain the service [47, 65, 82, 157]. This assumption makes the problem unsuitable for many types of facilities that rely on capturing users travelling to another destination, such as cash machines or gas stations. In [20, 90], a problem was introduced assuming that users are travelling on pre-planned trips between some origin

and destination (for example, on the daily route to and from work) and may consume a service if they pass through one of the facilities. This is the so-called origin-destination (OD) trip problem.

In the OD trip problem, the facility attraction is a function of the length of the shortest path from the origin to the destination through the facility. In this problem on networks, we have the set $\{\{u,v\}:\ u,v\in V\}$, of origin-destination pairs. Consumers in their way from origin $u$ to destination $v$ will stop at one facility, and the trip from $u$ to $v$ will imply a demand $\omega_{uv}>0$.

In this part of the dissertation, we particularize the OD trip problem to the Huff location problem on networks. In the Huff OD trip problem, inspired by [19, 20], users travelling from origin $u$ to destination $v$ will stop at one facility; they choose among the $r$ existing facilities, $y_1,\ldots,y_r$, and the new facilities at $x_1,\ldots,x_p$ as in problem (1.6): the demand from users in their way from $u$ to $v$ captured by each facility is inversely proportional to a positive nondecreasing function of the length of the path from the origin to the destination via the facility.

In other words, the demand captured by new facilities from users in their trip from origin $u$ to destination $v$ is given by

$$\omega_{uv}\frac{\displaystyle\sum_{j=1}^{p}1/\varphi_{uv}(d(u,x_j)+d(x_j,v))}{\displaystyle\sum_{j=1}^{p}1/\varphi_{uv}(d(u,x_j)+d(x_j,v))+\sum_{j=1}^{r}1/\varphi_{uv}(d(u,y_j)+d(y_j,v))}. \tag{1.8}$$

As in (1.4), the goal of the entering firm is the maximization of its market share. This is written as the following optimization problem:

$$\max_{\substack{x_1\in[0,l_{e_1}],\ldots,x_p\in[0,l_{e_p}]\\ e_1,\ldots,e_p\in E}}\sum_{u,v\in V}\omega_{uv}\frac{\displaystyle\sum_{j=1}^{p}1/\varphi_{uv}(d(u,x_j)+d(x_j,v))}{\displaystyle\sum_{j=1}^{p}1/\varphi_{uv}(d(u,x_j)+d(x_j,v))+\sum_{j=1}^{r}1/\varphi_{uv}(d(u,y_j)+d(y_j,v))}. \tag{1.9}$$

Defining for each $u,v\in V$ the positive constant $\beta_{uv}$,

$$\beta_{uv}=\sum_{j=1}^{r}\frac{1}{\varphi_{uv}(d(u,y_j)+d(y_j,v))},$$

it follows that Problem (1.9) can be rewritten as

$$\max_{\substack{x_1\in[0,l_{e_1}],\ldots,x_p\in[0,l_{e_p}]\\ e_1,\ldots,e_p\in E}}F(x_1,\ldots,x_p) \tag{1.10}$$

with $F$ defined as

$$F(x_1,\dots,x_p)=\sum_{u,v\in V}\omega_{uv}\frac{1}{1+\dfrac{\beta_{uv}}{\displaystyle\sum_{j=1}^{p}\dfrac{1}{\varphi_{uv}(d(u,x_j)+d(x_j,v))}}}. \tag{1.11}$$

## 1.3 Outline of Part I

In this part of the dissertation, Part I, we address two competitive location problems on networks. Chapter 2 studies the location of a single facility while the core of Chapter 3 is the location of multiple facilities.

In Chapter 2, the single-facility case of the Huff location problems on networks described in Sections 1.2.1 and 1.2.2 is addressed, by considering that users go directly to the facility or they visit the facility in their way to a destination respectively. Since the problems are multimodal, a branch and bound algorithm is proposed, in which two different bounding strategies, based on DC (difference of convex) and Interval Analysis (IA) tools, are used and compared. Computational results are given at the end of Chapter 2 for the two bounding procedures, showing that problems of rather realistic size can be solved in reasonable time.

In Chapter 3, we study the $p$-facility Huff location problem on networks. We propose two approaches for the initialization and division of subproblems in the branch and bound method used for its resolution. The first one is based on the straightforward idea of enumerating every possible combination of $p$ edges of the network as possible locations, thus a direct extension of the method described in Chapter 2 for the single-facility case. The second one is based on defining new structures that exploit the combinatorial and continuous part of the problem. Bounding rules are designed using DC and IA tools. In the computational study reported at the end of Chapter 3, we compare the approaches on a battery of 21 networks and show that both approaches can handle problems for $p \leq 4$ in reasonable computing time.

## 1.4 Datasets for Part I

The problems described in Sections 1.2.1 and 1.2.2 are tested in Chapters 2 and 3 on a battery of real-life and artificial networks, whose characteristics are reported in Table 1.1. The first three columns of Table 1.1 report the network name, number of nodes and number of edges. The fourth and fifth columns show if the network is used in Chapters 2 and 3 respectively.

The first 9 networks are artificial networks generated as described in [9]. The following network is the 55-node and 134-edge Swain's network [112, 143], see Table 1.2 for a detailed structure of the network. It is used as a toy-example for testing the problems from Chapter 2. The following 5 networks are proposed for $p$-median problems in [14] and also used in [18]. Finally, the last 43 networks are from [53, 133].

Each problem solved in Chapters 2 and 3 is obtained by randomly and independently generating the demands and the location of the existing facilities. Each vertex of the network is assumed to have a demand uniformly distributed in the interval $(0, 1)$ and in $(0, 20)$ for the artificial networks from [9]. To generate the locations of the existing

facilities, $r$ edges are randomly chosen with replacement; then, on each edge, the facility location is generated following a uniform distribution.

Table 1.1: Properties of the networks taken from [9, 14, 53, 133].

| Network | nodes | edges | Chapter 2 | Chapter 3 |
|---|---|---|---|---|
| art1 | 20 | 38 | | ✔ |
| art2 | 20 | 43 | | ✔ |
| art3 | 20 | 51 | | ✔ |
| art4 | 30 | 56 | | ✔ |
| art5 | 30 | 71 | | ✔ |
| art6 | 30 | 84 | | ✔ |
| art7 | 40 | 74 | | ✔ |
| art8 | 40 | 95 | | ✔ |
| art9 | 40 | 115 | | ✔ |
| Swain | 55 | 134 | ✔ | |
| pmed1 | 100 | 196 | | ✔ |
| pmed2 | 100 | 191 | | ✔ |
| pmed3 | 100 | 196 | | ✔ |
| pmed4 | 100 | 194 | | ✔ |
| pmed5 | 100 | 194 | | ✔ |
| KROB150G | 150 | 296 | ✔ | ✔ |
| KROA150G | 150 | 297 | ✔ | ✔ |
| PR152G | 152 | 296 | ✔ | ✔ |
| RAT195G | 195 | 336 | ✔ | ✔ |
| KROB200G | 200 | 386 | ✔ | ✔ |
| KROA200G | 200 | 392 | ✔ | ✔ |
| TS225G | 225 | 306 | ✔ | ✔ |
| UR532 | 298 | 597 | ✔ | |
| UR542 | 343 | 862 | ✔ | |
| UR552 | 388 | 1135 | ✔ | |
| UR562 | 416 | 1403 | ✔ | |
| UR732 | 452 | 915 | ✔ | |
| UR535 | 458 | 812 | ✔ | |
| UR545 | 476 | 1104 | ✔ | |
| UR555 | 490 | 1305 | ✔ | |
| UR537 | 493 | 868 | ✔ | |
| UR565 | 496 | 1513 | ✔ | |
| UR547 | 498 | 1112 | ✔ | |
| UR557 | 498 | 1310 | ✔ | |
| UR567 | 499 | 1426 | ✔ | |
| UR742 | 538 | 1325 | ✔ | |
| UR752 | 580 | 1735 | ✔ | |
| UR762 | 593 | 2089 | ✔ | |
| UR132 | 605 | 1122 | ✔ | |
| UR735 | 662 | 1200 | ✔ | |
| UR142 | 709 | 1815 | ✔ | |
| UR745 | 713 | 1616 | ✔ | |
| UR755 | 724 | 1966 | ✔ | |
| UR765 | 741 | 2278 | ✔ | |
| UR737 | 744 | 1315 | ✔ | |
| UR747 | 745 | 1659 | ✔ | |
| UR757 | 748 | 1969 | ✔ | |
| UR767 | 749 | 2314 | ✔ | |
| UR152 | 766 | 2390 | ✔ | |
| UR162 | 802 | 2897 | ✔ | |
| UR135 | 892 | 1619 | ✔ | |
| UR145 | 929 | 2117 | ✔ | |
| UR155 | 975 | 2680 | ✔ | |
| UR137 | 980 | 1744 | ✔ | |
| UR165 | 980 | 3068 | ✔ | |
| UR147 | 996 | 2254 | ✔ | |
| UR157 | 1000 | 2690 | ✔ | |
| UR167 | 1000 | 3083 | ✔ | |

Table 1.2: The Swain dataset, [112, 143].

| Initial node | Final node | Edge length | Initial node | Final node | Edge length | Initial node | Final node | Edge length |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3.1623 | 13 | 47 | 4.4721 | 25 | 49 | 7.0000 |
| 1 | 5 | 2.0000 | 14 | 16 | 10.2956 | 26 | 36 | 8.6023 |
| 1 | 8 | 4.4721 | 14 | 22 | 12.0416 | 27 | 38 | 9.2195 |
| 1 | 13 | 2.2361 | 14 | 27 | 9.8489 | 27 | 54 | 7.2111 |
| 1 | 43 | 10.0000 | 15 | 18 | 6.4031 | 29 | 31 | 3.6056 |
| 1 | 44 | 4.1231 | 15 | 25 | 7.2801 | 30 | 33 | 4.0000 |
| 2 | 3 | 4.4721 | 15 | 31 | 7.0000 | 30 | 45 | 5.0000 |
| 2 | 4 | 3.0000 | 15 | 36 | 6.7082 | 32 | 33 | 5.0000 |
| 2 | 8 | 3.1623 | 15 | 41 | 6.3246 | 32 | 38 | 4.4721 |
| 2 | 42 | 2.8284 | 15 | 42 | 6.3246 | 32 | 45 | 4.0000 |
| 3 | 7 | 4.2426 | 16 | 22 | 6.7082 | 34 | 43 | 6.0828 |
| 3 | 8 | 3.1623 | 16 | 27 | 6.4031 | 34 | 45 | 3.0000 |
| 3 | 19 | 6.4031 | 16 | 32 | 6.3246 | 35 | 36 | 7.6158 |
| 3 | 30 | 5.0000 | 16 | 33 | 6.7082 | 35 | 48 | 7.6158 |
| 3 | 31 | 6.0828 | 16 | 38 | 7.2111 | 36 | 48 | 6.0000 |
| 3 | 34 | 5.3852 | 17 | 22 | 5.0990 | 37 | 47 | 10.0499 |
| 4 | 5 | 3.0000 | 17 | 23 | 5.3852 | 37 | 50 | 10.4403 |
| 4 | 9 | 2.0000 | 17 | 28 | 7.0000 | 37 | 53 | 7.8102 |
| 4 | 42 | 2.2361 | 18 | 23 | 9.0000 | 38 | 43 | 8.2462 |
| 5 | 11 | 1.4142 | 18 | 26 | 7.0711 | 38 | 45 | 7.2111 |
| 5 | 13 | 2.2361 | 18 | 29 | 5.3852 | 38 | 54 | 6.4031 |
| 6 | 9 | 3.6056 | 18 | 31 | 4.4721 | 38 | 55 | 7.8102 |
| 6 | 10 | 5.0000 | 18 | 36 | 8.2462 | 39 | 40 | 11.7047 |
| 6 | 15 | 5.8310 | 19 | 22 | 8.2462 | 39 | 54 | 6.0828 |
| 6 | 41 | 4.2426 | 19 | 23 | 5.3852 | 39 | 55 | 9.8489 |
| 6 | 42 | 5.0990 | 19 | 29 | 3.6056 | 40 | 46 | 8.9443 |
| 7 | 15 | 5.8310 | 19 | 30 | 5.0990 | 40 | 55 | 8.2462 |
| 7 | 31 | 3.6056 | 19 | 31 | 5.0990 | 43 | 44 | 7.2801 |
| 7 | 42 | 4.2426 | 20 | 21 | 4.4721 | 43 | 45 | 6.3246 |
| 8 | 34 | 3.6056 | 20 | 25 | 9.2195 | 43 | 46 | 5.0000 |
| 9 | 10 | 6.0000 | 20 | 41 | 8.2462 | 43 | 55 | 5.0000 |
| 9 | 11 | 4.1231 | 20 | 49 | 6.0000 | 44 | 46 | 7.2111 |
| 9 | 42 | 3.6056 | 20 | 51 | 9.2195 | 44 | 47 | 6.0000 |
| 10 | 20 | 8.0623 | 21 | 37 | 7.2111 | 46 | 47 | 11.6619 |
| 10 | 21 | 9.0000 | 21 | 51 | 7.2801 | 46 | 52 | 15.6205 |
| 10 | 37 | 7.8102 | 22 | 33 | 4.2426 | 46 | 55 | 6.0000 |
| 10 | 41 | 6.0828 | 23 | 26 | 12.2066 | 47 | 52 | 16.1245 |
| 10 | 47 | 8.6023 | 23 | 28 | 7.0711 | 47 | 53 | 5.6569 |
| 11 | 13 | 2.2361 | 23 | 29 | 4.4721 | 48 | 49 | 6.4031 |
| 11 | 47 | 3.6056 | 24 | 26 | 7.2111 | 49 | 51 | 12.0416 |
| 12 | 14 | 10.6301 | 24 | 35 | 4.4721 | 50 | 52 | 8.6023 |
| 12 | 17 | 6.3246 | 24 | 36 | 9.0554 | 50 | 53 | 5.8310 |
| 12 | 22 | 8.6023 | 25 | 36 | 5.6569 | 52 | 53 | 12.1655 |
| 12 | 28 | 7.8102 | 25 | 41 | 4.1231 | 54 | 55 | 10.1980 |
| 13 | 44 | 2.8284 | 25 | 48 | 4.4721 | | | |

# Chapter 2

# Single-Facility Huff Location Problems on Networks

In this chapter [30] two Huff location problems on networks are addressed, by considering that users go directly to the facility or they visit the facility in their way to a destination. This chapter contributes to location analysis by proposing a branch and bound algorithm in which two different bounding strategies, based on IA tools and DC optimization, are used and compared.

## 2.1 Introduction

In this chapter we study the single-facility case of the two competitive location problems on networks presented in Chapter 1. The first problem is the classic Huff location problem, (1.6)-(1.7), in which customers perceive the facility attractiveness in terms of their distance to the facility, while the second problem is the Huff OD trip problem (1.10)-(1.11). In the OD trip problem the facility attraction is a function of the length of the shortest path from the origin to the destination through the facility.

We can see that both the location and the OD trip problems yield an optimization problem of the same form, namely, (1.6) or (1.10), with a rather similar function $F$, as given by (1.7) and (1.11) respectively. For the single-facility case, both functions $F$ can be written in the form

$$F(x) = \sum_{\delta \in \Delta} \omega_\delta \frac{1}{1 + \beta_\delta \varphi_\delta(d_\delta(x))}, \tag{2.1}$$

where

$$\Delta = V \text{ and, for } \delta \in \Delta,\ d_\delta(x) = d(\delta, x)$$

for the location problem, and

$$\Delta = \{\{u, v\} :\ u, v \in V\} \text{ and, for } \delta \in \Delta,\ d_\delta(x) = d(u, x) + d(x, v)$$

for the OD trip problem.

The remainder of this chapter is structured as follows. In Section 2.2 a branch and bound algorithm is designed to solve both problems and two different procedures to obtain bounds are presented. Computational results are given in Section 2.3, comparing

the two bounding strategies implemented. Finally, some conclusions are presented in Section 2.4.

## 2.2 Solving the problems

### 2.2.1 Multimodality

The optimization problems described in Sections 1.2.1 and 1.2.2 are, in general, multimodal, and standard optimization methods get stuck at local optima. This can be seen in simple examples, even when the network is a segment and is illustrated in the following examples for the Huff location problem (1.6)-(1.7).

First, data for one problem with 2 users and 1 facility on a segment was randomly generated. The objective function $F$ of such instance is plotted in Figure 2.1 (left). One can see that the problem is bimodal. 100 runs of a local search procedure starting with a random point were performed.

In Figure 2.1 (right) one can see the histogram of the objective values provided by the optimizer: below a 50% of the runs yielded the global optimum, whereas the remaining runs stopped at the local not globally optimal solution.

Another instance, with 500 users and 100 facilities was also generated. In Figure 2.2 the problem is shown to be multimodal, and just below a 14% of the runs solved with the optimizer yielded the global optimum.

In the right part of Figures 2.1-2.2 the $x$ axis is normalized, so that 1 corresponds to the best found objective value, $z^*$, and a bar at $x \in [0, 1]$ indicates that an objective value $x \cdot z^*$ was found.

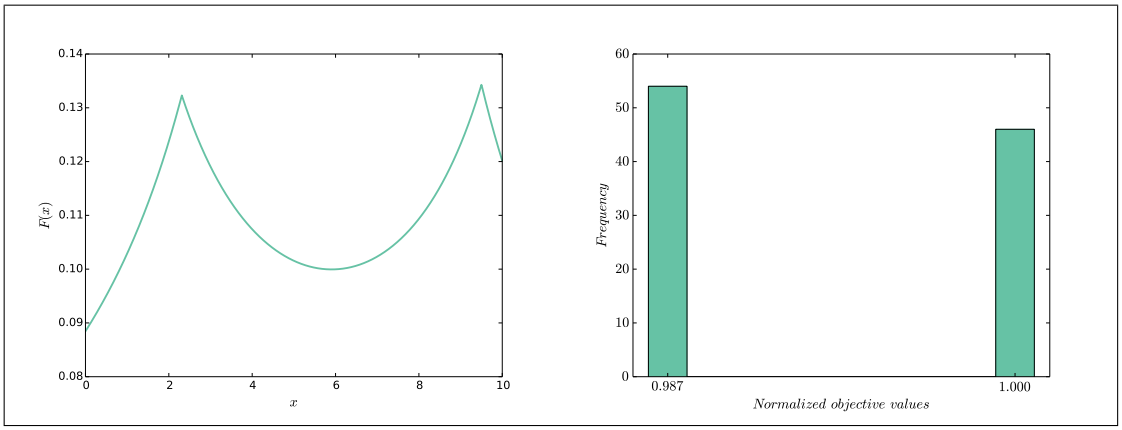


**Figure 2.1:** Example for $card(V) = 2$, $r = 1$, $\lambda_v = 1\ \forall v \in V$.

Since multimodality appears even in the simplest cases, and local search procedures can get stuck at very bad local optima, global optimization tools are needed if the global optimum is sought.

### 2.2.2 Algorithm

We will use a branch and bound algorithm [66, 87, 104, 128] to solve the problems under consideration. We first outline the algorithm that finds an optimal solution within a relative accuracy of $\varepsilon$, and later give the details on the bounding process, which will

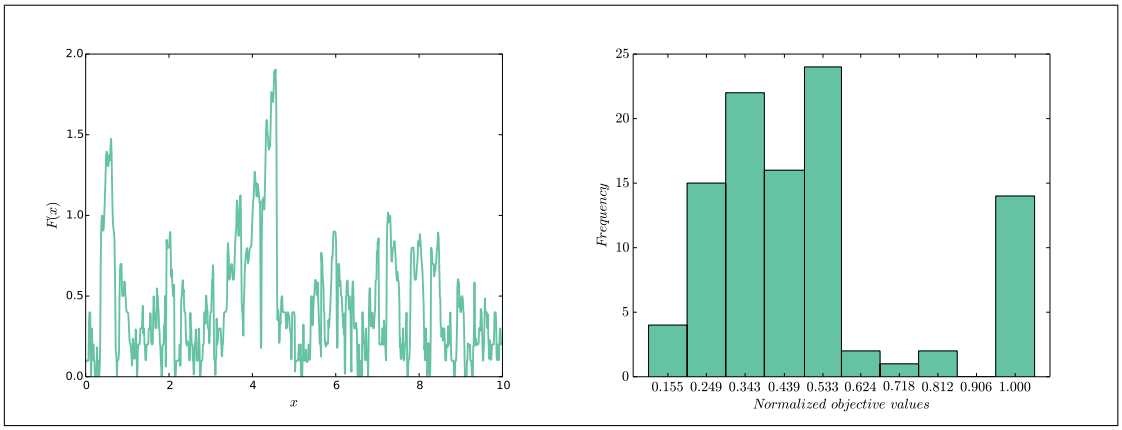


**Figure 2.2:** Example for $card(V) = 500$, $r = 100$, $\lambda_v = 20\ \forall v \in V$.

exploit monotonicity properties or the fact that the objective function is DC on each edge, [26, 92, 150, 151]. We remind the reader that a function $h$ is DC if it can be written as $h = h^+ - h^-$, where $h^+$, $h^-$ are both convex. The expression $h^+ - h^-$ is called a DC decomposition of $h$. In the description of the algorithm, we have $\Delta = V$ for the Huff location problem and $\Delta = \{\{u, v\} : u, v \in V\}$ for the Huff OD trip problem.

- Phase 1: Initialization
  - Fix the required accuracy $\varepsilon > 0$.
  - Initialize the lower bound ($LB$), $LB = 0$.
  - Compute the all-pairs distance matrix.
  - Calculate $\beta_\delta$, $\forall \delta \in \Delta$.
  - Set the list $H$ of remaining segments as empty.
- Phase 2: Prepare the list of segments
  - Consider the edge $e$ as segment with its nodes as the segment vertices.
  - The value of the objective function is evaluated at the segment midpoint. If the value is greater than $LB$, then $LB$ is updated to such value and the midpoint stored as incumbent.
  - Calculate an upper bound ($UB$) for the segment $e$, $UB(e)$.
  - In case $UB(e) \geq LB \cdot (1 + \varepsilon)$, insert $e$ into $H$.
- Phase 3: Branch and bound process.

  Repeat as long as no stop is reached:
  - Select from $H$ the highest upper bound, $UB_{max}$, with $LB$ as $\varepsilon$-optimal value and the incumbent as an $\varepsilon$-optimal solution.

    If $UB_{max} \leq LB \cdot (1 + \varepsilon)$, stop the algorithm with $LB$ as the optimal value.
  - The segment with the highest upper bound, $UB_{max}$, is selected for a split at its midpoint into two smaller segments.

- The value of the objective function at the midpoint of the two small segments is calculated.

  If any of these values is greater than $LB$, then $LB$ is updated and all segments from $H$ whose upper bound is lower than $LB$ are discarded.
- An upper bound for each small segment is calculated.
- All segments whose upper bound is greater than $LB \cdot (1 + \varepsilon)$ are added to $H$.

In the algorithm above, the segment midpoint yielding the best upper bound is given as $\varepsilon$-optimal solution. Observe that the full list of segments in $H$ contain all $\varepsilon$-optimal solutions, and can thus be used in a two-phase process, as suggested in the GBSSS algorithm, [128].

Let us detail the algorithm steps previously outlined. To calculate the all-pairs distance matrix we use the Floyd algorithm [21], which uses the edge length matrix to build recursively the distance matrix.

The computation of bounds requires more detail. The algorithm needs the calculation of an upper bound, $UB(s)$, for each segment $s$. We present two procedures, one based on IA tools, and the other on properties of DC functions.

### 2.2.3 Interval analysis bound

When the distance $d_\delta(x)$ decreases, the market share as given by the objective function (2.1) increases. Hence we obtain an upper bound on the market share $F(x)$ for any location $x$ on a segment $s = [x_0, x_1] \subset e \in E$ by replacing $d_\delta(x)$ by the lowest possible of these distances on the segment. Defining therefore such lowest distance for the location problem as in [18] $d_\delta(s) = \min\{d(v, x_0), d(v, x_1)\}$ for $\delta = v$, and for the OD trip problem as $d_\delta(s) = \min\{d(u, x_0) + d(v, x_0), d(u, x_1) + d(v, x_1)\}$ for $\delta = \{u, v\}$, we obtain the IA bound

$$UB^{IA}(s) = \sum_{\delta \in \Delta} \omega_\delta \frac{1}{1 + \beta_\delta \varphi_\delta(d_\delta(s))}. \tag{2.2}$$

### 2.2.4 DC bound

An upper bound obtained by making use of the fact that the objective function is DC on each edge exploits the following properties:

**Proposition 2.1.** *Let $I \subset \mathbb{R}$ be an interval. Let $d : I \to \mathbb{R}$ be a concave function on $I$, and let $g : \mathbb{R} \to \mathbb{R}$ be DC, with a DC decomposition given by $g(x) = g^+(x) - g^-(x)$, with both $g^+$ and $g^-$ nonincreasing functions. Then, the function $f : I \to \mathbb{R}$ defined as $f(x) = g(d(x))$ is DC on $I$ and a DC decomposition is given by $f(x) = f^+(x) - f^-(x)$, where $f^+(x) = g^+(d(x))$ and $f^-(x) = g^-(d(x))$.*

*Proof:* The proof follows directly from the fact that the compositions $g^+(d(x))$ and $g^-(d(x))$ are also convex functions. □

**Remark 2.2.** Proposition 2.1 makes use of a function $g$ which can be written as the difference of two convex functions, $g^+$ and $g^-$. Since such convex functions are also nonincreasing, it turns out that $g$ belongs to a subclass of DC functions, namely DCM functions, as introduced in [25], which are those functions expressed as the difference of two convex monotonic functions, as $g^+, g^-$ are. See [25] for further properties.

We are now in position to give a bound for $F$ on an edge exploiting the fact that $F$ is DC. Let $d_\delta(x)$ be the concave function given in (1.1). Assuming $\varphi_\delta(d) = d_\delta^\lambda$, (2.1) can be rewritten as

$$F(x) = \sum_{\delta \in \Delta} \omega_\delta \frac{1}{1 + \beta_\delta d_\delta^\lambda(x)}. \tag{2.3}$$

Let us define a simpler function:

$$g(t) = \frac{1}{1 + \beta t^\lambda}. \tag{2.4}$$

The following DC decomposition [15] is available for function $g$, $g(t) = g^+(t) - g^-(t)$ where:

$$g^+(t) = \begin{cases} g(c) + g'(c)(t-c) & if \;\; t \le c \\ g(t) & if \;\; t > c \end{cases}$$

$$g^-(t) = \begin{cases} g(c) + g'(c)(t-c) - g(t) & if \;\; t \le c \\ 0 & if \;\; t > c \end{cases}$$

$$c = \left( \frac{\lambda - 1}{(\lambda + 1)\beta} \right)^{1/\lambda}.$$

Applying Proposition 2.1 we have that a DC decomposition for $F$ as defined in (2.3) is:

$$F(x) = \sum_{\delta \in \Delta} F_\delta^+(x) - \sum_{\delta \in \Delta} F_\delta^-(x) = \sum_{\delta \in \Delta} (F_\delta^+(x) - F_\delta^-(x)) \tag{2.5}$$

where:

$$F_\delta^+(x) = \omega_\delta \, g^+(d_\delta(x)),$$

$$F_\delta^-(x) = \omega_\delta \, g^-(d_\delta(x)).$$

To construct an upper bound, $\mathrm{UB}^{DC}$, first we obtain a convex minorant of $F_\delta^-(x)$ as in [25]:

$$F_\delta^-(x) \ge F_\delta^-(x_0) + \xi_\delta(x - x_0)$$

$$F(x) \le \sum_{\delta \in \Delta} (F_\delta^+(x) - F_\delta^-(x_0) - \xi_\delta(x - x_0)) \qquad \text{for } \xi_\delta \in \partial F_\delta^-(x_0), \tag{2.6}$$

where $\partial F_\delta^-(x_0)$ denotes the set of subgradients (subdifferential) of $F_\delta^-$ at $x_0$ [150].

Define for each $\delta \in \Delta$ the function from (2.6):

$$U(x) = \sum_{\delta \in \Delta} (F_\delta^+(x) - F_\delta^-(x_0) - \xi_\delta(x - x_0)) \qquad \text{for } \xi_\delta \in \partial F_\delta^-(x_0). \tag{2.7}$$

Since $U$ is convex, an upper bound for a segment $s$ is obtained:

$$UB^{DC}(s) = \max\{U(v_1), U(v_2)\}, \tag{2.8}$$

$v_1$, $v_2$ being vertices of $s$.

## 2.3 Computational results

The algorithm described in Section 2.2 was programmed in an Intel Fortran Compiler XE 12.0 and executed using an Intel Core i7 computer with 8.00 Gb of RAM memory at 2.8 Ghz. The solutions were found within an accuracy of $10^{-10}$ and for $\lambda_v = 2, \forall v \in V$.

The Huff location problem was first tested on the 55-node and 134-edge Swain's network [143, 112], see Table 1.2, with both bounding strategies.

Several instances of the problem were generated using different values for the number $r$ of existing facilities, ranging from low-saturated markets ($r = 10\%$ of the number of edges of the network, $card(E)$) to high-saturated markets ($r = 90\% card(E)$). For each value of $r$, 10 different problems were solved. The results are shown in Table 2.1.

Table 2.1: Results of the 55-node and 134-edge Swain's network for the Huff location problem.

| | $r(\%card(E))$ | iter | | | | MaxList | | | | time | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | min | max | mean | std | min | max | mean | std | min | max | mean | std |
| DC | 10 | 111 | 196 | 144.40 | 34.68 | 9 | 61 | 31.50 | 17.25 | 0.00 | 0.02 | 0.00 | 0.00 |
| | 20 | 80 | 193 | 140.10 | 34.32 | 15 | 83 | 42.70 | 22.76 | 0.00 | 0.00 | 0.00 | 0.00 |
| | 30 | 51 | 209 | 117.80 | 47.86 | 25 | 96 | 66.70 | 22.42 | 0.00 | 0.00 | 0.00 | 0.00 |
| | 40 | 50 | 256 | 176.40 | 55.33 | 15 | 107 | 84.80 | 30.51 | 0.00 | 0.02 | 0.00 | 0.00 |
| | 50 | 81 | 290 | 172.20 | 67.40 | 74 | 126 | 107.50 | 14.92 | 0.00 | 0.02 | 0.00 | 0.00 |
| | 60 | 121 | 317 | 166.50 | 60.54 | 72 | 121 | 110.10 | 16.09 | 0.00 | 0.02 | 0.00 | 0.00 |
| | 70 | 94 | 323 | 190.00 | 64.79 | 113 | 131 | 123.80 | 6.36 | 0.00 | 0.02 | 0.00 | 0.00 |
| | 80 | 145 | 332 | 209.70 | 58.43 | 117 | 133 | 123.70 | 5.62 | 0.00 | 0.02 | 0.00 | 0.01 |
| | 90 | 137 | 296 | 211.40 | 55.51 | 111 | 129 | 121.40 | 5.66 | 0.00 | 0.02 | 0.00 | 0.00 |
| IA | 10 | 158 | 318 | 223.70 | 50.81 | 26 | 120 | 64.20 | 31.72 | 0.00 | 0.02 | 0.00 | 0.00 |
| | 20 | 185 | 178915 | 18100.40 | 56504.50 | 29 | 58170 | 5890.10 | 18369.30 | 0.00 | 0.59 | 0.06 | 0.19 |
| | 30 | 203 | 70911 | 17779.60 | 25278.44 | 75 | 25465 | 6219.20 | 8904.76 | 0.00 | 0.20 | 0.05 | 0.07 |
| | 40 | 243 | 243922 | 24639.40 | 77048.06 | 96 | 84425 | 8551.40 | 26659.27 | 0.00 | 0.75 | 0.08 | 0.24 |
| | 50 | 187 | 37594 | 9132.40 | 13971.49 | 118 | 13306 | 3143.50 | 4794.67 | 0.00 | 0.11 | 0.03 | 0.05 |
| | 60 | 186 | 29111 | 7734.70 | 12130.03 | 118 | 9927 | 2703.30 | 4174.80 | 0.00 | 0.09 | 0.03 | 0.04 |
| | 70 | 212 | 10748 | 3029.00 | 4148.73 | 138 | 3489 | 1038.00 | 1349.47 | 0.00 | 0.03 | 0.01 | 0.02 |
| | 80 | 172 | 1950 | 866.60 | 677.23 | 127 | 648 | 310.10 | 192.99 | 0.00 | 0.02 | 0.00 | 0.00 |
| | 90 | 224 | 22520 | 4243.00 | 7225.49 | 147 | 7248 | 1442.20 | 2338.18 | 0.00 | 0.09 | 0.02 | 0.03 |

The percentage $r$ of existing facilities is shown in the first column. Then, it is reported the minimum, maximum, mean and standard deviation (std) for the number of iterations (iter), i.e., the number of executions of Phase 3, the maximum size of the branch and bound tree (MaxList) during execution and the CPU time in seconds. In all cases, the best solution is found in less than 0.02 seconds for DC bound, and 0.75 seconds for IA bound. It is remarkable that DC bound leads to a very stable procedure, as can be seen for all values of $r$, in memory requirements as well as computational time. On the other hand, when using IA bound one can see very extreme cases. There is always a huge difference between the minimum and maximum for the number of iterations, branch and bound list size and time. This means that in the ten runs that are solved for each value of $r$, IA bound is quite erratic for problems of the same difficulty. Therefore, we have an algorithm that when using DC bound spends the same resources (computing time) than when using IA bound, but is much more stable and reliable.

Afterwards the algorithm was tested for both problems on a battery of test instances of larger dimensions (up to 1000 nodes and 3083 edges) to analyze the dependence of running time and memory requirements with respect to the size of the network and the number of facilities for both bounding strategies.

To attain this end, the problems were tested on 43 networks obtained from [1, 53, 133], see Table 1.1. For $r = 10\%card(E)$ and $r = 90\%card(E)$ of existing facilities, 10 instances of the problem were solved. Results for the comparison between the two methods for obtaining bounds are shown in the following tables. Results for the Huff location problem are shown in Tables 2.2 and 2.3 comparing both bounding procedures applied in Phase 2 and Phase 3 of the algorithm. Results for the Huff OD trip problem are shown in Tables 2.4 and 2.5. Results from Swain's network, summarized in Table 2.1, show that DC bound seems to be sharper than IA bound but more computationally expensive; therefore, it seems desirable to delete segments in the initialization (Phase 2 of the algorithm) using easy bound (IA), and then use DC bound for the rest of the segments (Phase 3). Hence, let us note that for the OD trip problem, IA bound is used always in Phase 2 of the algorithm, and then, in Phase 3, the two different bounding procedures are compared. This strategy was also tested for the Huff location problem with no improvement and therefore not used.

Table 2.2: Results of test instances for the Huff location problem with 10%$card(E)$ facilities.

| | DC bound | | | | | | IA bound | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Network | iter | | MaxList | | time | | iter | | MaxList | | time | |
| | mean | std | mean | std | mean | std | mean | std | mean | std | mean | std |
| KROB150G | 136.00 | 18.38 | 28.60 | 11.64 | 0.01 | 0.01 | 256.80 | 53.90 | 71.70 | 25.31 | 0.01 | 0.01 |
| KROA150G | 147.30 | 21.31 | 22.50 | 9.17 | 0.01 | 0.01 | 257.40 | 60.95 | 62.30 | 19.75 | 0.01 | 0.01 |
| PR152G | 104.20 | 46.91 | 34.40 | 7.92 | 0.01 | 0.01 | 51433.40 | 98074.47 | 16950.10 | 32330.65 | 0.40 | 0.76 |
| RAT195G | 110.00 | 22.32 | 46.60 | 21.96 | 0.02 | 0.00 | 254.00 | 54.23 | 130.40 | 51.90 | 0.01 | 0.01 |
| KROB200G | 127.90 | 39.50 | 31.70 | 14.60 | 0.01 | 0.00 | 12954.80± | 26959.17 | 4329.50 | 9047.62 | 0.15 | 0.27 |
| KROA200G | 142.70 | 28.06 | 29.70 | 11.83 | 0.02 | 0.00 | 249.60 | 37.98 | 78.80 | 25.49 | 0.01 | 0.01 |
| TS225G | 104.90 | 15.20 | 30.40 | 9.65 | 0.02 | 0.01 | 182.40 | 31.70 | 67.10 | 20.12 | 0.02 | 0.00 |
| UR532 | 114.80 | 36.14 | 21.90 | 9.28 | 0.04 | 0.01 | 44770.00 | 140751.59 | 16124.10 | 50692.35 | 0.72 | 2.17 |
| UR542 | 131.30 | 50.06 | 44.10 | 16.29 | 0.06 | 0.00 | 40444.00 | 81734.28 | 13882.90 | 28532.14 | 0.77 | 1.45 |
| UR552 | 173.00 | 30.49 | 48.50 | 13.94 | 0.10 | 0.01 | 365.40 | 145.20 | 197.30 | 94.94 | 0.09 | 0.01 |
| UR562 | 204.30 | 35.22 | 49.90 | 14.58 | 0.12 | 0.01 | 388.30 | 113.50 | 262.30 | 116.58 | 0.11 | 0.00 |
| UR732 | 99.30 | 29.19 | 36.00 | 13.69 | 0.11 | 0.01 | 25906.60 | 81051.80 | 9207.20 | 28797.07 | 0.71 | 1.89 |
| UR535 | 122.00 | 17.33 | 32.20 | 12.95 | 0.12 | 0.01 | 214.80 | 42.78 | 74.50 | 32.52 | 0.11 | 0.00 |
| UR545 | 173.60 | 21.97 | 55.20 | 22.23 | 0.15 | 0.01 | 268.90 | 41.01 | 140.40 | 35.71 | 0.14 | 0.01 |
| UR555 | 169.50 | 68.40 | 57.40 | 22.45 | 0.17 | 0.01 | 12713.60 | 39128.99 | 4248.30 | 12756.97 | 0.48 | 1.01 |
| UR537 | 130.00 | 27.01 | 41.30 | 9.44 | 0.14 | 0.00 | 225.40 | 46.83 | 82.30 | 29.69 | 0.13 | 0.01 |
| UR565 | 155.40 | 56.02 | 60.70 | 31.20 | 0.18 | 0.01 | 26442.70 | 54904.79 | 9154.30 | 18533.07 | 0.85 | 1.42 |
| UR547 | 148.70 | 35.15 | 52.20 | 27.28 | 0.16 | 0.01 | 283.50 | 44.66 | 158.50 | 84.24 | 0.15 | 0.01 |
| UR557 | 168.00 | 47.13 | 71.90 | 32.18 | 0.18 | 0.01 | 16840.90 | 52028.01 | 5471.30 | 16692.26 | 0.61 | 1.40 |
| UR567 | 180.60 | 65.51 | 72.00 | 17.67 | 0.18 | 0.01 | 8613.60 | 25838.69 | 3126.20 | 8980.20 | 0.39 | 0.68 |
| UR742 | 166.30 | 69.15 | 48.40 | 13.79 | 0.22 | 0.01 | 5654.30 | 16741.63 | 2084.40 | 5937.85 | 0.37 | 0.51 |
| UR752 | 179.60 | 50.92 | 57.20 | 12.93 | 0.29 | 0.01 | 401.10 | 137.60 | 285.80 | 94.01 | 0.28 | 0.01 |
| UR762 | 187.00 | 34.16 | 88.00 | 26.85 | 0.33 | 0.01 | 511.90 | 129.44 | 554.50 | 131.18 | 0.31 | 0.01 |
| UR132 | 126.60 | 22.12 | 26.10 | 11.78 | 0.27 | 0.00 | 224.20 | 44.83 | 95.00 | 67.85 | 0.26 | 0.01 |
| UR735 | 116.70 | 34.47 | 30.20 | 12.49 | 0.33 | 0.01 | 47379.90 | 149070.53 | 16336.00 | 51404.94 | 2.22 | 6.04 |
| UR142 | 171.70 | 40.81 | 40.50 | 14.43 | 0.46 | 0.01 | 300.30 | 77.67 | 161.10 | 62.28 | 0.44 | 0.01 |
| UR745 | 133.40 | 40.35 | 47.20 | 12.93 | 0.44 | 0.00 | 22245.80 | 69367.84 | 7879.60 | 24392.62 | 1.36 | 2.96 |
| UR755 | 177.60 | 49.02 | 61.70 | 22.51 | 0.49 | 0.01 | 8604.90 | 25951.73 | 3014.50 | 8640.13 | 0.83 | 1.06 |
| UR765 | 185.80 | 61.95 | 90.50 | 39.18 | 0.58 | 0.01 | 388.30 | 159.45 | 373.50 | 185.09 | 0.55 | 0.01 |
| UR737 | 146.20 | 47.17 | 36.70 | 13.47 | 0.44 | 0.01 | 6668.80 | 20303.00 | 2269.40 | 6874.07 | 0.72 | 0.90 |
| UR747 | 137.80 | 38.54 | 49.90 | 16.42 | 0.50 | 0.02 | 22822.50 | 71347.15 | 8003.30 | 24891.27 | 1.47 | 3.14 |
| UR757 | 172.10 | 27.42 | 75.40 | 20.79 | 0.55 | 0.00 | 342.90 | 109.03 | 306.70 | 121.76 | 0.53 | 0.01 |
| UR767 | 186.10 | 26.79 | 55.60 | 21.85 | 0.58 | 0.02 | 432.40 | 82.81 | 335.20 | 135.51 | 0.56 | 0.01 |
| UR152 | 205.10 | 70.77 | 68.00 | 21.48 | 0.63 | 0.01 | 472.50 | 163.57 | 395.90 | 149.69 | 0.61 | 0.03 |
| UR162 | 182.60 | 98.28 | 63.80 | 25.69 | 0.78 | 0.01 | 29666.00 | 61694.27 | 10082.00 | 20547.08 | 2.13 | 2.93 |
| UR135 | 124.60 | 21.78 | 35.20 | 13.51 | 0.77 | 0.03 | 207.90 | 32.32 | 87.80 | 29.58 | 0.73 | 0.01 |
| UR145 | 137.80 | 44.58 | 45.40 | 23.59 | 0.96 | 0.01 | 902.00 | 2002.12 | 362.00 | 672.69 | 0.98 | 0.11 |
| UR155 | 173.90 | 36.48 | 49.60 | 14.58 | 1.24 | 0.03 | 451.60 | 424.32 | 223.70 | 46.83 | 1.19 | 0.03 |
| UR137 | 127.10 | 22.33 | 31.50 | 15.04 | 1.00 | 0.03 | 208.30 | 31.13 | 88.00 | 42.40 | 0.97 | 0.00 |
| UR165 | 189.90 | 57.79 | 117.10 | 22.39 | 1.32 | 0.02 | 27238.10 | 84692.99 | 9609.40 | 28744.20 | 2.86 | 4.95 |
| UR147 | 144.00 | 30.24 | 69.20 | 13.28 | 1.13 | 0.03 | 296.70 | 94.75 | 222.10 | 109.46 | 1.10 | 0.01 |
| UR157 | 157.70 | 27.09 | 92.30 | 32.34 | 1.29 | 0.03 | 392.80 | 179.80 | 384.40 | 174.79 | 1.24 | 0.02 |
| UR167 | 200.20 | 52.30 | 78.30 | 28.60 | 1.32 | 0.02 | 487.50 | 270.66 | 425.40 | 213.96 | 1.29 | 0.03 |

Table 2.3: Results of test instances for the Huff location problem with $90\% card(E)$ facilities.

| | DC bound | | | | | | IA bound | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Network | iter | | MaxList | | time | | iter | | MaxList | | time | |
| | mean | std | mean | std | mean | std | mean | std | mean | std | mean | std |
| KROB150G | 124.70 | 114.90 | 111.80 | 56.52 | 0.01 | 0.01 | 122523.30 | 86721.97 | 42553.00 | 30707.86 | 0.92 | 0.65 |
| KROA150G | 57.80 | 34.26 | 95.40 | 39.59 | 0.01 | 0.01 | 122187.70 | 71891.54 | 41687.50 | 24061.80 | 0.92 | 0.54 |
| PR152G | 84.20 | 62.10 | 146.90 | 38.55 | 0.01 | 0.01 | 125552.60 | 116399.32 | 43041.20 | 40982.46 | 0.97 | 0.89 |
| RAT195G | 193.40 | 75.58 | 220.00 | 47.86 | 0.02 | 0.00 | 23516.10 | 20715.10 | 8296.90 | 7377.71 | 0.25 | 0.21 |
| KROB200G | 55.80 | 29.67 | 120.60 | 76.71 | 0.02 | 0.00 | 133369.40 | 87608.49 | 45817.30 | 30144.88 | 1.35 | 0.88 |
| KROA200G | 92.20 | 57.67 | 134.30 | 62.82 | 0.02 | 0.00 | 85623.70 | 83432.90 | 29232.40 | 28551.07 | 0.87 | 0.84 |
| TS225G | 114.30 | 44.14 | 157.60 | 57.11 | 0.02 | 0.01 | 54545.00 | 68301.12 | 19091.30 | 24086.07 | 0.63 | 0.76 |
| UR532 | 78.70 | 37.54 | 116.50 | 67.46 | 0.04 | 0.01 | 118298.20 | 96988.07 | 40879.50 | 34653.07 | 1.82 | 1.45 |
| UR542 | 73.50 | 26.35 | 267.30 | 168.52 | 0.07 | 0.01 | 96761.40 | 66415.73 | 33537.40 | 23163.56 | 1.77 | 1.18 |
| UR552 | 92.30 | 38.03 | 335.20 | 180.15 | 0.10 | 0.01 | 80499.70 | 70906.95 | 27384.10 | 24108.84 | 1.69 | 1.43 |
| UR562 | 102.50 | 112.03 | 348.30 | 283.73 | 0.12 | 0.01 | 155678.90 | 101564.05 | 54765.40 | 36433.31 | 3.45 | 2.17 |
| UR732 | 52.30 | 14.19 | 83.70 | 53.48 | 0.12 | 0.01 | 146740.60 | 84982.85 | 49974.80 | 28853.93 | 3.51 | 1.98 |
| UR535 | 58.80 | 25.82 | 90.80 | 56.03 | 0.12 | 0.01 | 124581.70 | 75273.00 | 42193.70 | 24984.67 | 3.06 | 1.78 |
| UR545 | 81.50 | 36.51 | 187.10 | 57.12 | 0.15 | 0.01 | 133664.90 | 120595.19 | 46760.60 | 43088.04 | 3.44 | 3.02 |
| UR555 | 137.80 | 97.07 | 385.00 | 174.05 | 0.18 | 0.01 | 115100.50 | 101518.18 | 39688.50 | 35018.62 | 3.12 | 2.62 |
| UR537 | 64.90 | 51.69 | 162.20 | 93.81 | 0.14 | 0.00 | 122309.80 | 77142.00 | 41193.00 | 26456.18 | 3.30 | 2.01 |
| UR565 | 85.90 | 68.70 | 227.20 | 238.52 | 0.19 | 0.01 | 156940.30 | 123176.94 | 54343.30 | 43896.68 | 4.25 | 3.20 |
| UR547 | 73.40 | 31.36 | 382.70 | 185.60 | 0.17 | 0.01 | 85720.80 | 81049.90 | 29284.20 | 27294.53 | 2.38 | 2.11 |
| UR557 | 100.50 | 41.85 | 422.40 | 197.55 | 0.18 | 0.01 | 88274.50 | 68354.12 | 30293.50 | 23842.36 | 2.47 | 1.79 |
| UR567 | 69.40 | 30.50 | 535.90 | 265.81 | 0.19 | 0.01 | 176179.80 | 141980.77 | 61802.30 | 50933.74 | 4.77 | 3.69 |
| UR742 | 91.10 | 64.65 | 291.10 | 236.64 | 0.22 | 0.01 | 106046.20 | 76582.35 | 37192.30 | 27427.96 | 3.29 | 2.27 |
| UR752 | 94.20 | 79.80 | 379.00 | 242.29 | 0.30 | 0.01 | 165162.90 | 111967.65 | 57192.20 | 39367.44 | 5.73 | 3.72 |
| UR762 | 145.80 | 85.85 | 468.60 | 344.81 | 0.34 | 0.01 | 162436.40 | 79937.30 | 57279.50 | 28859.45 | 5.72 | 2.64 |
| UR132 | 62.10 | 20.57 | 179.70 | 106.36 | 0.27 | 0.00 | 145409.40 | 119728.83 | 50053.60 | 40595.93 | 5.33 | 4.24 |
| UR735 | 70.40 | 36.73 | 127.10 | 78.70 | 0.33 | 0.01 | 133680.00 | 78116.64 | 46433.90 | 27209.94 | 5.52 | 3.13 |
| UR142 | 90.10 | 56.57 | 207.40 | 166.81 | 0.47 | 0.02 | 178459.20 | 112835.73 | 62576.90 | 40798.83 | 7.72 | 4.46 |
| UR745 | 78.70 | 60.75 | 170.80 | 128.32 | 0.45 | 0.01 | 101158.10 | 34508.89 | 34965.60 | 12181.32 | 4.51 | 1.36 |
| UR755 | 76.80 | 45.70 | 450.70 | 206.22 | 0.51 | 0.01 | 142853.50 | 72731.98 | 49434.30 | 25240.04 | 6.52 | 3.12 |
| UR765 | 107.40 | 120.99 | 683.90 | 321.98 | 0.59 | 0.02 | 139391.60 | 95422.39 | 48571.10 | 33389.80 | 6.60 | 4.16 |
| UR737 | 50.60 | 10.00 | 137.40 | 64.96 | 0.46 | 0.01 | 171229.60 | 88598.53 | 59189.20 | 31967.97 | 7.54 | 3.57 |
| UR747 | 100.00 | 69.53 | 436.30 | 190.51 | 0.50 | 0.01 | 103101.70 | 86649.55 | 35734.40 | 30112.29 | 4.88 | 3.72 |
| UR757 | 111.40 | 122.22 | 565.50 | 253.79 | 0.57 | 0.02 | 123438.30 | 55123.42 | 42512.80 | 18497.15 | 5.86 | 2.27 |
| UR767 | 106.30 | 51.87 | 402.80 | 282.60 | 0.59 | 0.01 | 114120.10 | 77627.96 | 39036.10 | 27210.09 | 5.49 | 3.49 |
| UR152 | 183.70 | 215.90 | 666.50 | 256.88 | 0.65 | 0.02 | 146234.20 | 95509.84 | 50240.20 | 33865.44 | 7.10 | 4.20 |
| UR162 | 222.00 | 190.88 | 949.20 | 277.23 | 0.81 | 0.02 | 150684.40 | 74064.68 | 52525.50 | 25731.91 | 7.88 | 3.51 |
| UR135 | 48.10 | 20.44 | 166.90 | 82.96 | 0.76 | 0.01 | 123350.90 | 59285.41 | 42688.70 | 20694.34 | 7.17 | 3.07 |
| UR145 | 66.30 | 31.51 | 237.90 | 183.17 | 0.98 | 0.01 | 163381.20 | 83966.41 | 56867.10 | 29219.51 | 9.97 | 4.67 |
| UR155 | 147.00 | 208.69 | 764.30 | 242.78 | 1.25 | 0.02 | 112298.20 | 69895.72 | 39005.50 | 25084.91 | 7.66 | 4.07 |
| UR137 | 91.00 | 104.47 | 342.10 | 185.67 | 1.04 | 0.03 | 93091.80 | 49939.66 | 31886.30 | 17344.81 | 6.33 | 2.80 |
| UR165 | 114.10 | 62.51 | 688.40 | 247.01 | 1.34 | 0.02 | 173681.80 | 107225.32 | 59126.40 | 35851.23 | 11.62 | 6.26 |
| UR147 | 110.10 | 95.34 | 381.20 | 244.67 | 1.14 | 0.01 | 96126.40 | 102777.49 | 33813.10 | 36415.99 | 6.73 | 6.10 |
| UR157 | 73.20 | 24.66 | 538.00 | 136.81 | 1.28 | 0.01 | 161473.50 | 99420.21 | 56795.40 | 34944.47 | 10.67 | 5.86 |
| UR167 | 87.80 | 34.37 | 615.20 | 256.40 | 1.34 | 0.01 | 108227.60 | 112302.86 | 37786.50 | 40089.84 | 7.64 | 6.57 |

Table 2.4: Results of test instances for the Huff OD trip problem with $10\% card(E)$ facilities.

| | DC bound | | | | | | IA bound | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Network | iter | | MaxList | | time | | iter | | MaxList | | time | |
| | mean | std | mean | std | mean | std | mean | std | mean | std | mean | std |
| KROB150G | 97.20 | 21.93 | 46.60 | 6.82 | 0.19 | 0.03 | 131.00 | 9.66 | 46.60 | 6.82 | 0.17 | 0.01 |
| KROA150G | 124.20 | 29.94 | 39.50 | 8.44 | 0.21 | 0.03 | 156.90 | 17.88 | 39.50 | 8.44 | 0.18 | 0.01 |
| PR152G | 108.30 | 27.86 | 46.70 | 7.76 | 0.20 | 0.03 | 124.90 | 11.58 | 46.70 | 7.76 | 0.17 | 0.01 |
| RAT195G | 82.50 | 11.04 | 83.60 | 8.93 | 0.32 | 0.02 | 117.10 | 16.02 | 83.60 | 8.93 | 0.29 | 0.01 |
| KROB200G | 121.40 | 16.03 | 34.00 | 10.26 | 0.43 | 0.03 | 141.70 | 8.06 | 34.00 | 10.26 | 0.36 | 0.01 |
| KROA200G | 94.00 | 27.53 | 30.10 | 3.28 | 0.38 | 0.05 | 131.30 | 6.60 | 30.30 | 3.53 | 0.35 | 0.01 |
| TS225G | 97.50 | 14.49 | 55.40 | 9.81 | 0.43 | 0.04 | 123.50 | 3.78 | 55.40 | 9.81 | 0.38 | 0.01 |
| UR532 | 120.80 | 13.26 | 54.50 | 8.75 | 1.29 | 0.06 | 157.30 | 12.56 | 54.60 | 8.95 | 1.11 | 0.03 |
| UR542 | 163.70 | 28.17 | 131.70 | 21.41 | 2.45 | 0.16 | 228.50 | 24.53 | 133.00 | 20.74 | 2.19 | 0.07 |
| UR552 | 127.30 | 22.23 | 95.80 | 5.98 | 3.72 | 0.18 | 212.50 | 12.03 | 95.80 | 5.98 | 3.64 | 0.06 |
| UR562 | 290.90 | 59.12 | 194.20 | 12.87 | 7.02 | 0.60 | 372.10 | 35.92 | 207.00 | 19.54 | 6.04 | 0.26 |
| UR732 | 95.80 | 18.45 | 42.20 | 4.05 | 4.45 | 0.21 | 131.00 | 6.45 | 42.20 | 4.05 | 4.27 | 0.08 |
| UR535 | 130.20 | 8.66 | 36.50 | 8.97 | 4.69 | 0.11 | 125.00 | 21.75 | 36.60 | 9.06 | 4.02 | 0.19 |
| UR545 | 150.10 | 30.86 | 88.60 | 8.30 | 6.68 | 0.43 | 148.80 | 6.29 | 88.60 | 8.30 | 5.90 | 0.24 |
| UR555 | 160.70 | 27.28 | 115.90 | 25.43 | 8.22 | 0.42 | 208.00 | 13.23 | 115.90 | 25.43 | 7.83 | 0.40 |
| UR537 | 83.60 | 15.09 | 39.80 | 12.24 | 5.23 | 0.22 | 125.20 | 13.96 | 39.80 | 12.24 | 5.04 | 0.12 |
| UR565 | 204.70 | 53.16 | 120.00 | 11.77 | 10.17 | 0.87 | 289.20 | 34.91 | 125.70 | 11.66 | 9.31 | 0.31 |
| UR547 | 110.50 | 8.38 | 80.90 | 7.50 | 6.88 | 0.16 | 147.70 | 7.90 | 80.90 | 7.50 | 6.49 | 0.07 |
| UR557 | 190.50 | 34.95 | 120.10 | 22.69 | 9.08 | 0.52 | 258.10 | 27.00 | 120.10 | 22.69 | 8.38 | 0.31 |
| UR567 | 130.60 | 43.59 | 92.80 | 9.21 | 8.68 | 0.68 | 186.00 | 26.19 | 92.80 | 9.21 | 8.22 | 0.24 |
| UR742 | 188.40 | 46.01 | 148.90 | 9.42 | 11.16 | 0.89 | 223.40 | 20.24 | 148.90 | 9.42 | 9.94 | 0.28 |
| UR752 | 232.10 | 33.86 | 160.20 | 11.69 | 17.04 | 0.79 | 330.80 | 70.44 | 160.20 | 11.69 | 16.15 | 1.04 |
| UR762 | 266.80 | 15.37 | 202.00 | 13.52 | 21.13 | 0.40 | 489.40 | 32.67 | 202.00 | 13.52 | 21.10 | 0.35 |
| UR132 | 120.50 | 54.90 | 24.60 | 2.17 | 11.48 | 1.48 | 186.00 | 28.42 | 24.60 | 2.17 | 10.93 | 0.41 |
| UR735 | 78.90 | 13.30 | 40.00 | 1.49 | 13.37 | 0.42 | 120.20 | 4.89 | 40.00 | 1.49 | 12.79 | 0.15 |
| UR142 | 115.00 | 16.57 | 94.70 | 13.98 | 23.59 | 0.65 | 179.30 | 30.44 | 94.70 | 13.98 | 22.73 | 0.62 |
| UR745 | 146.90 | 31.56 | 76.30 | 2.91 | 23.02 | 1.27 | 172.90 | 26.59 | 76.30 | 2.91 | 20.52 | 0.47 |
| UR755 | 213.60 | 28.51 | 252.60 | 24.49 | 30.11 | 1.20 | 290.40 | 53.88 | 252.60 | 24.49 | 27.37 | 1.11 |
| UR765 | 213.70 | 9.41 | 111.20 | 3.77 | 35.85 | 0.29 | 250.60 | 11.48 | 111.20 | 3.77 | 31.92 | 0.33 |
| UR737 | 107.60 | 17.78 | 49.40 | 7.18 | 20.64 | 0.81 | 137.70 | 8.43 | 49.40 | 7.18 | 19.15 | 0.22 |
| UR747 | 122.60 | 31.26 | 56.00 | 2.31 | 25.11 | 1.31 | 163.90 | 15.77 | 56.00 | 2.31 | 23.22 | 0.33 |
| UR757 | 155.90 | 28.57 | 165.10 | 16.54 | 30.83 | 1.30 | 256.00 | 22.63 | 165.10 | 16.54 | 29.70 | 0.50 |
| UR767 | 274.60 | 58.70 | 236.60 | 21.26 | 39.65 | 2.59 | 333.50 | 22.87 | 236.60 | 21.26 | 34.84 | 0.51 |
| UR152 | 208.30 | 39.61 | 155.40 | 11.81 | 39.98 | 1.88 | 268.30 | 16.64 | 155.40 | 11.81 | 36.43 | 0.38 |
| UR162 | 407.20 | 107.72 | 288.50 | 31.36 | 62.94 | 5.80 | 504.60 | 39.82 | 288.50 | 31.36 | 55.42 | 1.04 |
| UR135 | 49.80 | 9.35 | 33.00 | 0.82 | 34.40 | 0.69 | 132.50 | 16.64 | 33.00 | 0.82 | 36.79 | 0.61 |
| UR145 | 136.40 | 21.10 | 115.20 | 6.73 | 57.26 | 1.59 | 243.80 | 18.41 | 116.30 | 6.11 | 57.94 | 0.74 |
| UR155 | 174.70 | 31.59 | 195.80 | 21.34 | 83.70 | 2.69 | 242.80 | 25.37 | 195.80 | 21.34 | 85.42 | 4.06 |
| UR137 | 94.60 | 13.78 | 48.90 | 5.99 | 53.14 | 1.17 | 120.30 | 3.09 | 48.90 | 5.99 | 54.90 | 0.28 |
| UR165 | 155.10 | 26.17 | 181.40 | 18.46 | 93.71 | 2.34 | 236.00 | 37.40 | 181.40 | 18.46 | 98.32 | 1.94 |
| UR147 | 89.00 | 13.47 | 132.30 | 7.53 | 69.48 | 1.34 | 169.50 | 10.26 | 132.30 | 7.53 | 71.75 | 2.24 |
| UR157 | 154.00 | 21.55 | 139.70 | 12.14 | 83.01 | 1.90 | 256.30 | 13.62 | 139.70 | 12.14 | 82.30 | 0.99 |
| UR167 | 229.10 | 33.85 | 254.70 | 24.43 | 100.38 | 3.03 | 313.70 | 42.59 | 254.70 | 24.43 | 98.70 | 2.65 |

Table 2.5: Results of test instances for the Huff OD trip problem with 90%$card(E)$ facilities.

| | DC bound | | | | | | IA bound | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Network | iter | | MaxList | | time | | iter | | MaxList | | time | |
| | mean | std | mean | std | mean | std | mean | std | mean | std | mean | std |
| KROB150G | 88.50 | 3.66 | 62.70 | 4.30 | 0.22 | 0.00 | 137.80 | 3.71 | 62.70 | 4.30 | 0.21 | 0.01 |
| KROA150G | 92.70 | 19.97 | 41.70 | 6.00 | 0.22 | 0.02 | 167.20 | 10.12 | 41.70 | 6.00 | 0.22 | 0.01 |
| PR152G | 71.90 | 3.00 | 58.50 | 9.69 | 0.20 | 0.00 | 122.60 | 2.01 | 58.50 | 9.69 | 0.20 | 0.01 |
| RAT195G | 55.90 | 4.82 | 93.90 | 6.42 | 0.36 | 0.01 | 119.10 | 11.12 | 93.90 | 6.42 | 0.38 | 0.02 |
| KROB200G | 90.90 | 22.34 | 34.10 | 3.96 | 0.47 | 0.04 | 144.90 | 5.63 | 34.10 | 3.96 | 0.45 | 0.01 |
| KROA200G | 78.60 | 3.27 | 37.30 | 1.89 | 0.45 | 0.01 | 129.60 | 2.17 | 37.30 | 1.89 | 0.44 | 0.01 |
| TS225G | 65.80 | 2.49 | 57.10 | 3.28 | 0.46 | 0.02 | 120.20 | 0.42 | 57.10 | 3.28 | 0.46 | 0.01 |
| UR532 | 97.40 | 4.01 | 59.10 | 5.22 | 1.53 | 0.02 | 155.60 | 4.38 | 59.10 | 5.22 | 1.41 | 0.01 |
| UR542 | 161.40 | 8.77 | 150.30 | 6.55 | 3.11 | 0.04 | 240.00 | 14.00 | 150.30 | 6.55 | 2.83 | 0.05 |
| UR552 | 113.80 | 4.32 | 99.40 | 4.48 | 4.74 | 0.06 | 211.60 | 3.75 | 99.40 | 4.48 | 4.60 | 0.03 |
| UR562 | 204.00 | 10.14 | 195.20 | 8.47 | 7.69 | 0.09 | 356.40 | 13.94 | 199.10 | 12.56 | 7.29 | 0.09 |
| UR732 | 79.90 | 3.38 | 45.10 | 2.42 | 5.44 | 0.08 | 130.20 | 2.15 | 45.10 | 2.42 | 5.22 | 0.04 |
| UR535 | 81.60 | 4.53 | 44.40 | 4.62 | 5.19 | 0.08 | 137.80 | 9.73 | 44.40 | 4.62 | 4.98 | 0.09 |
| UR545 | 86.80 | 14.46 | 91.40 | 4.35 | 7.42 | 0.23 | 148.20 | 8.11 | 91.40 | 4.35 | 7.21 | 0.10 |
| UR555 | 156.00 | 29.30 | 133.00 | 23.67 | 10.30 | 0.48 | 223.20 | 18.90 | 133.00 | 23.67 | 9.49 | 0.17 |
| UR537 | 66.60 | 2.63 | 39.10 | 6.40 | 6.44 | 0.15 | 127.20 | 3.33 | 39.10 | 6.40 | 6.25 | 0.03 |
| UR565 | 153.50 | 9.62 | 128.70 | 2.98 | 11.96 | 0.19 | 281.70 | 28.41 | 136.30 | 5.58 | 11.43 | 0.24 |
| UR547 | 79.40 | 3.50 | 77.50 | 3.92 | 8.31 | 0.09 | 139.90 | 4.36 | 77.50 | 3.92 | 7.90 | 0.04 |
| UR557 | 132.10 | 10.12 | 122.90 | 13.92 | 10.40 | 0.18 | 270.80 | 18.26 | 122.90 | 13.92 | 10.26 | 0.17 |
| UR567 | 112.00 | 2.67 | 95.10 | 2.77 | 10.84 | 0.03 | 187.50 | 5.25 | 95.10 | 2.77 | 10.26 | 0.04 |
| UR742 | 162.20 | 13.79 | 162.30 | 9.13 | 13.17 | 0.29 | 246.00 | 16.07 | 162.30 | 9.13 | 12.39 | 0.21 |
| UR752 | 194.70 | 11.72 | 160.10 | 7.37 | 20.15 | 0.36 | 361.80 | 73.69 | 160.10 | 7.37 | 19.55 | 0.96 |
| UR762 | 303.80 | 77.63 | 191.50 | 15.41 | 27.38 | 2.02 | 445.70 | 38.44 | 191.50 | 15.41 | 24.96 | 0.57 |
| UR132 | 73.50 | 13.33 | 26.10 | 0.32 | 12.95 | 0.37 | 185.20 | 28.54 | 26.10 | 0.32 | 13.41 | 0.60 |
| UR735 | 72.80 | 3.55 | 43.60 | 1.51 | 16.70 | 0.15 | 123.50 | 4.17 | 43.60 | 1.51 | 16.46 | 0.81 |
| UR142 | 105.50 | 3.89 | 94.60 | 6.74 | 29.68 | 0.23 | 157.00 | 6.02 | 94.60 | 6.74 | 27.89 | 0.32 |
| UR745 | 127.60 | 44.94 | 81.50 | 2.59 | 28.40 | 1.88 | 188.00 | 4.69 | 81.50 | 2.59 | 25.80 | 0.18 |
| UR755 | 211.70 | 16.17 | 259.10 | 6.17 | 38.18 | 0.69 | 308.20 | 28.83 | 259.10 | 6.17 | 34.46 | 0.62 |
| UR765 | 135.80 | 4.73 | 111.90 | 3.96 | 41.83 | 0.35 | 258.40 | 15.54 | 111.90 | 3.96 | 40.24 | 0.58 |
| UR737 | 69.50 | 2.22 | 54.00 | 2.67 | 24.32 | 0.25 | 142.20 | 2.57 | 54.00 | 2.67 | 24.75 | 0.48 |
| UR747 | 86.50 | 8.45 | 56.90 | 2.08 | 30.30 | 0.31 | 162.50 | 14.03 | 56.90 | 2.08 | 29.43 | 1.03 |
| UR757 | 145.80 | 8.12 | 172.90 | 4.18 | 38.24 | 0.35 | 288.10 | 35.09 | 172.90 | 4.18 | 37.58 | 1.21 |
| UR767 | 195.00 | 13.23 | 241.50 | 10.74 | 46.02 | 0.75 | 327.40 | 22.74 | 241.50 | 10.74 | 44.82 | 1.23 |
| UR152 | 178.80 | 26.39 | 160.30 | 3.50 | 49.72 | 1.31 | 267.00 | 5.29 | 160.30 | 3.50 | 46.27 | 1.14 |
| UR162 | 328.10 | 22.26 | 324.70 | 14.35 | 74.00 | 1.49 | 550.80 | 38.55 | 324.70 | 14.35 | 68.54 | 1.19 |
| UR135 | 50.50 | 10.22 | 33.50 | 0.53 | 42.73 | 1.64 | 136.60 | 10.36 | 33.50 | 0.53 | 45.95 | 0.72 |
| UR145 | 133.10 | 5.65 | 127.50 | 3.69 | 67.30 | 0.42 | 266.10 | 24.08 | 127.50 | 3.69 | 71.57 | 1.10 |
| UR155 | 171.90 | 7.31 | 204.00 | 16.49 | 99.39 | 0.66 | 266.40 | 15.06 | 204.00 | 16.49 | 98.18 | 0.90 |
| UR137 | 72.80 | 3.22 | 51.00 | 4.08 | 61.46 | 0.28 | 124.60 | 4.20 | 51.00 | 4.08 | 62.38 | 0.52 |
| UR165 | 161.20 | 8.42 | 199.10 | 7.92 | 111.06 | 0.75 | 235.40 | 4.81 | 199.10 | 7.92 | 111.63 | 0.68 |
| UR147 | 91.40 | 7.89 | 135.60 | 4.48 | 82.76 | 0.73 | 171.10 | 3.21 | 135.60 | 4.48 | 86.18 | 0.51 |
| UR157 | 136.50 | 4.28 | 136.10 | 2.18 | 98.28 | 0.41 | 247.50 | 7.53 | 136.10 | 2.18 | 99.93 | 0.80 |
| UR167 | 216.80 | 7.25 | 262.90 | 9.26 | 118.59 | 0.77 | 327.70 | 19.31 | 262.90 | 9.26 | 116.10 | 1.34 |

The name of the network is reported in the first column of Tables 2.2-2.5. The remaining columns have the same meaning than those in Table 2.1. In Table 2.2, when dealing with $r = 10\%card(E)$ facilities in the Huff location problem, both bounding methods are comparable, but when dealing with a saturated market, Table 2.3, DC bound clearly outperforms IA bound. This is due to the simplicity of IA bound and because DC bound is sharper. One can see that, when using DC bound, the MaxList size and the number of iterations is much smaller than with the other bound, which means that DC bound is harder to calculate, but sharper.

For the Huff OD trip problem, as the number of computations of the bounds increases, both methods become comparable, as there is an equilibrium between the computational cost of bounds and how sharp they are.

One can see that, although the OD trip problem is, in terms of computing time, much harder to solve than the location problem, the number of iterations is relatively smaller, while the size of the branch and bound tree is comparable for most networks. Note that the relevant increase of the computing time for the OD trip problem with respect to the Huff location problem is due to the complexity of the objective function, which involves evaluating many more terms. In both cases all problems can be solved in reasonable time.

## 2.4 Conclusions

In this chapter we have addressed two Huff problems on a network, by considering that users choose the facility according to the distance from their location (Huff location problem, [18]) or according to the length of the shortest path from the origin to the destination visiting the facility (Huff OD trip problem), which is a new problem. Since these problems are shown to be multimodal, in order to obtain a global optimum, a branch and bound algorithm based on two different bounding procedures, IA and DC optimization, is proposed.

The computational experience reported shows that large networks can be successfully handled with both bounding procedures where the DC procedure seems to be more stable in both time and memory requirements.

# Chapter 3

# $p$-Facility Huff Location Problem on Networks

In this chapter [29] we extend to the $p$-facility case the Huff location problem on networks addressed in Chapter 2. This chapter contributes to location analysis by proposing two approaches for the initialization and division of subproblems in the branch and bound method, solving problems for $p \leq 4$ in reasonable computing time.

## 3.1 Introduction

In this chapter, we study the $p$-facility Huff location problem on networks (1.6)-(1.7) presented in Chapter 1. In this case, we will study the gravitational model, i.e., $\lambda_v = 2, \forall v \in V$. Hence, we aim at solving the following Mixed Integer Nonlinear Programming (MINLP) problem:

$$\max_{\substack{x_1 \in [0,l_{e_1}],\ldots,x_p \in [0,l_{e_p}] \\ e_1,\ldots,e_p \in E}} F(x_1,\ldots,x_p) \tag{3.1}$$

where $F$ is defined as

$$F(x_1,\ldots,x_p) = \sum_{v \in V} \omega_v \frac{1}{1 + \dfrac{\beta_v}{\displaystyle\sum_{j=1}^{p} \dfrac{1}{(d(v,x_j))^2}}}. \tag{3.2}$$

The MINLP problem (3.1) is formed by a combinatorial and a continuous part. First, we need to solve the combinatorial problem of choosing a set of $p$ edges to locate the facilities, and then solve a continuous location problem on the edges.

The remainder of this chapter is organized as follows. In Section 3.2, a branch and bound method with different initialization and branching rules is described. Section 3.3 is devoted to procedures for calculating lower and upper bounds. Computational results are reported in Section 3.4, where the $p$-facility Huff location problem is solved using the different branching and bounding rules for 12 real-life and 9 artificial networks. Finally, Section 3.5 contains a brief summary, final conclusions and some lines for future research.

## 3.2 The methodology

The natural way to solve the MINLP formulation of the $p$-facility Huff location problem is to use a branch and bound method. In these methods we differenciate two main phases: the initialization phase and the branch and bound phase. In the initialization phase the initial exploration tree is prepared. In the branch and bound phase, an element of the list is selected iteratively (until the termination rule is fulfilled) according to a selection criterion, and then is divided into new elements that are included into the list if they cannot be eliminated by their bounds. In this phase, division, bounding, selection, elimination and termination rules are required.

In this chapter we propose different approaches for the initialization phase, division and bounding rules. As selection, elimination and termination rules, we always apply the usual ones from the literature [18]: the element to be evaluated is selected as the one with the largest upper bound, elements whose upper bound are lower than the current lower bound are eliminated, and the optimization is terminated when the relative error between the largest upper bound and the current lower bound is less than a fixed tolerance. This section is aimed at describing two types of initialization and division rules. Bounding rules will be discussed in Section 3.3.

### 3.2.1 Total enumeration

The straightforward way of solving Problem (3.2) is to separate the combinatorial and the continuous part of the problem: we first fix a set of $p$ edges to locate the facilities, and then solve a continuous location problem on the edges. This means the branch and bound approach starts with a partition of the search space formed by the cartesian product of $p$-uples, thus a direct extension of the method described in Chapter 2. The $p$-uples are formed by every possible combination of $p$-edges, taking into account that several facilities can be located at the same edge, i.e., repetitions of the same edge are allowed in the elements of the partition. But obviously, permutations of the $p$-uples are not taken into account.

We denote by $\underline{s} = (s_1, \ldots, s_k)$ an element of the partition, where each component $s_i$ is a (sub)edge that has a multiplicity $m(s_i)$, i.e., the number of facilities located at $s_i$ is $m(s_i)$. Hence, $m(s_1) + \ldots + m(s_k) = p$. To avoid symmetric sets, for any element of the partition $\underline{s} = (s_1, \ldots, s_k)$, and any $s_i = [l, u] \subseteq [0, l_e], e \in E, s_i \in \underline{s}$, the cartesian product $\Pi_{j=1}^{m(s_i)} s_i$ is replaced by $\{l \leq x_1 \leq \ldots \leq x_{m(s_i)} \leq u\}$.

The subdivision of each element of the partition is done by splitting each (sub)edge by its midpoint, obtaining two new smaller segments for each (sub)edge, namely lower and upper segments. Then, the new elements of the partition are built by replacing each (sub)edge $s_i$ by either its lower or upper segment, $s_i^L, s_i^U$ respectively. In the case of (sub)edges with multiplicity greater than 1, the above-described method is used to avoid symmetric sets. For instance, Figure 3.1 depicts the subdivision process for $p = 2$, of the element $\underline{s} = (s_1)$, with $m(s_1) = 2$, identified with the coloured area of the big square. Then, the subdivision of $\underline{s}$ leads to three new elements, identified with the coloured area of the small squares.

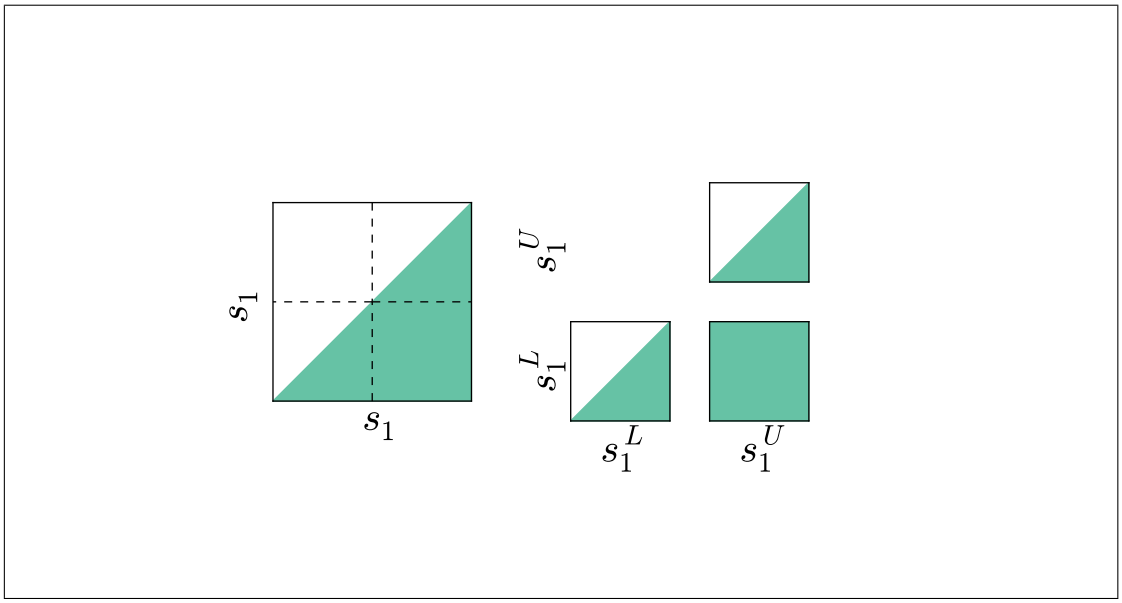


**Figure 3.1:** Subdivision process of $\underline{s} = (s_1)$ with $m(s_1) = 2$.

### 3.2.2 Superset

A more sophisticated data structure for location problems on networks has been proposed in [28], exploiting together the structure of the combinatorial and continuous part of a covering problem on networks.

In order to avoid the enumeration of every possible combination of $p$ edges, [28] proposes to construct clusters of (sub)edges, called hereafter *edgesets*, and define a subproblem of (3.1) over a collection of edgesets called a *superset*.

To be precise, an edgeset is a finite collection of (sub)edges of $E$; a superset $S$ is any uple of the form $(E_1, p_1; \ldots; E_k, p_k)$, where $E_1, \ldots, E_k$ are disjoint edgesets, and $p_j$ are strictly positive integer numbers with

$$\sum_{j=1}^{k} p_j = p,$$

indicating, for each $j = 1, \ldots, k$, that exactly $p_j$ facilities are to be located within the (sub)edges in $E_j$.

For this data structure, the subproblem to be solved at this stage on superset $S$ has the form

$$\max_{(x_1,\ldots,x_p)\in S} F(x_1, \ldots, x_p)$$

with $F$ defined as in (3.2), and $(x_1, \ldots, x_p) \in S$ understood as $x_1, \ldots, x_{p_1} \in E_1$; $x_{p_1+1}, \ldots, x_{p_1+p_2} \in E_2$; $\ldots$; $x_{p-p_k+1}, \ldots, x_p \in E_k$.

Supersets will be identified with nodes in the branch and bound tree. The root node of the branch and bound tree is the original superset $S_0 = (E, p)$. $E$ is first subdivided into a given partition $E^{(1)}, \ldots, E^{(p)}$ of $E$: we add to the branch and bound exploration tree the $\binom{2p-1}{p}$ supersets of the form $(E_1, p_1; \ldots; E_k, p_k)$, where $\{E_1, \ldots, E_k\} \subset \{E^{(1)}, \ldots, E^{(p)}\}$ and $p_1 + \ldots + p_k = p$.

First, we need to define how the edges of the network conforming $S_0$, are split

into the partition of $p$ edgesets $E^{(1)}, \ldots, E^{(p)}$. In the first step, $E$ is divided into 2 edgesets by a distance criterion, namely the diameter of the edgeset, defined as the maximum of the minimal distance between each pair of nodes. Then, the nodes giving the diameter are selected as centres of the two new (sub)edgesets. Each edge of the edgeset is assigned to the closest (sub)edgeset, where the distance from an edge to an (sub)edgeset is measured as the distance from the edge to the (sub)edgeset centre. Then, we will repeat the process until obtaining $p$ edgesets, selecting the largest edgeset to be subdivided at each step, where the size of the edgeset is understood as the sum of the (sub)edgelengths.

The subdivision of a superset $S = (E_1, p_1; \ldots; E_k, p_k)$ during the branch and bound is done by partitioning the largest edgeset $E_i$. If $E_i$ contains only one (sub)edge, the subdivision is done by bisecting the (sub)edge at its midpoint, otherwise $E_i$ is partitioned to edgesets $E_{i_1}, E_{i_2}$ by its diameter as done in the initial subdivision of $S_0$. Thus, the following supersets substitute $S$:

$$S_j = (E_1, p_1; \ldots; E_{i-1}, p_{i-1}; E_{i_j}, p_i; E_{i+1}, p_{i+1}; \ldots; E_k, p_k), j = 1, 2$$

and additionally if $1 < p_i$, for $j = 1, \ldots, p_i - 1$

$$S_{2+j} = (E_1, p_1; \ldots; E_{i-1}, p_{i-1}; E_{i_1}, j; E_{i_2}, p_i - j; E_{i+1}, p_{i+1}; \ldots; E_k, p_k).$$

This means, that in each step $p_i + 1$ new supersets are created.

## 3.3 Lower and upper bounds

A branch and bound algorithm requires the calculation of tight upper and lower bounds. In this section we present different bounding approaches for the branch and bound used to solve (3.1). We propose two upper bounds and two lower bounds: IA bound and DC bound as upper bounds, Huff discrete bound (HuffDisc) and midpoint bound (MidPoint) as lower bounds. Note that when a superset contains edgesets with $card(E_j) = 1 \; \forall j$, it corresponds to a $p$-uple of (sub)edges. Each $p$-uple of (sub)edges has its unique superset correspondance. Thus, the following bounds are valid for $p$-uples of edges as well, i.e., for the enumeration approach in Section 3.2.1.

### 3.3.1 Upper bounds

The IA bound considers only endpoints of (sub)edges as possible location of facilities. This bound is the extension to the $p$-facility case of the IA bound addressed in Section 2.2.3. For a superset $S = (E_1, p_1; \ldots; E_k, p_k)$ we obtain the IA bound by replacing in (3.2) $d(v, x_j)$ by the distance from $v$ to the closest vertex of the (sub)edges that belong to $E_j$, i.e., by

$$d(v, E_j) = \min_{e=[u_1,u_2], e \in E_j} \{d(v, u_1), d(v, u_2)\}.$$

For any $x \in E_j$ it holds that $d(v, E_j) \leq d(v, x) \; \forall j = 1, \ldots, p$. Hence, the following is a valid upper bound for (3.2):

$$UB^{IA}(S) := \sum_{v \in V} \omega_v \frac{1}{1 + \frac{\beta_v}{\sum_{j=1}^{p} \frac{p_j}{(d(v, E_j))^2}}}.$$

The second upper bound approach is based on the DC bound (2.8) of the single facility Huff location problem on networks,

$$\max_{x\in[0,l_e],\, e\in E} F_{single}(x),$$

with $F_{single}(x) = \sum_{v\in V} \omega_v \frac{1}{1+\beta_v(d(v,x))^2}$, a particular case of (3.2) for $p = 1$, the problem addressed in Section 1.2.1. First, for a given edge $e$ of the network, $F_{single}(x)$ is expressed as a difference of convex functions, $F_{single}(x) = \sum_{v\in V}(F_v^+(x) - F_v^-(x))$, namely, its DC decomposition. Then, an upper bound $UB^{DC}_{single}(e)$ for any edge $e \in E$ with $u_1,\ u_2$ being endpoints of $e$ is defined as

$$UB^{DC}_{single}(e) = \max_{e=[u_1,u_2],e\in E}\{U(u_1), U(u_2)\}$$

with

$$U(x) = \sum_{v\in V}(F_v^+(x) - F_v^-(x_0) - \xi_v(x - x_0))$$

for $\xi_v \in \partial F_v^-(x_0)$ where $\partial F_v^-(x_0)$ denotes the set of subgradients of $F_v^-$ at $x_0$ [150]. Therefore, it holds that

$$UB^{DC}_{single}(e) \geq F_{single}(x), \forall x \in [0, l_e],\ e \in E. \tag{3.3}$$

A DC bound over an edgeset $E_j$ is defined as the maximum DC bound of the edges from $E_j$, i.e.,

$$UB^{DC}(E_j) := \max_{e\in E_j} UB^{DC}_{single}(e) \geq F_{single}(x), \forall x \in [0, l_e],\ \forall e \in E_j.$$

Given a superset $S = (E_1, p_1; \ldots; E_k, p_k)$, a DC bound of (3.2) is calculated as

$$UB^{DC}(S) := \sum_{j=1}^{k} p_j \cdot UB^{DC}(E_j). \tag{3.4}$$

This DC bound is a valid upper bound since it holds that for any feasible point $(x_1, \ldots, x_p)$

$$\sum_{j=1}^{p} F_{single}(x_j) = \sum_{j=1}^{p}\sum_{v\in V} \omega_v \frac{1}{1+\beta_v(d(v,x_j))^2}. \tag{3.5}$$

Since $\frac{1}{(d(v,x_j))^2} \leq \sum_{j=1}^{p} \frac{1}{(d(v,x_j))^2}$, we have:

$$(3.5) = \sum_{j=1}^{p}\sum_{v\in V} \omega_v \frac{1/(d(v,x_j))^2}{1/(d(v,x_j))^2 + \beta_v} \geq \sum_{j=1}^{p}\sum_{v\in V} \omega_v \frac{1/(d(v,x_j))^2}{\sum_{i=1}^{p} 1/(d(v,x_i))^2 + \beta_v} =$$

$$= \sum_{v\in V} \omega_v \frac{\sum_{j=1}^{p} 1/(d(v,x_j))^2}{\sum_{i=1}^{p} 1/(d(v,x_i))^2 + \beta_v} = \sum_{v\in V} \omega_v \frac{1}{1 + \frac{\beta_v}{\sum_{j=1}^{p} \frac{1}{(d(v,x_j))^2}}} = F(x_1, \ldots, x_p).$$

### 3.3.2 Lower bounds

Given a superset $S$, both of our lower bounding approaches are based on the calculation of the objective function at a feasible solution $(\tilde{x}_1, \ldots, \tilde{x}_p) \in S$. Then, a valid lower bound is given by

$$LB(S) := F(\tilde{x}_1, \ldots, \tilde{x}_p) \leq \max_{(x_1,\ldots,x_p)\in S} F(x_1, \ldots, x_p).$$

Let us now focus on possible feasible solutions. The first lower bound, namely Huff discrete bound ($LB^{HuffDisc}$), is a greedy procedure based on solving iteratively $p$ times the single facility Huff location problem at the vertices of the edges of the superset. For a given superset $S = (E_1, p_1; \ldots; E_k, p_k)$, let $\tilde{x}_1$ be the optimal solution of a single facility Huff location problem on the vertices of the (sub)edges of $E_1$. In the next step, we will consider that a facility is already located at $\tilde{x}_1$, and will locate $\tilde{x}_2$ solving the single facility Huff location problem at the vertices of the edges of the corresponding edgeset. In the last step, we will choose location $\tilde{x}_p$ as the optimal solution of the single facility Huff location problem on the vertices of the edges of $E_k$, considering that $p-1$ facilities are already located at $\tilde{x}_1, \ldots, \tilde{x}_{p-1}$. Since only vertices are considered as candidates, each step of the greedy procedure is executed by complete enumeration of the candidate points.

The second lower bound, namely the midpoint bound $LB^{MidPoint}$, is calculated by randomly choosing an edge from an edgeset $E_j$, and locating $p_j$ facilities at its midpoint $\forall j \leq k$.

## 3.4 Computational results

The approaches described in Sections 3.2 and 3.3 were implemented in Fortran and executed on an Intel Core i7 computer with 16.00 Gb of RAM memory. The solutions were found within an accuracy of $10^{-3}$.

We tested the approaches on a battery of 21 networks, whose characteristics are shown in Table 1.1. The number $r$ of existing facilities is set as $r = 10\%$ of the number of edges of the network, $card(E)$.

Tables 3.1-3.8 show a comparison between the two branching rules, total enumeration (Section 3.2.1) and superset (Section 3.2.2), and the different bounding approaches: IA bound, IA bound with DC bound (IA+DC), midpoint evaluation (MidPoint) and Huff discrete bound (HuffDisc). The bounding approach IA+DC consists of using IA bound in the initialization phase of the algorithm, and then, using DC bound during the branch and bound phase. Results for DC bound as the only upper bound are not reported because they were systematically outperformed by the results in Tables 3.1-3.8. The results for the combinations of upper and lower bounds are shown in four blocks of columns. The first column shows the maximum size of the branch and bound tree (MaxList) during execution. The sign "×" in the MaxList column means that the size limit ($10^8$) was exceeded so the method was stopped. The second column reports the CPU time in seconds. Time limit is set to 6 hours (21600 seconds) and when it is exceeded, it is denoted with "×". In such case, the third column shows the gap in % achieved by the approach, measured as $\frac{Upper\ bound - Lower\ bound}{Lower\ bound} 100\%$.

We start with the analysis of $p = 2$, in Tables 3.1 and 3.2. All strategies are able to solve the problem on the 21 networks in less than an average time of 2 seconds. For both approaches, the best upper and lower bound choice is IA+DC and MidPoint respectively, with superset being the fastest approach and enumeration achieving the best MaxList size.

For $p = 3$, using supersets we achieve the best computing time, while using enumeration we achieve the best Maxlist result. For the superset approach, the choice of lower bound affects the behaviour of computing times, with an average improvement of about 200 seconds for MidPoint bound compared to HuffDisc bound. In the case of the enumeration approach, the choice of upper bound affects the MaxList size. Using IA+DC bound halves the MaxList size compared to using only IA bound. For both approaches, the best upper and lower bound choice is IA+DC and MidPoint respectively.

Tables 3.5 and 3.6 report results for $p = 4$. Using enumeration with HuffDisc bound solves 15 networks regardless of the upper bound used, while with MidPoint it solves 17 with IA bound and 18 if using IA+DC bound. Supersets with HuffDisc bound solves 15 networks regardless of the upper bound used, while with MidPoint we achieve successful results for 20 out of 21 networks. The $RAT195G$ network is not solved by any of the approaches. Using enumeration, its gap reduces from 32.24% to 17.67% if IA+DC bound is used. Using supersets with IA+DC bound its gap reduces from 15.94% to 8.24% when HuffDisc bound is used and from 12.57% to 9.93% for MidPoint bound. In terms of time, the best choice is to use MidPoint as lower bound, while the choice of upper bound does not make big difference for the superset approach, but for the enumeration approach, IA bound is the best choice. If we compare the results only for $art1 - art9$ networks, which are very small, see Table 1.1, the enumeration approach becomes the best in terms of all criteria. However, when the size of the network increases, the superset approach outperforms the enumeration one. For the enumeration approach, the difference between the bounding approaches in terms of MaxList size is hardly noticeable, being the best choice IA+DC as upper bound and MidPoint as lower bound. For the superset approach, slightly better MaxList sizes are achieved when using HuffDisc bound.

Finally, we analyze results for $p = 5$, Tables 3.7 and 3.8. Using enumeration we are able to solve the problem on 8 networks while with supersets on 7 networks. Using enumeration, there is a big outperformance in terms of gap achieved by IA bound over IA+DC bound. In terms of computing time, lower bound makes a small difference, MidPoint bound being the fastest one. In terms of MaxList size, there is no big difference between the bounding approaches. Using supersets, the choice of lower bound makes a big difference in terms of MaxList size and time. There is an average improvement of more than 3 hours of MidPoint bound over HuffDisc bound. On the contrary, in terms of MaxList size, HuffDisc bound outperforms MidPoint bound, the MaxList size of the latter being about the double of HuffDisc MaxList size. These big differences in the behaviour when changing lower bound are due to HuffDisc bound being computationally expensive, which makes the approach stop due to time limit, and MidPoint bound less efficient, which explodes the size of the MaxList tree. Better gaps are achieved using IA+DC bound. For the enumeration approach, we achieve better results when using IA bound with HuffDisc bound. For the superset approach, the best lower bound choice is using MidPoint bound while the choice of upper bound is irrelevant.

In summary, we can say that both approaches are comparable for $p = 2, 3$ while for $p = 4, 5$ the superset approach outperforms the enumeration approach. When faced with the choice of the best upper bound, using IA+DC bound as upper bound is the best choice for both approaches and all values of $p$, except for $p = 5$ for the enumeration approach. In terms of lower bound, we observe that using MidPoint as lower bound is, in general, the best choice, except for $p = 5$ for the enumeration approach.

Table 3.1: Maximum branch and bound tree size and running times for $p = 2$ for the enumeration approach.

| | enumeration | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Upper bound | IA | | | | IA+DC | | | |
| Lower bound | HuffDisc | | MidPoint | | HuffDisc | | MidPoint | |
| Network | MaxList | time | MaxList | time | MaxList | time | MaxList | time |
| art1 | 535 | 0.00 | 642 | 0.00 | 682 | 0.00 | 798 | 0.02 |
| art2 | 320 | 0.02 | 346 | 0.00 | 664 | 0.00 | 664 | 0.02 |
| art3 | 378 | 0.02 | 382 | 0.00 | 478 | 0.02 | 478 | 0.00 |
| art4 | 377 | 0.02 | 390 | 0.00 | 861 | 0.03 | 861 | 0.00 |
| art5 | 1256 | 0.02 | 1346 | 0.00 | 1301 | 0.03 | 1302 | 0.02 |
| art6 | 1351 | 0.03 | 1362 | 0.03 | 1145 | 0.03 | 1145 | 0.02 |
| art7 | 1594 | 0.03 | 1612 | 0.03 | 931 | 0.02 | 934 | 0.00 |
| art8 | 1417 | 0.03 | 1512 | 0.03 | 922 | 0.03 | 938 | 0.03 |
| art9 | 1497 | 0.06 | 1538 | 0.03 | 1636 | 0.08 | 1640 | 0.03 |
| pmed1 | 1094 | 0.25 | 1130 | 0.16 | 425 | 0.19 | 429 | 0.17 |
| pmed2 | 2747 | 0.22 | 2796 | 0.17 | 805 | 0.16 | 809 | 0.17 |
| pmed3 | 1184 | 0.22 | 1184 | 0.16 | 334 | 0.17 | 334 | 0.17 |
| pmed4 | 675 | 0.20 | 677 | 0.16 | 175 | 0.17 | 175 | 0.17 |
| pmed5 | 3323 | 0.31 | 3365 | 0.17 | 1164 | 0.20 | 1164 | 0.17 |
| KROB150G | 4143 | 1.53 | 4561 | 0.61 | 381 | 0.76 | 451 | 0.58 |
| KROA150G | 4897 | 1.70 | 5003 | 0.66 | 1039 | 0.95 | 1039 | 0.61 |
| PR152G | 712 | 0.95 | 1014 | 0.56 | 494 | 0.87 | 586 | 0.61 |
| RAT195G | 17877 | 7.13 | 17916 | 1.51 | 2149 | 1.84 | 2149 | 1.09 |
| KROB200G | 3489 | 2.76 | 3792 | 1.31 | 1237 | 2.22 | 1315 | 1.40 |
| KROA200G | 2675 | 2.36 | 2828 | 1.28 | 546 | 1.93 | 546 | 1.40 |
| TS225G | 3822 | 1.61 | 3843 | 0.90 | 1325 | 1.20 | 1325 | 0.97 |
| Average | 2636 | 0.93 | 2725 | 0.37 | 890 | 0.52 | 908 | 0.36 |

Table 3.2: Maximum branch and bound tree size and running times for $p = 2$ for the superset approach.

| | superset | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Upper bound | IA | | | | IA+DC | | | |
| Lower bound | HuffDisc | | MidPoint | | HuffDisc | | MidPoint | |
| Network | MaxList | time | MaxList | time | MaxList | time | MaxList | time |
| art1 | 298 | 0.03 | 324 | 0.02 | 298 | 0.02 | 324 | 0.02 |
| art2 | 920 | 0.03 | 1087 | 0.03 | 920 | 0.03 | 1087 | 0.02 |
| art3 | 569 | 0.03 | 1602 | 0.02 | 466 | 0.03 | 1336 | 0.00 |
| art4 | 580 | 0.03 | 835 | 0.02 | 580 | 0.03 | 835 | 0.00 |
| art5 | 1310 | 0.09 | 2106 | 0.03 | 1185 | 0.08 | 1895 | 0.02 |
| art6 | 1299 | 0.09 | 1884 | 0.03 | 1299 | 0.09 | 1884 | 0.02 |
| art7 | 1641 | 0.11 | 1972 | 0.03 | 1605 | 0.12 | 1969 | 0.03 |
| art8 | 2370 | 0.22 | 2432 | 0.05 | 2370 | 0.23 | 2426 | 0.06 |
| art9 | 3367 | 0.31 | 4709 | 0.06 | 2425 | 0.30 | 4184 | 0.06 |
| pmed1 | 3423 | 1.22 | 3738 | 0.20 | 3260 | 1.20 | 3380 | 0.20 |
| pmed2 | 2371 | 0.80 | 3703 | 0.14 | 2137 | 0.80 | 3706 | 0.14 |
| pmed3 | 1950 | 0.64 | 2286 | 0.11 | 1854 | 0.61 | 2247 | 0.11 |
| pmed4 | 5883 | 1.54 | 8262 | 0.27 | 5682 | 1.51 | 8273 | 0.27 |
| pmed5 | 3466 | 1.08 | 4045 | 0.17 | 2050 | 1.08 | 3046 | 0.19 |
| KROB150G | 5373 | 3.07 | 6482 | 0.37 | 3006 | 2.64 | 5055 | 0.30 |
| KROA150G | 5070 | 3.06 | 6113 | 0.37 | 4432 | 2.54 | 4970 | 0.33 |
| PR152G | 465 | 0.67 | 638 | 0.08 | 381 | 0.39 | 533 | 0.05 |
| RAT195G | 21719 | 13.73 | 25581 | 1.61 | 11951 | 13.56 | 15576 | 1.51 |
| KROB200G | 4529 | 5.02 | 5499 | 0.51 | 2477 | 2.54 | 3018 | 0.28 |
| KROA200G | 3548 | 4.49 | 4687 | 0.47 | 1624 | 4.07 | 2134 | 0.39 |
| TS225G | 4336 | 3.81 | 5733 | 0.42 | 3887 | 3.78 | 5135 | 0.42 |
| Average | 3547 | 1.91 | 4463 | 0.24 | 2566 | 1.70 | 3477 | 0.21 |

## 3.5 Conclusions

In this chapter we have addressed the $p$-facility case of the Huff location problem on networks, Section 1.2.1. We propose two branch and bound based approaches and show results for $p \leq 5$. Computational results show that both division approaches are able

Table 3.3: Maximum branch and bound tree size and running times for $p = 3$ for the enumeration approach.

| | enumeration | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Upper bound | IA | | | | IA+DC | | | |
| Lower bound | HuffDisc | | MidPoint | | HuffDisc | | MidPoint | |
| Network | MaxList | time | MaxList | time | MaxList | time | MaxList | time |
| art1 | 8267 | 0.11 | 9910 | 0.09 | 9866 | 0.39 | 15477 | 0.23 |
| art2 | 6188 | 0.08 | 7800 | 0.08 | 13756 | 0.55 | 13922 | 0.23 |
| art3 | 10584 | 0.36 | 11793 | 0.17 | 18592 | 0.62 | 20144 | 0.31 |
| art4 | 7916 | 0.41 | 9198 | 0.23 | 28851 | 1.90 | 29185 | 0.73 |
| art5 | 40940 | 0.90 | 41153 | 0.51 | 59592 | 3.56 | 59663 | 1.29 |
| art6 | 41664 | 1.47 | 43088 | 0.80 | 71196 | 3.67 | 71751 | 1.51 |
| art7 | 45641 | 1.25 | 45862 | 0.84 | 57556 | 2.56 | 57750 | 1.14 |
| art8 | 58719 | 1.54 | 59989 | 0.97 | 98178 | 3.39 | 98979 | 1.75 |
| art9 | 52667 | 3.84 | 53788 | 1.73 | 135925 | 13.17 | 135925 | 4.04 |
| pmed1 | 69966 | 18.99 | 70310 | 13.28 | 58693 | 22.95 | 58923 | 16.94 |
| pmed2 | 156224 | 16.99 | 156992 | 13.10 | 93928 | 17.33 | 93952 | 15.10 |
| pmed3 | 55064 | 19.06 | 55064 | 13.24 | 36455 | 19.83 | 36455 | 16.58 |
| pmed4 | 46950 | 17.11 | 47374 | 12.76 | 22048 | 18.19 | 22048 | 15.91 |
| pmed5 | 201832 | 34.30 | 205948 | 16.05 | 116076 | 24.74 | 116218 | 17.25 |
| KROB150G | 336291 | 268.09 | 353294 | 86.27 | 48164 | 112.79 | 48164 | 84.57 |
| KROA150G | 379461 | 324.26 | 385955 | 90.07 | 137050 | 194.86 | 137050 | 93.49 |
| PR152G | 50488 | 131.56 | 51722 | 69.84 | 66332 | 159.98 | 66332 | 90.03 |
| RAT195G | 1822569 | 1306.57 | 1823360 | 231.01 | 249618 | 280.55 | 249618 | 167.53 |
| KROB200G | 390512 | 597.89 | 399780 | 217.18 | 223063 | 513.26 | 223063 | 262.47 |
| KROA200G | 207484 | 324.93 | 208160 | 203.71 | 58563 | 290.55 | 58563 | 254.16 |
| TS225G | 270334 | 216.11 | 270894 | 115.80 | 123379 | 191.80 | 124312 | 141.23 |
| Average | 202846 | 156.47 | 205306 | 51.80 | 82232 | 89.36 | 82738 | 56.50 |

Table 3.4: Maximum branch and bound tree size and running times for $p = 3$ for the superset approach.

| | superset | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Upper bound | IA | | | | IA+DC | | | |
| Lower bound | HuffDisc | | MidPoint | | HuffDisc | | MidPoint | |
| Network | MaxList | time | MaxList | time | MaxList | time | MaxList | time |
| art1 | 3880 | 0.39 | 5397 | 0.09 | 3880 | 0.39 | 5397 | 0.09 |
| art2 | 16975 | 1.05 | 27467 | 0.27 | 16975 | 1.06 | 27467 | 0.28 |
| art3 | 24193 | 2.17 | 53640 | 0.53 | 23876 | 2.17 | 53525 | 0.51 |
| art4 | 9803 | 0.92 | 18388 | 0.22 | 9803 | 0.94 | 18388 | 0.20 |
| art5 | 41377 | 4.07 | 66646 | 0.94 | 41148 | 4.07 | 66605 | 0.95 |
| art6 | 58757 | 6.12 | 75996 | 1.19 | 58757 | 6.16 | 75996 | 1.23 |
| art7 | 67579 | 7.69 | 73556 | 1.61 | 67579 | 7.74 | 73556 | 1.65 |
| art8 | 88324 | 11.62 | 117422 | 2.18 | 88324 | 11.65 | 117422 | 2.26 |
| art9 | 139697 | 18.83 | 227709 | 3.29 | 139596 | 18.84 | 227624 | 3.31 |
| pmed1 | 271844 | 169.71 | 288840 | 21.76 | 271947 | 167.11 | 288840 | 21.90 |
| pmed2 | 222849 | 111.34 | 378409 | 14.59 | 222851 | 110.09 | 378514 | 14.63 |
| pmed3 | 135698 | 64.88 | 187136 | 8.22 | 135370 | 63.82 | 186754 | 8.30 |
| pmed4 | 418453 | 167.26 | 631093 | 21.82 | 409319 | 165.45 | 621040 | 21.81 |
| pmed5 | 223885 | 106.75 | 264197 | 13.65 | 219336 | 105.53 | 260945 | 13.65 |
| KROB150G | 494950 | 394.23 | 593159 | 39.98 | 438955 | 368.16 | 522850 | 36.40 |
| KROA150G | 478892 | 410.13 | 578185 | 42.04 | 470628 | 408.25 | 568164 | 41.04 |
| PR152G | 58848 | 81.81 | 68886 | 8.03 | 61306 | 78.16 | 67491 | 7.88 |
| RAT195G | 2190021 | 2026.48 | 2768940 | 189.42 | 1242889 | 1877.61 | 1687197 | 166.50 |
| KROB200G | 436217 | 647.47 | 549967 | 54.12 | 340731 | 479.09 | 429621 | 38.58 |
| KROA200G | 262735 | 435.37 | 341576 | 35.07 | 214576 | 397.79 | 290266 | 30.37 |
| TS225G | 315713 | 379.91 | 399584 | 36.02 | 306847 | 376.56 | 388696 | 34.24 |
| Average | 283842 | 240.39 | 367438 | 23.57 | 227843 | 221.46 | 302684 | 21.23 |

to solve problems of rather realistic size up to $p = 4$ facilities while for $p = 5$ only small problems are solved. For small values of $p$, both approaches are comparable, and when the number of facilities increases, the superset approach outperforms the enumeration approach. We conclude with three promising extensions.

As shown in Section 3.4, for high values of $p$ and for both approaches, some problems remain unsolved because the MaxList size limit is reached. It could be interesting to design a heuristic approach able to reduce the number of elements of the partition, and exploit the benefits of the branch and bound tree evolution.

As second extension, both approaches could be applied to different $p$-facility location problems on networks, such as the $p$-median problem with continuous demand on a network [27].

Finally, parallelization techniques deserve further study. Parallelizing the approach can solve the problem of reaching the MaxList size limit and may reduce the computational cost linearly, which will definitely lead to solving the problem for higher values of $p$.

Table 3.5: Maximum branch and bound tree size, running times and achieved gaps for $p = 4$ for the enumeration approach.

| | enumeration | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Upper bound | IA | | | | | | IA+DC | | | | | |
| Lower bound | HuffDisc | | | MidPoint | | | HuffDisc | | | MidPoint | | |
| Network | MaxList | time | gap | MaxList | time | gap | MaxList | time | gap | MaxList | time | gap |
| art1 | 87598 | 1.68 | – | 117441 | 1.22 | – | 101277 | 8.63 | – | 198557 | 4.29 | – |
| art2 | 74095 | 1.23 | – | 101784 | 1.58 | – | 163003 | 15.16 | – | 167512 | 5.40 | – |
| art3 | 177508 | 8.55 | – | 293743 | 4.43 | – | 307426 | 29.08 | – | 490398 | 12.36 | – |
| art4 | 159413 | 8.46 | – | 185915 | 5.24 | – | 454552 | 68.27 | – | 459312 | 21.04 | – |
| art5 | 873673 | 39.11 | – | 881711 | 16.61 | – | 1150102 | 199.56 | – | 1177881 | 59.86 | – |
| art6 | 975158 | 51.75 | – | 984445 | 23.57 | – | 2080085 | 327.80 | – | 2129849 | 84.52 | – |
| art7 | 1028331 | 58.73 | – | 1036206 | 24.90 | – | 1347049 | 268.88 | – | 1406366 | 71.64 | – |
| art8 | 1616556 | 92.37 | – | 1651432 | 40.20 | – | 3423277 | 659.45 | – | 3426785 | 158.54 | – |
| art9 | 1100215 | 141.68 | – | 1111150 | 63.32 | – | 5890324 | 1527.42 | – | 5890324 | 322.58 | – |
| pmed1 | 3920397 | 1314.92 | – | 3927269 | 851.34 | – | 6667549 | 3020.66 | – | 6667781 | 1292.31 | – |
| pmed2 | 7644966 | 959.14 | – | 7646396 | 763.20 | – | 9792019 | 1369.41 | – | 9792181 | 1027.34 | – |
| pmed3 | 2342677 | 1217.93 | – | 2344253 | 832.70 | – | 4280821 | 2344.15 | – | 4280821 | 1212.02 | – |
| pmed4 | 2652444 | 1252.69 | – | 2685214 | 817.60 | – | 2767490 | 1840.00 | – | 2767490 | 1100.93 | – |
| pmed5 | 8830603 | 2538.98 | – | 8979523 | 1041.57 | – | 9393530 | 3119.91 | – | 9398491 | 1350.89 | – |
| KROB150G | 27979117 | × | 0.04 | 29666987 | 9798.09 | – | 8609286 | 20606.66 | – | 8609286 | 9043.41 | – |
| KROA150G | × | 5256.88 | 34.17 | × | 5348.99 | 34.17 | 24167062 | × | 0.03 | 24203649 | 11270.48 | – |
| PR152G | 2993638 | 13190.67 | – | 3099354 | 6683.47 | – | 8203867 | × | 0.01 | 8203867 | 9327.78 | – |
| RAT195G | × | 2956.78 | 32.24 | × | 3000.29 | 32.24 | × | 12331.88 | 17.67 | × | 12191.70 | 17.67 |
| KROB200G | × | 21119.03 | 26.10 | × | 20559.01 | 26.10 | × | 20059.76 | 27.89 | 24150000 | × | 0.27 |
| KROA200G | 22110803 | × | 0.18 | 22149983 | × | 0.23 | 8670000 | × | 0.26 | 8670000 | × | 0.26 |
| TS225G | 22200741 | × | 0.02 | 22243742 | 12332.64 | – | 18379844 | × | 0.04 | 18379844 | 15321.28 | – |
| Average | 19369901 | 5476.69 | 4.42 | 19481264 | 3990.95 | 4.42 | 15040407 | 7342.70 | 2.19 | 11450971 | 5098.97 | 0.87 |

Table 3.6: Maximum branch and bound tree size, running times and achieved gaps for $p = 4$ for the superset approach.

| | superset | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Upper bound | IA | | | | | | IA+DC | | | | | |
| Lower bound | HuffDisc | | | MidPoint | | | HuffDisc | | | MidPoint | | |
| Network | MaxList | time | gap | MaxList | time | gap | MaxList | time | gap | MaxList | time | gap |
| art1 | 157141 | 19.22 | – | 152898 | 4.23 | – | 157141 | 20.22 | – | 152898 | 4.35 | – |
| art2 | 181037 | 17.05 | – | 326290 | 3.81 | – | 181037 | 17.89 | – | 326290 | 3.87 | – |
| art3 | 1093070 | 156.98 | – | 2659877 | 34.73 | – | 1093070 | 165.64 | – | 2659878 | 36.05 | – |
| art4 | 101627 | 13.63 | – | 248993 | 2.65 | – | 101627 | 14.23 | – | 248998 | 2.70 | – |
| art5 | 784382 | 109.43 | – | 1730167 | 21.61 | – | 784350 | 114.36 | – | 1730170 | 21.96 | – |
| art6 | 1346499 | 196.92 | – | 1845530 | 33.52 | – | 1346499 | 202.04 | – | 1845530 | 34.68 | – |
| art7 | 1415772 | 222.43 | – | 1581166 | 39.30 | – | 1415772 | 225.20 | – | 1581166 | 40.17 | – |
| art8 | 3459070 | 702.52 | – | 5218221 | 116.56 | – | 3459070 | 708.77 | – | 5218221 | 120.68 | – |
| art9 | 4195351 | 802.08 | – | 7068510 | 125.58 | – | 4195351 | 809.40 | – | 7068510 | 125.33 | – |
| pmed1 | 10765649 | 7786.70 | – | 12980089 | 859.97 | – | 10765649 | 7677.53 | – | 12980089 | 857.10 | – |
| pmed2 | 11143314 | 7410.62 | – | 19962438 | 834.42 | – | 10984446 | 7273.61 | – | 19944050 | 819.15 | – |
| pmed3 | 6524060 | 4148.35 | – | 9334140 | 464.82 | – | 6522592 | 4116.63 | – | 9332340 | 458.92 | – |
| pmed4 | 16174587 | 9091.02 | – | 24213155 | 1031.35 | – | 16174587 | 9017.36 | – | 24213155 | 1023.01 | – |
| pmed5 | 8687562 | 5509.46 | – | 10884353 | 604.88 | – | 8687776 | 5464.92 | – | 10884353 | 599.86 | – |
| KROB150G | 35348181 | × | 3.26 | 42047737 | 3354.30 | – | 35289997 | × | 3.24 | 41974612 | 3347.03 | – |
| KROA150G | 31579343 | × | 2.44 | 38836554 | 3146.35 | – | 31579345 | × | 2.43 | 38836554 | 3172.58 | – |
| PR152G | 2879248 | 5482.97 | – | 3759971 | 460.70 | – | 2796219 | 5290.21 | – | 3657358 | 450.09 | – |
| RAT195G | 39163633 | × | 15.94 | × | 3893.07 | 12.57 | 40806156 | × | 8.27 | × | 3779.84 | 9.93 |
| KROB200G | 27205462 | × | 4.64 | 38996783 | 4209.31 | – | 25044941 | × | 3.44 | 36820724 | 3896.58 | – |
| KROA200G | 18774302 | × | 3.08 | 24445462 | 2700.81 | – | 16949758 | × | 2.43 | 22853487 | 2503.93 | – |
| TS225G | 22020591 | × | 2.73 | 28041441 | 2878.48 | – | 22020758 | × | 2.73 | 28041441 | 2869.51 | – |
| Average | 11571423 | 8155.68 | 1.53 | 17825418 | 1181.93 | 0.60 | 11445531 | 8129.43 | 1.07 | 17636658 | 1150.83 | 0.47 |

Table 3.7: Maximum branch and bound tree size, running times and achieved gaps for $p = 5$ for the enumeration approach.

| | enumeration | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Upper bound | IA | | | | | | IA+DC | | | | | |
| Lower bound | HuffDisc | | | MidPoint | | | HuffDisc | | | MidPoint | | |
| Network | MaxList | time | gap | MaxList | time | gap | MaxList | time | gap | MaxList | time | gap |
| art1 | 761505 | 22.62 | – | 954269 | 14.45 | – | 850668 | 164.28 | – | 2884315 | 70.31 | – |
| art2 | 833806 | 15.69 | – | 937528 | 11.62 | – | 1533922 | 324.22 | – | 1655465 | 102.99 | – |
| art3 | 2218569 | 200.29 | – | 5398052 | 101.84 | – | 3472626 | 799.30 | – | 8899181 | 296.18 | – |
| art4 | 2072133 | 210.15 | – | 2228330 | 81.31 | – | 5461464 | 1934.30 | – | 5461464 | 509.90 | – |
| art5 | 13916512 | 1285.29 | – | 13942105 | 433.95 | – | 17259376 | 7172.07 | – | 18616932 | 1732.50 | – |
| art6 | 17322753 | 1884.49 | – | 17534221 | 681.16 | – | 39108015 | 17341.40 | – | × | 763.47 | 83.84 |
| art7 | 17115252 | 1830.39 | – | 21231438 | 685.81 | – | 21110879 | 11568.02 | – | 25765937 | 3450.06 | – |
| art8 | × | 286.62 | 28.17 | × | 388.10 | 34.34 | × | 416.04 | 80.16 | × | 509.84 | 91.73 |
| art9 | 17510258 | 3538.34 | – | 17525808 | 1663.44 | – | × | 443.31 | 99.23 | × | 590.92 | 99.23 |
| pmed1 | × | 5610.09 | 34.84 | × | 5318.60 | 34.84 | × | 2075.53 | 73.16 | × | 2099.87 | 73.16 |
| pmed2 | × | 2043.19 | 40.74 | × | 1877.88 | 40.74 | × | 1071.52 | 48.64 | × | 1089.60 | 48.64 |
| pmed3 | × | 3195.31 | 32.60 | × | 5224.86 | 33.86 | × | 2865.89 | 53.84 | × | 2843.13 | 53.84 |
| pmed4 | × | 1703.33 | 27.98 | × | 5537.69 | 28.46 | × | 3068.20 | 53.54 | × | 3033.74 | 53.54 |
| pmed5 | × | 6864.04 | 35.36 | × | 1704.56 | 39.75 | × | 1733.62 | 52.25 | × | 1713.50 | 52.25 |
| KROB150G | × | 3326.66 | 31.34 | × | 3234.95 | 31.34 | × | 8541.38 | 26.04 | × | 8453.09 | 26.04 |
| KROA150G | × | 4453.25 | 28.90 | × | 4371.62 | 28.90 | × | 2252.59 | 31.32 | × | 2242.95 | 31.32 |
| PR152G | 17820377 | × | 0.17 | 17968869 | × | 0.20 | × | 13184.08 | 45.96 | × | 13009.80 | 45.96 |
| RAT195G | × | 2947.69 | 28.09 | × | 2885.96 | 28.09 | × | 3035.05 | 23.54 | × | 3019.45 | 23.54 |
| KROB200G | × | 12617.27 | 26.03 | × | 12737.00 | 26.03 | × | 6584.91 | 32.84 | × | 6603.30 | 32.84 |
| KROA200G | × | 3645.04 | 40.16 | × | 3671.02 | 40.16 | × | 8926.33 | 43.28 | × | 8845.65 | 43.28 |
| TS225G | × | 9715.15 | 26.58 | × | 9851.17 | 26.58 | × | 8832.73 | 37.19 | × | 8742.36 | 37.19 |
| Average | 61408150 | 4142.61 | 18.14 | 61796220 | 3908.43 | 18.73 | 70895092 | 4873.08 | 33.38 | 74442061.62 | 3320.12 | 37.92 |

Table 3.8: Maximum branch and bound tree size, running times and achieved gaps for $p = 5$ for the superset approach.

| | superset | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Upper bound | IA | | | | | | IA+DC | | | | | |
| Lower bound | HuffDisc | | | MidPoint | | | HuffDisc | | | MidPoint | | |
| Network | MaxList | time | gap | MaxList | time | gap | MaxList | time | gap | MaxList | time | gap |
| art1 | 1452057 | 310.71 | – | 1451329 | 58.63 | – | 1452056 | 307.24 | – | 1451328 | 59.09 | – |
| art2 | 1994545 | 270.01 | – | 3696608 | 55.65 | – | 1994543 | 270.26 | – | 3696616 | 55.88 | – |
| art3 | 36610591 | 6922.33 | – | 92922451 | 1502.35 | – | 36610591 | 6910.81 | – | 92922451 | 1490.92 | – |
| art4 | 1228104 | 235.41 | – | 2984440 | 43.12 | – | 1228104 | 235.27 | – | 2984440 | 44.16 | – |
| art5 | 20255341 | 4142.36 | – | 45869245 | 785.61 | – | 20255346 | 4144.18 | – | 45869251 | 799.97 | – |
| art6 | 22539544 | 4639.56 | – | 29757712 | 727.59 | – | 22539545 | 4643.0 | – | 29757713 | 733.52 | – |
| art7 | 25822697 | 5571.06 | – | 31479371 | 896.23 | – | 25822697 | 5580.98 | – | 31479371 | 897.07 | – |
| art8 | × | 9939.57 | 4.60 | × | 1094.05 | 8.10 | × | 9937.81 | 4.60 | × | 1088.79 | 8.10 |
| art9 | × | 10181.58 | 8.96 | × | 1058.50 | 24.56 | × | 10103.78 | 8.96 | × | 1049.59 | 24.56 |
| pmed1 | 68661960 | × | 18.38 | × | 2901.09 | 19.27 | 68470552 | × | 18.40 | × | 2839.00 | 19.27 |
| pmed2 | 70797011 | × | 19.05 | × | 2774.52 | 22.43 | 70727962 | × | 18.98 | × | 2738.64 | 22.35 |
| pmed3 | 68662438 | × | 15.47 | × | 2865.02 | 19.57 | 68029779 | × | 15.54 | × | 2893.29 | 19.57 |
| pmed4 | 70145377 | × | 23.65 | × | 2827.24 | 26.15 | 71493389 | × | 22.38 | × | 2727.99 | 27.76 |
| pmed5 | 69754129 | × | 18.18 | × | 2756.32 | 17.42 | 69989853 | × | 18.08 | × | 2780.73 | 17.41 |
| KROB150G | 31327042 | × | 26.10 | × | 4152.59 | 24.74 | 36024130 | × | 21.09 | × | 3880.06 | 20.15 |
| KROA150G | 32557892 | × | 24.46 | × | 4095.34 | 21.55 | 35501051 | × | 20.57 | × | 3957.60 | 20.96 |
| PR152G | 34521048 | × | 6.92 | × | 4522.50 | 4.34 | 35281520 | × | 5.57 | × | 4524.22 | 4.13 |
| RAT195G | 20016605 | × | 48.90 | × | 5492.69 | 44.99 | 27657715 | × | 20.40 | × | 4631.51 | 24.53 |
| KROB200G | 17064415 | × | 29.03 | × | 5865.73 | 20.12 | 22428893 | × | 15.32 | × | 5085.60 | 15.25 |
| KROA200G | 16281856 | × | 29.79 | × | 5778.46 | 22.11 | 20716500 | × | 15.35 | × | 5165.29 | 14.37 |
| TS225G | 18788148 | × | 30.97 | × | 6013.59 | 23.07 | 21635793 | × | 21.94 | × | 5668.51 | 20.94 |
| Average | 39451467 | 14352.98 | 14.50 | 76579103 | 2679.37 | 14.21 | 40850477 | 14349.60 | 10.82 | 76579103 | 2529.12 | 12.35 |

# Part II

# Support Vector Machines

# Chapter 4

# Supervised Classification and Support Vector Machines

## 4.1 An introduction to Supervised Classification

The Hubble Space Telescope transmits about 120 gigabytes of data every week [74, 94]. This is just a fact that illustrates how the evolution of the digital world and the storage technology has lead to the growth of available data [85]. Data acquisition is the first step to exploit the power of learning. Predictions of the type "this person is likely to develop a cancer disease", "this product is likely to be a success" or "this client is not likely to return the loan" can be performed if one can derive patterns from data, such as clusters; or models, such as linear equations that classify data according to features we aim to study. Therefore, the interest in extracting valuable information from data is increasing constantly [135, 140, 145].

Data mining is the science of extracting useful information from large datasets [85]. The aim of data mining is to obtain and analyze information from datasets and transform it into understandable output for the user. Data mining algorithms are able to discover existing relationships between data or different level of representativeness of the data, such as predicting future country occupations or ranking patient features for medical diagnosis. It is a science attracting lots of research due to its increasing applicability, such as in the hot topic of social media [7, 10, 131].

One of the most important tasks in data mining is *Supervised Classification* [2, 6, 85, 160], in which we are given a set of objects $\Omega$ partitioned into classes and the aim is to build a procedure for classifying new objects. In its simplest form, each object $i \in \Omega$ has associated a pair $(x_i, y_i)$, where the feature vector $x_i$ takes values on a set $X \subseteq \mathbb{R}^d$ and $y_i \in \{-1, +1\}$ is the class membership of object $i$, also known as the label of object $i$, see Figure 4.1. In Part II, we will refer to the negative class as that in which $y_i = -1$, and to the positive class as that in which $y_i = +1$. Class membership is only known in a subset of $\Omega$, the *training sample*. Supervised Classification aims at finding a classification rule that predicts class membership for the remaining objects, that is, a function $f : X \to \{-1, +1\}$ which assigns class $f(x) \in \{-1, +1\}$ to feature vector $x$.

Supervised Classification plays an important role in many fields. Notable examples are found in health care, such as drug efficacy [89]; in demography, such as census income predictions [5]; in security, such as nuclear terrorism prevention [163]; or in computer vision, such as surveillance [71]. In the context of business applications

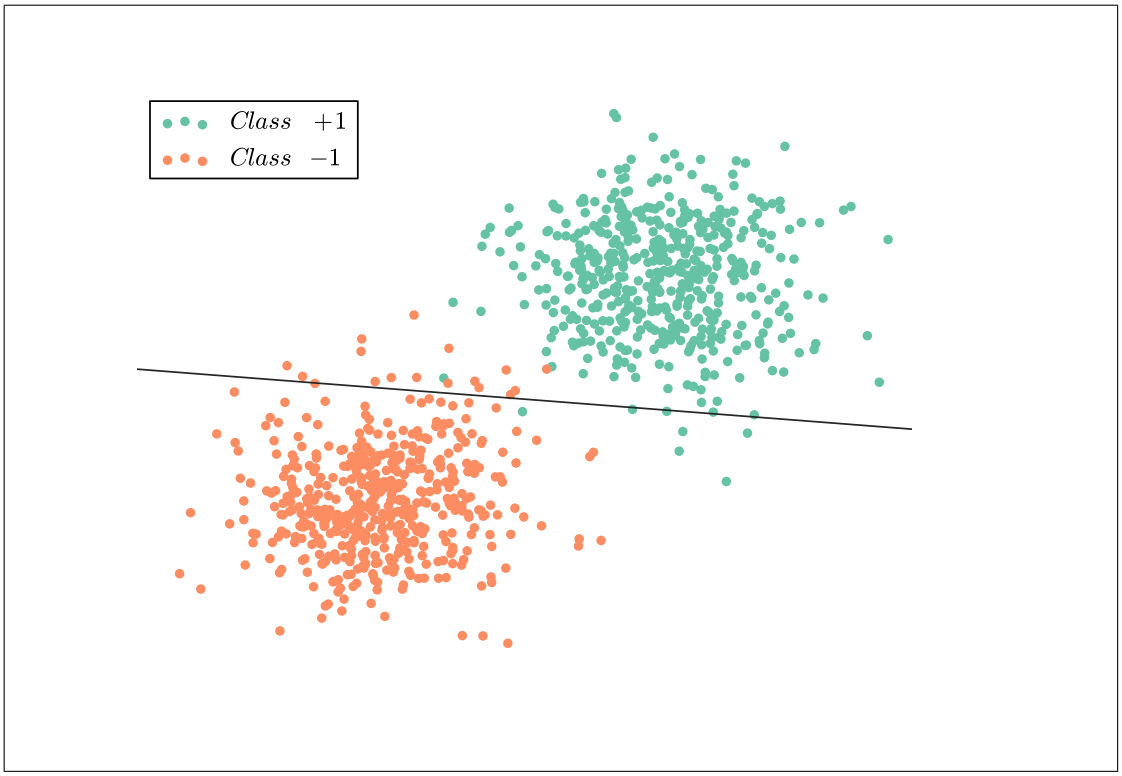


**Figure 4.1:** Objects from two classes, represented with different colors: objects with label $-1$ in red and with label $+1$ in green. The black line represents one possible classification rule. Objects below the line are classified into class $-1$ and into class $+1$ objects above it.

there are many typical examples, such as credit scoring [11], fraud detection [43] and customer targeting [8, 57]. For instance, it is a common business task to classify customers based on a number of features available to a company. In the case of credit scoring, the positive class is identified with customers with good credit risk, while the negative class with customers with bad credit risk. If we take as an example the `german` credit scoring dataset [23], widely used in the context of Supervised Classification and interpretability, such as in [11], each object $i$ represents a customer asking for a loan. For each customer, we know several characteristics such as "credit history", "purpose" (purpose for asking the loan, for instance, to buy a car or a television), or "housing" (for instance, if the customer owns, rents or has free housing), that compose the feature vector $x_i$. It is essential for the company to extract information from the dataset and build a procedure to decide if a given customer would pay back the loan, which means $y_i = +1$, or on the contrary would not pay back, i.e., $y_i = -1$.

The remainder of this chapter is organized as follows. In Section 4.2 we introduce Support Vector Machines. In Section 4.3 we define the criteria to measure the quality of a given classifier, and in Section 4.4 we discuss the cost of building it. In Section 4.5 we briefly discuss the contribution of each chapter in Part II. We end the chapter in Section 4.6 presenting the datasets used in this part of the dissertation.

## 4.2 Support Vector Machines

Support Vector Machines (SVM) [56, 152, 153] have proved to be one of the state-of-the-art methods for Supervised Classification [2, 6, 85, 160]. The SVM is a learning algorithm that, given a *training sample* with labeled objects, aims at separating classes by means of a classifier defined by a hyperplane, $\omega^\top x+b=0$, see Figure 4.2. See [42] for a recent review on Mathematical Optimization and the SVM. Successful applications of the SVM are found, for instance, in health care [22, 46, 81], fraud detection [43], credit scoring [113] and cancellations forecasting [138].

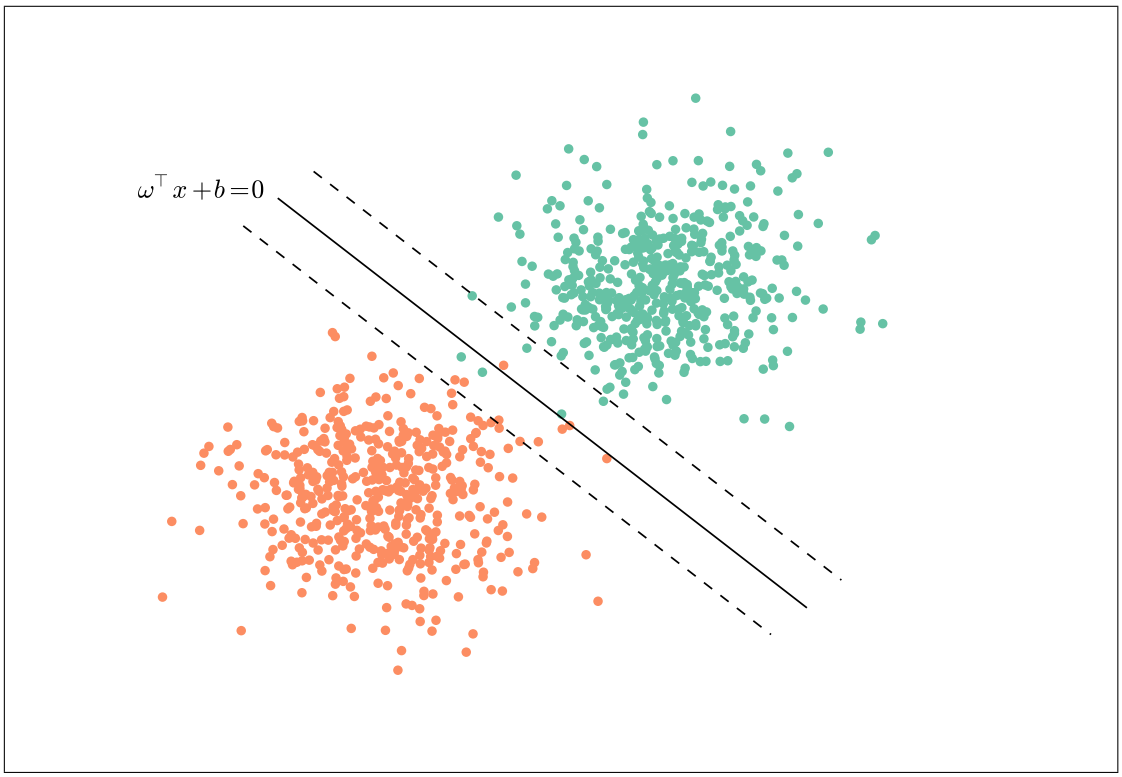


**Figure 4.2:** Training sample and the corresponding SVM classifier, where the discontinuous lines $\omega^\top x + b = 1$ and $\omega^\top x + b = -1$ define the margin boundary.

The SVM, in its typical form, finds the hyperplane $\omega^\top x + b = 0$ by minimizing the sum of the squared $L_2$ norm of the score vector $\omega$ and the so-called *hinge loss* function [42]. The SVM classifier is obtained by solving the following Quadratic Programming (QP) formulation with linear constraints:

$$\min_{\omega,b,\xi} \frac{1}{2}\sum_{j=1}^{d}\omega_j^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \tag{4.1}$$

s.t. (SVM)

$$y_i(\omega^\top x_i + b) \geq 1 - \xi_i \qquad \forall i = 1,\ldots,n \tag{4.2}$$

$$\xi_i \geq 0 \qquad \forall i = 1,\ldots,n \tag{4.3}$$

$$\omega \in \mathbb{R}^d \tag{4.4}$$

$$b \in \mathbb{R}, \tag{4.5}$$

where $n$ is the size of the training sample, $C$ is a nonnegative tradeoff parameter and $\xi = (\xi_i)$ is the vector of deviation variables.

The classifier obtained by solving the SVM, like any linear classifier, provides valuable information on the role of each feature in the classifier. Indeed, the set of features can be partitioned into three clusters, namely those with positive $\omega_j$, those with negative $\omega_j$, and the ones with $\omega_j$ equal to zero. We can say that features with positive (or negative) $\omega_j$ point towards the positive (negative) class, and thus have a positive (negative) contribution in the classifier, i.e., contribute to make $sign(\omega^\top x_i + b)$ equal to $+1$ $(-1)$. Features with $\omega_j = 0$ have no contribution in the classifier and are therefore irrelevant.

Many datasets are composed by continuous and categorical features. A given feature is designated categorical if it takes on a number of distinct values such as feature "continent" with categories "Africa", "America", "Asia", "Europe", "Oceania" and "Antarctica". For instance, the categorical feature "property" from the `german` dataset is composed by four categories, namely, "real estate", "building society savings agreement/life insurance", "car or other" and "unknown/no property". We denote with $J'$ the number of categorical features in the dataset, and with $K_{j'}$ the number of categories of feature $j'$. In the presence of both continuous and categorical features, an object $i \in \Omega$, is represented by the vector $(x_i, x'_i, y_i)$ where the feature vector $x'_i$ is associated with categorical features, that are binarized by splitting each feature into a series of 0-1 dummy features, one for each category, and takes values on a set $X' \subseteq \{0,1\}^{\sum_{j'=1}^{J'} K_{j'}}$. Then, the corresponding classifier is denoted by $(\omega)^\top x + (\omega')^\top x' + b = 0$, where $\omega$ is associated with the continuous features and $\omega'$ is associated with the categorical features. For the sake of simplicity, we will only use the notation $\omega'$ to refer to the scores of the categorical features when it is relevant to the methodology, i.e., in Chapter 6. In the rest of the chapters in Part II, we will assume that the categorical features are already incorporated into $x$.

## 4.3 Quality of the classifier

The quality of a classifier on a given dataset is measured by the classification accuracy. Given an object $i$, it is classified in the positive or the negative class according to the value of the score function, $sign(\omega^\top x_i + b)$, while for the case $\omega^\top x_i + b = 0$, the object is classified randomly. The classification accuracy is defined as the percentage of objects correctly classified by the classifier on such dataset. In the presence of outliers, misclassified objects that affect the performance of the algorithm and damage the classification accuracy, we say that the classifier is sensible to outliers and therefore the robustness of the classifier could be improved.

There are two other quality criteria that can be measured for a given classifier: sparsity and complexity. Sparsity is measured as the percentage of features with zero score, i.e., the sparsity of the SVM classifier is given by

$$\frac{card(\{\omega_j = 0\})}{d} \cdot 100\%. \tag{4.6}$$

Higher sparsity leads to cheaper classification cost as it reduces the number of features to be evaluated in the score function to compute the label of new objects. A desirable

property such as for gene expression [81] or credit scoring [11] datasets. The second criterion, complexity, is defined as the complement of sparsity; it quantifies (in percentage) the fraction of relevant features of the score vector, i.e., the complexity of the SVM classifier is given by

$$\frac{card(\{\omega_j \neq 0\})}{d} \cdot 100\%. \tag{4.7}$$

In the presence of different types of features, one can also measure the complexity for each type separately. It is interesting to measure the complexity of a classifier with respect to the categorical features, namely categorical complexity. It quantifies (in percentage) the fraction of relevant dummy features of the score vector associated with the categorical features, i.e., the categorical complexity of the SVM classifier is given by

$$\frac{card(\{\omega'_{j',k} \neq 0\})}{\sum_{j'=1}^{J'} K_{j'}} \cdot 100\%, \tag{4.8}$$

where $K_{j'}$ is the number of categories of categorical feature $j'$. Straightforward implementations of the SVM could lead to undesirable high values of the complexity for categorical features due to unexploiting the structure properties of categorical features. If we take as an example the `german` dataset, the SVM classifier leads to a complexity of 95.08% and to a categorical complexity of 94.23%. The low difference between the overall and the categorical complexity and the fact that the dataset is composed by 17% of continuous features and 83% of dummy features, means that reducing the categorical complexity of the classifier will benefit dramatically the overall complexity. Thanks to measuring the categorical complexity, we are able to discover that the high complexity values are due to the categorical features.

## 4.4 Building the classifier

The cost of classification is directly related to the cost of tuning parameters and building the classifier [37, 42]. As customary in Supervised Classification, the optimization of the SVM calls for tuning the tradeoff parameter $C$ [37, 42] by inspecting a grid of values. In the optimization of the SVM-based formulations of Chapter 5, the tuning is also performed for a vector of parameters $(c_1, \ldots, c_S)$. To perform the tuning process, the corresponding dataset is split into three sets, the so-called, training, testing and validation sets.

For each vector of parameters, the mathematical programming formulation is run on the training set. The different classifiers built in this way are compared according to their accuracy on the testing set. The vector of parameters with the highest accuracy on the testing set is chosen, and its accuracy on the validation set is reported, see Figure 4.3.

Tuning efficiently the parameters is a necessary and challenging problem [37], as shown in Figure 4.4, where the accuracy evolution in % is shown for a given grid of $C$ for the `careval` dataset solved by the SVM. The `careval` dataset [23], see Table 4.1, is a car evaluation dataset that evaluates the car acceptability according to 21 features that measure the car price, technical characteristics, comfort and safety.

Building the classifier, in the case of the SVM, becomes an easy task as it involves solving a QP formulation. In this part of the dissertation, Part II, the cost of building

**Step 1.** For each value of the parameter vector $(C, c_1, \ldots, c_S)$ in the grid,

(i) Obtain the corresponding classifier $\omega^\top x + b$ using the training set.

(ii) Let $\pi^{\text{test}}(C, c_1, \ldots, c_S)$ be the accuracy of $\omega^\top x + b$ in the testing set.

**Step 2.** Let $(C^*, c_1^*, \ldots, c_S^*) \in \arg\max_{(C,c_1,\ldots,c_S)} \pi^{\text{test}}(C, c_1, \ldots, c_S)$ and $(\omega^*)^\top x + b^*$ be the corresponding classifier.

(i) Report the quality of $(\omega^*)^\top x + b^*$ in the validation set.

**Figure 4.3:** Pseudocode of the tuning procedure of the tradeoff parameter $C$ as well as of a vector of $S$ parameters, $(c_1, \ldots, c_S)$.

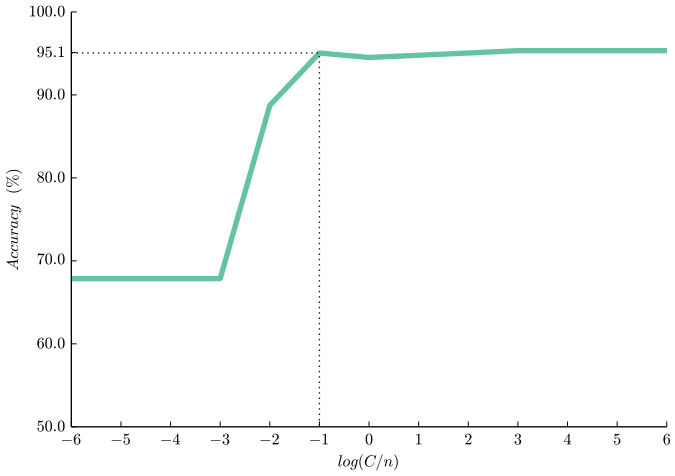


**Figure 4.4:** Accuracy evolution for different values of $C$ for the `careval` dataset solved by the SVM.

the classifier becomes a more complex task, as Chapters 5, 6 and 7 involve Mixed Integer Linear Programming (MILP), Mixed Integer Nonlinear Programming (MINLP), Mixed Integer Quadratic Programming (MIQP) and Quadratically Constrained Quadratic Programming (QCQP) formulations. Different quality criteria benefit from that: in Chapter 5, the use of the SVM-type classifiers obtained from these programming formulations benefits sparsity, in Chapter 6 categorical complexity is improved, while in the model addressed in Chapters 7 and 8 there is an increase in robustness.

## 4.5 Outline of Part II

In this part of the dissertation, Part II, we present four different works based on the SVM. Interpretability is the core of Chapters 5 and 6 while robustness is the main focus of the model addressed in Chapters 7 and 8. We will briefly discuss the contribution of each chapter in Part II.

In linear classifiers, such as the SVM, a score is associated with each feature and objects are assigned to classes based on the linear combination of the scores and the values of the features. The size of the score values associated with features is a valuable information that can be analyzed to obtain an interpretable classifier. Building interpretable classifiers is a desirable property that can be exploited in fields such as in health care or business. In Chapter 5, inspired by discrete psychometric scales, which measure the extent to which a factor is in agreement with a statement, we propose the Discrete Level Support Vector Machines (DILSVM) where the feature scores can only take on a discrete number of values, defined by the so-called feature rating levels. The DILSVM classifier benefits from interpretability as it can be seen as a collection of Likert scales, one for each feature, where we rate the level of agreement with the positive class. To build the DILSVM classifier, we propose an MILP approach, as well as a collection of strategies to reduce the building times. The computational experience reported at the end of Chapter 5, shows that the 3-point and the 5-point DILSVM classifiers have comparable accuracy to the SVM with a substantial gain in interpretability and sparsity, thanks to the appropriate choice of the feature rating levels.

The standard approach to handle datasets in the presence of categorical features is to binarize the categories into dummy features, disregarding its potential structure. Exploiting the structure of categorical features can benefit the process of building the classifier, obtaining less complex classifiers, a desirable property in health care and business applications. In Chapter 6, a methodology is proposed with the aim of reducing the categorical complexity of the SVM classifier in the presence of categorical features, the Cluster Support Vector Machines (CLSVM). The CLSVM methodology lets categories cluster around their peers and builds an SVM classifier using the clustered dataset. Four strategies for building the CLSVM classifier are presented based on solving: the original SVM formulation, a QCQP formulation, and an MIQP formulation as well as its continuous relaxation. The computational study reported at the end of Chapter 6, illustrates the quality of the CLSVM classifier using two clusters. In the tested datasets the CLSVM methodology achieves comparable accuracy to that of the SVM with original data but with a dramatic decrease in categorical complexity.

The SVM in its most popular form penalizes misclassified objects with the convex *hinge loss* function, $\sum_{i=1}^{n} \xi_i$, see Figure 4.5 (left). The *hinge loss* function measures the sum of the losses at the different objects as a continuous function, yielding smooth convex optimization problems, in fact, convex quadratic. These convex problems have been addressed in the literature by a collection of competitive algorithms and lead to computationally easy optimization problems but with an increased sensitivity to outlier observations. On the contrary, we have nonconvex loss functions such as the ramp loss, Figure 4.5 (right). In this case, the SVM formulation becomes an MIQP formulation and the optimization task becomes NP-hard but delivers a more robust classifier, see Figure 4.6, which makes the SVM with the ramp loss (RLM) an attractive model from the computational point of view. Chapters 7 and 8 are devoted to study of the RLM

model from two different viewpoints.

Chapter 7 proposes two heuristics to obtain the RLM classifier, the first one based on solving the continuous relaxation of an MINLP formulation of the RLM and the second one based on the training of an SVM classifier on a reduced dataset identified by an integer linear problem. The computational results reported at the end of Chapter 7, illustrate the ability of our heuristics to handle datasets of much larger size than those previously addressed in the literature of the RLM.

On the other hand, Chapter 8 focuses on solving the RLM to optimality by means of an MINLP reformulation. MILP models are commonly used to model indicator constraints, which either hold or are relaxed depending on the value of a binary variable. The RLM model is an important application of such models. MINLP models are usually dismissed because they cannot be solved as efficiently. However, the computational experience reported in Chapter 8 shows that the RLM model can be solved much more efficiently by an MINLP formulation with nonconvex constraints. This calls for a reconsideration of the modeling of these indicator constraints.

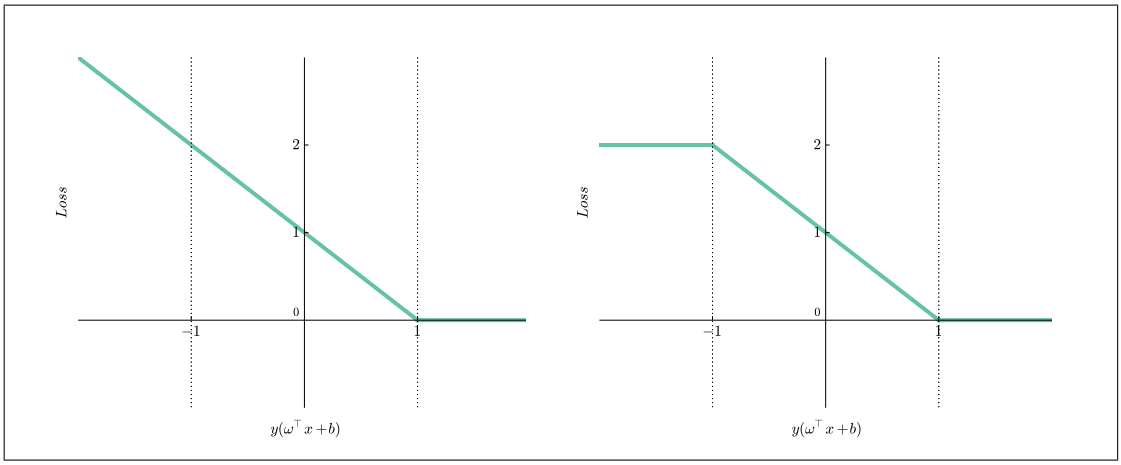


**Figure 4.5:** Hinge loss and ramp loss functions. Objects well classified have 0 loss for both functions: the case $y = +1$ with $\omega^\top x + b \geq 1$ and the case $y = -1$ with $\omega^\top x + b \leq -1$. When objects are misclassified, the hinge loss has an unbounded continuous loss while the ramp loss has a maximum loss of 2.

## 4.6 Datasets for Part II

To illustrate the performance of the strategies from Chapter 5, 6, 7 and 8, we use 15 real-life datasets and 2 synthetic datasets, see Table 4.1. Real-life datasets are first obtained from the StatLib repository [154], the LIBSVM repository [45], and the UCI repository [23], and then normalized. Synthetic datasets are obtained using the so-called `TypeA` and `TypeB` generators (for $d$ equal to 2, 5 and 10) from [33]. Datasets containing categorical features are transformed splitting the categories into binary dummy features. Regression datasets are transformed into 2-class datasets using the median, and multi-class datasets are transformed into 2-class, treating the largest class as class $+1$ and the remaining classes as class $-1$.

A description of these datasets can be found in Table 4.1, whose first four columns report the dataset name, full name given in the repository, total size of the dataset

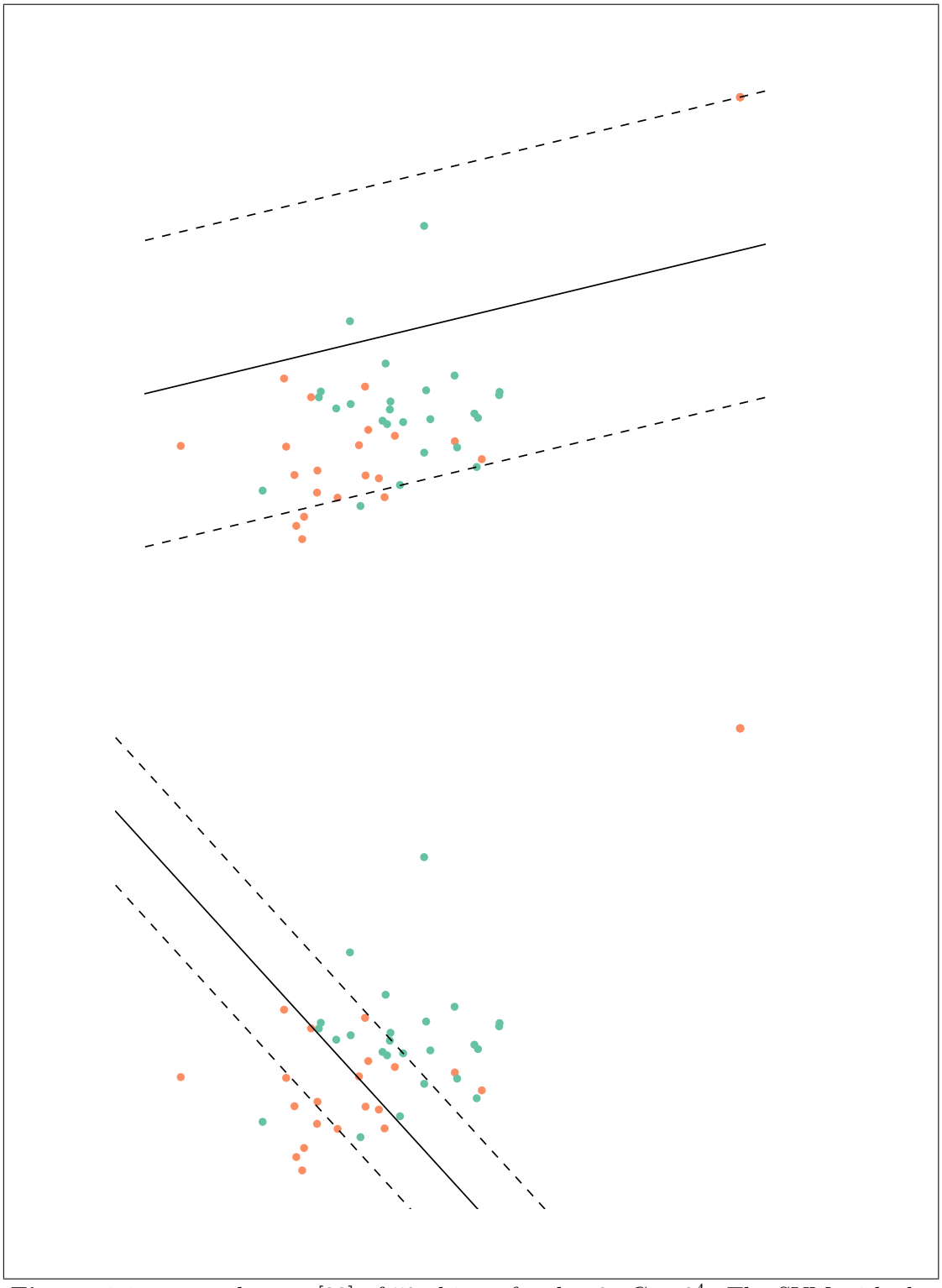

**Figure 4.6:** `TypeA` dataset [33] of 50 objects for $d = 2$, $C = 2^4$. The SVM with the hinge loss classifier (top) leads to a classification accuracy of 44% due to its sensivity to outliers. The SVM with the ramp loss yields a more robust classifier (bottom) with a classification accuracy of 78%.

($|\Omega|$) and number of features ($d$). Columns 5, 6, 7 and 8 report the size of the training set ($n$) used in Chapters 5, 6, 7 and 8 respectively, where "$-$" means the dataset is not used. The remaining records in the dataset are equally split between the testing and validation sets. Finally, the last column of Table 4.1 reports the class split in % in the dataset.

To obtain sharp estimates for the quality criteria, repeated random subsampling validation is used, where ten instances are run for each dataset. For synthetic datasets, the ten instances differ in the seed used to generate random data, whereas for real-life datasets, the seed is used to shuffle the set and then obtain different training, testing and validation sets.

Table 4.1: Real-life and synthetic datasets. They appear in increasing order with respect to the size of the dataset $|\Omega|$. Synthetic datasets are placed at the bottom of the table.

| Name | Name in Repository | $\|\Omega\|$ | $d$ | $n$ | | | | Class split |
|---|---|---|---|---|---|---|---|---|
| | | | | Chapter 5 | Chapter 6 | Chapter 7 | Chapter 8 | |
| `census income` | Census-Income (KDD) Dataset | 95130 | 500 | - | 5000 | - | - | 94/6 |
| `cod-rna` | cod-rna | 59535 | 8 | 30000 | - | 20000 | - | 33/67 |
| `shuttle` | Statlog (Shuttle) | 58000 | 9 | 30000 | - | - | - | 20/80 |
| `ijcnn1` | ijcnn1 | 35000 | 22 | 20000 | - | 20000 | - | 9/91 |
| `adult` | Adult | 30956 | 120 | 15000 | 5000 | 15000 | - | 24/76 |
| `calhous` | California Housing | 20460 | 8 | 10000 | - | - | - | 50/50 |
| `gamma` | MAGIC Gamma Telescope | 19020 | 10 | 10000 | - | 10000 | - | 32/68 |
| `mushrooms` | Mushrooms | 8124 | 115 | 4000 | 5000 | - | - | 48/52 |
| `coil 2000` | Insurance Company Benchmark (COIL 2000) | 5822 | 157 | - | 3900 | - | - | 94/6 |
| `abalone` | Abalone | 4177 | 10 | 2000 | 2800 | - | - | 50/50 |
| `molecular` | Molecular Biology (Splice-junction Gene Sequences) | 3190 | 480 | - | 2200 | - | - | 52/58 |
| `careval` | Car Evaluation | 1728 | 21 | 1000 | 1200 | - | - | 30/70 |
| `solar-c` | Solar Flare Dataset | 1066 | 28 | - | 800 | - | - | 83/17 |
| `german` | Statlog (German Credit Data) | 1000 | 61 | 500 | 700 | - | - | 30/70 |
| `australian` | Statlog (Australian Credit Approval) | 690 | 39 | - | 500 | - | - | 56/44 |
| `TypeA` | - | 15000 | 2,5,10 | - | - | 5000 | - | 50/50 |
| `TypeB` | - | 15000 | 2,5,10 | - | - | 5000 | 100 | 50/50 |

# Chapter 5

# Strongly Agree or Strongly Disagree?: Rating Features in Support Vector Machines

In this chapter [38] we propose an SVM-type classifier where feature scores can only take on a discrete number of values. By adding parameters at the right place we improve the sparsity of the classifier and thus interpretability. Therefore, the classifier can be seen as a (small) collection of Likert scales, easy to understand for nonexperts. This chapter contributes to the literature on SVM, but more generally, on data mining, because it significantly improves interpretability of learning methods and their visualization, without comprising accuracy.

## 5.1 Introduction

There are several desirable managerial properties in Supervised Classification methods. Incorporating domain knowledge at the time of building the classifier is a very appealing property [43]. One may also like to keep the costs of learning the classifier low, where these costs arise from retrieving the features [35, 149], obtaining feature information from the classifier [141], or refreshing the classifier to include new data [71]. Another desirable property is the comprehensibility/interpretability of the classifier. Since conciseness of the classifier is closely related with its interpretability, much effort has been devoted to increasing its sparsity, i.e., to reducing the number of active features in the classifier [81], discretizing the features to detect active ranges of the features [108, 138] or relevant thresholds for the features [36]. When trading off between accuracy and interpretability, another popular approach has been to extract easy–to–understand structures, such as if–then rules, decision trees and decision tables, from powerful black–box type classifiers [11, 44, 113, 114, 125, 126]. Yet another way of achieving interpretability is by building discrete linear classifiers [48, 79]. Classifier interpretability is a major challenge in datasets that contain a huge number of features, where most of these are irrelevant. To address this issue, models have been proposed aimed at increasing the sparsity of the classifier, as is done with the Lasso, see [106] and [139].

In Supervised Classification, linear classifiers are based on score functions. For

instance, the $CHADS_2$ score [77, 105] is widely used in medicine to predict the risk of stroke in patients with atrial fibrillation based on five different symptoms of the patient, whereas the extended $CHA_2DS_2$–$VAS_c$ score [107] includes three additional risk factors. A score is associated with each feature, and objects are assigned to classes based on the linear combination of the scores and the values of the features. The role each feature plays in the classifier is related to the magnitude of the corresponding score, while the sign gives information on how the feature points towards a given class.

Inspired by discrete psychometric scales, which measure the extent to which a factor is in agreement with a statement, [102], and by linear classifiers in which the weights are allowed to take values on a discrete set, [48, 79], we propose the Discrete Level Support Vector Machines (DILSVM) where the feature scores can only take on a discrete number of levels, defined by the so-called feature rating levels. The DILSVM classifier benefits from interpretability, as it can be seen as a collection of Likert scales, one for each feature, where we rate the level of agreement with the positive class.

We formulate the DILSVM as a Mixed Integer Linear Programming (MILP) problem. The parameters of our model, namely, the tradeoff parameter $C$ and the rating levels, may heavily influence the quality of the DILSVM and, therefore, have to be carefully chosen. Thus, building the DILSVM classifier involves solving a series of MILP problems, which, if solved to optimality, may make the overall computational cost high for large datasets. For this reason we propose three strategies to alleviate the computational burden, with different tradeoffs between quality and reduction in computational cost. Such strategies use the guidance of related but simpler optimization models, and reduce the computational cost associated with each parameter vector and/or the number of parameter vectors to be inspected.

In our computational experience, we compare the DILSVM against the SVM classifier in terms of two quality criteria, namely accuracy and sparsity. We show that the 3-point and 5-point DILSVM classifiers, built using the MILP approach, have comparable accuracy to the SVM, with a substantial gain in sparsity. Moreover, by the very nature of our procedure (only a few levels, to be interpreted as intensities, are allowed), interpretability is definitely improved allowing us to visualize the classification process via Likert scales. The tests illustrating the quality of the reduction strategies reveal a clear competitiveness in terms of sparsity, while the accuracy depends on the magnitude of the reduction, being close to the SVM accuracies. In our computational experience, we also compare the DILSVM against the 3-point DILSVM classifier with fixed parameters using ten real-life datasets. The 3-point DILSVM with fixed parameters is inspired by the model in [48]. The results illustrate the relevance of tuning parameters to ensure that accuracy and sparsity are not compromised.

The remainder of this chapter is organized as follows. In Section 5.2 we introduce the DILSVM classifier. In Section 5.3 we discuss procedures for building the DILSVM classifier. In Section 5.4 we report our computational results using real-life datasets. We end the chapter in Section 5.5 with some conclusions and directions for future research.

## 5.2 The DILSVM classifier

The Discrete Level Support Vector Machines (DILSVM) is a variant of the SVM classifier where for each feature $j$, the score $\omega_j$ can only take on a discrete number of values.

Let $\mathscr{A} \subset \mathbb{R}$ be a finite set that includes the value 0, which models the marginal impact of the feature on the classifier. We can formulate the DILSVM as the following Discrete Quadratic Programming problem:

$$\min_{\omega, b, \xi} \frac{1}{2} \sum_{j=1}^{d} \omega_j^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i \tag{5.1}$$

s.t.

$$y_i(\omega^\top x_i + b) \geq 1 - \xi_i \qquad \forall i = 1, \ldots, n \tag{5.2}$$
$$\xi_i \geq 0 \qquad \forall i = 1, \ldots, n \tag{5.3}$$
$$\omega_j \in \mathscr{A} \qquad \forall j = 1, \ldots, d \tag{5.4}$$
$$b \in \mathbb{R}. \tag{5.5}$$

For adequate choices of $\mathscr{A}$, this model gains in interpretability and visualization. For instance, let us consider $\mathscr{A} = \{-a, 0, a\}$, $a > 0$. In the DILSVM classifier, some of the $\omega_j$ will be equal to $a$ or $-a$, while the rest will be equal to zero. Features for which $\omega_j = a$ will have a positive impact on the classifier and therefore will point towards the positive class, those for which $\omega_j = -a$ will point towards the negative class, while the rest have no impact. We can view this in an alternative way: for a given rating level $a$, the DILSVM detects those features which *strongly agree* with the positive class, those which *strongly disagree* (and therefore strongly agree with the negative class), and those which are irrelevant to the classifier. This DILSVM classifier can be represented as a collection of 3-point Likert scales, one for each feature, measuring the extent to which the feature is in agreement with the positive class. When looking for more granularity of the scale, we can increase the size of $\mathscr{A}$. For $\mathscr{A} = \{-a_1, -a_2, 0, a_2, a_1\}$, $a_1 > a_2 > 0$, the DILSVM classifier can be seen as a collection of 5-point Likert scales where features $j$ with $\omega_j = a_1$ are seen to *strongly agree* with the positive class, those with $\omega_j = a_2$ *agree* (but not so strongly), while $\omega_j = -a_1$ ($-a_2$) *strongly disagree* (*disagree*).

As an illustration, let us consider the `german` dataset introduced in Chapter 4, which is one of the datasets used in in our computational tests in Section 5.4, with 61 features in total. Table 5.4 displays the DILSVM classifier with $\mathscr{A} = \{-1, 0, 1\}$ and $C/n = 10^6$ as a collection of 3-point Likert scales. Let us focus on three of the features, namely "having no debtor", "having a co-applicant" and 'having a guarantor'. The only relevant feature is "having a guarantor", strongly contributing to the positive class, while for the other two features the score is equal to zero. This is a pattern that can be extended to the overall classifier, where more than half of the features are irrelevant (41 out 61). The remaining features are roughly equally split between the left and right side of the scale. Thus, in addition to the gain in interpretability, this DILSVM classifier gains in sparsity too, see equation (4.6). The 5-point Likert scale representation of the DILSVM classifier with $\mathscr{A} = \{-1, -1/2, 0, 1/2, 1\}$ and $C/n = 10^6$ is given in Table 5.5. The feature "having a guarantor" still strongly contributes to the positive class, but "having a co-applicant" now contributes negatively (but not strongly), while "having no debtor" is still irrelevant. Similarly to the 3-point Likert scale classifier, the split between the left and right side of the scale is balanced, but the number of irrelevant features is about a third (27 out of 61).

Apart from the gain in interpretability and sparsity, once the DILSVM classifier

has been obtained, its evaluation (i.e., classifying new objects) is as inexpensive as for the SVM. However, these advantages come with an increased computational burden, related to the choice of the set $\mathscr{A}$, to build the classifier. We analyze this issue in the next section.

## 5.3 Constructing the DILSVM classifier

Inspired by Likert scales, we assume that the set $\mathscr{A}$ is symmetric and defined as $\mathscr{A} = \{-a_1, \ldots, -a_K, 0, a_K, \ldots, a_1\}$, where $a_1 > \ldots > a_K > 0$ are the so-called rating levels telling us about the extent to which each feature is in agreement with the positive class. We denote this model by DILSVM$^{(K)}$. Please note that $\mathscr{A}$ could be considered asymmetric without loss of generality.

Our model involves $K + 1$ parameters, namely the $K$ rating levels as well as the tradeoff parameter $C$. In the following, we formulate the DILSVM$^{(K)}$, when the $K+1$ parameters are fixed, as an MILP problem. As pointed in Section 4.4, we will illustrate in Section 5.4, that tuning efficiently the parameters is a necessary and challenging problem, [37]. In order to alleviate this computational burden, a collection of strategies is proposed.

### 5.3.1 An MILP formulation

In this section we formulate (5.1)–(5.5) with $\mathscr{A} = \{-a_1, \ldots, -a_K, 0, a_K, \ldots, a_1\}$ as an MILP problem. For each feature $j$ and each rating level $a_k$, let $\alpha_{jk}$ be equal to either $-1, 1$ or 0, indicating whether $\omega_j$ is equal to $-a_k$, $a_k$ or none of these two. For each feature $j$, at most one $\alpha_{jk}$ variable can be different from zero. We can now rewrite

$$\begin{aligned} \omega_j &= \sum_{k=1}^{K} a_k \alpha_{jk} \\ \sum_{j=1}^{d} \omega_j^2 &= \sum_{j=1}^{d} \sum_{k=1}^{K} a_k^2 \alpha_{jk}^2 = \sum_{j=1}^{d} \sum_{k=1}^{K} a_k^2 |\alpha_{jk}|, \end{aligned}$$

where the latter follows from the fact that $\alpha_{jk}\alpha_{jk'} = 0$ for $k \neq k'$. It is straightforward to see that by making these substitutions in (5.1)–(5.5) and adding the constraints relating to $\alpha_{jk}$, the DILSVM$^{(K)}$ can be formulated as:

$$\min_{\alpha, b, \xi} \frac{1}{2} \sum_{j=1}^{d} \sum_{k=1}^{K} a_k^2 |\alpha_{jk}| + \frac{C}{n} \sum_{i=1}^{n} \xi_i$$

s.t.

$$
\begin{aligned}
y_i \left( \sum_{j=1}^{d} \sum_{k=1}^{K} a_k \alpha_{jk} x_{ij} + b \right) &\geq 1 - \xi_i && \forall i = 1, \ldots, n \\
\xi_i &\geq 0 && \forall i = 1, \ldots, n \\
\sum_{k=1}^{K} |\alpha_{jk}| &\leq 1 && \forall j = 1, \ldots, d \\
\alpha_{jk} &\in \{-1,\ 0,\ 1\} && \forall j = 1, \ldots, d,\ \forall k = 1, \ldots, K \\
b &\in \mathbb{R}.
\end{aligned}
$$

Using the usual trick to transform an absolute value into linear constraints, i.e., $\alpha_{jk} = \alpha^+_{jk} - \alpha^-_{jk}$, and $|\alpha_{jk}| = \alpha^+_{jk} + \alpha^-_{jk}$, with $\alpha^+_{jk}, \alpha^-_{jk} \in \{0, 1\}$, we can reformulate the DILSVM$^{(K)}$ as an MILP problem:

$$\min_{\alpha^+, \alpha^-, b, \xi} \frac{1}{2} \sum_{j=1}^{d} \sum_{k=1}^{K} a_k^2 (\alpha^+_{jk} + \alpha^-_{jk}) + \frac{C}{n} \sum_{i=1}^{n} \xi_i \tag{5.6}$$

s.t. (DILSVM$^{(K)}$)

$$y_i \left( \sum_{j=1}^{d} \sum_{k=1}^{K} a_k (\alpha^+_{jk} - \alpha^-_{jk}) x_{ij} + b \right) \geq 1 - \xi_i \qquad \forall i = 1, \ldots, n \tag{5.7}$$

$$\xi_i \geq 0 \qquad \forall i = 1, \ldots, n \tag{5.8}$$

$$\sum_{k=1}^{K} (\alpha^+_{jk} + \alpha^-_{jk}) \leq 1 \qquad \forall j = 1, \ldots, d \tag{5.9}$$

$$\alpha^+_{jk}, \alpha^-_{jk} \in \{0,\ 1\} \qquad \forall j = 1, \ldots, d,\ \forall k = 1, \ldots, K \tag{5.10}$$

$$b \in \mathbb{R}. \tag{5.11}$$

This MILP formulation has $n + d$ constraints, $2dK$ binary decision variables, $n$ non-negative and 1 free, while the SVM formulation has a quadratic objective function but all the decision variables are continuous.

In terms of the choice of the number of rating levels, the DILSVM$^{(K)}$ may lead to a higher classification accuracy than the DILSVM$^{(K')}$, with $K > K'$, but it is computationally more expensive as the number of zero–one decision variables increases and the gains in interpretability and sparsity are less dramatic. Similar observations can be made when comparing the SVM and the DILSVM$^{(K)}$, since at the end of the spectrum is the SVM, which can be seen as DILSVM$^{(\infty)}$. In the tradeoff between accuracy and interpretability, we recommend the $K = 1$ and $K = 2$ versions of the model, namely the DILSVM$^{(1)}$ and the DILSVM$^{(2)}$.

### 5.3.2 Reduction strategies

We have formulated the DILSVM$^{(K)}$ as an MILP problem. Solving this MILP formulation to optimality is an NP-hard task even if we only consider one single rating level, namely $K = 1$ and $a_1 = 1$, and $C = \infty$, as shown in [48]. In addition, the quality of the DILSVM$^{(K)}$ classifier may be strongly influenced by the choice of the tradeoff param-

eter $C$ as well as the $K$ rating levels. Thus, building the DILSVM classifier involves solving a series of MILP problems, which, if solved to optimality, may make the overall computational cost high for large datasets. In this section we present three different strategies to alleviate the computational burden of building the DILSVM$^{(K)}$ classifier.

The proposed strategies use the guidance of related but simpler optimization models, and reduce the computational cost associated with each parameter vector $(C, a_1, \ldots, a_K)$ and/or the number of parameter vectors to be inspected. The first strategy is based on rounding the SVM classifier using each vector of rating levels. The second strategy proposes, for each parameter vector, the randomized rounding, [132], of the Linear Programming (LP) relaxation of the DILSVM$^{(K)}$. The third strategy aims at speeding up the process for the DILSVM$^{(K)}$ based on information readily available for the DILSVM$^{(K')}$, $K' < K$. Clearly, these strategies are of diverse nature and, as the computational experience will illustrate, offer different tradeoffs between quality and reduction in computational cost.

The first and second strategies will be presented for general $K$, while, and for the sake of clarity, the third strategy will be described for $K = 2$, though the process gracefully extends to arbitrary $K$. Apart from a grid of parameter vectors, we assume that we are given a selection criterion, [42], to choose the optimal classifier among those generated for each vector of the grid. This is usually the accuracy on an independent sample different from the one used to build the classifier, see Figure 4.3 from Chapter 4.

The first strategy, the *SVM rounding* (*RSVM*), is based on rounding the feature scores of the SVM classifier using the rating levels. For each value of $C$, the SVM classifier is obtained, and the rounding procedure is performed using each vector of rating levels. The pseudocode of this reduction strategy is given in Figure 5.1, where $a_0 = \infty$. While this strategy examines all the parameter vectors in the grid, it is appealing since it is cheaper to train an SVM than a DILSVM. In addition, for a given value of $C$, once the SVM classifier has been obtained, the rounding procedure takes $\mathcal{O}(dK)$ time for each vector of rating levels.

The second strategy is based on the randomized rounding of the LP relaxation of the DILSVM$^{(K)}$. We call this the *randomized rounding* (*RR*) strategy. For each parameter vector $(C, a_1, \ldots, a_K)$, the *RR* strategy solves the LP relaxation of the DILSVM$^{(K)}$, where constraints (5.10) are relaxed to $\alpha^+_{jk}, \alpha^-_{jk} \in [0, 1]$. Let $(\alpha^+, \alpha^-)$ be the (partial) optimal solution obtained. Without loss of optimality, $\alpha^+_{jk} \cdot \alpha^-_{jk} = 0$. Thus, for a given feature $j$, $\alpha^+_{jk}$ ($\alpha^-_{jk}$) can be seen as the desirability of setting the score of feature $j$ to value $a_k$ ($-a_k$). Noting that the assignment constraints (5.9) are satisfied, a randomized rounding procedure can be applied to derive the DILSVM classifier. In order to ensure feasibility with respect to the assignment constraints, the rounding needs to take place in a predetermined order of the rating levels, such that once a rating level and a sign has been assigned to a feature, we move to the next feature. In the current version of the *RR* strategy, the rating levels are arranged in decreasing order of $\max\{\alpha^+_{jk}, \alpha^-_{jk}\}$. The pseudocode of this reduction strategy can be found in Figure 5.2, where `rand`$(p)$ is a subroutine of random numbers generation, returning the value 1 with probability p and 0 otherwise.

In the third strategy, the *fixing* strategy, we use the output of the DILSVM$^{(1)}$ classifier to alleviate the burden of building the DILSVM$^{(2)}$ classifier. In this case, the reduction is twofold. First, the size of the parameter space is narrowed down.

**Step 1.** For each $C$,

(i) Solve the $SVM$ and obtain the (partial) optimal solution $(\omega, b)$.

(ii) For each $(a_1, \ldots, a_K)$,

For $j = 1, \ldots, d$

For $k = 1, \ldots, K$, set $\beta_{jk}^+ = \beta_{jk}^- = 0$

For $k = 1, \ldots, K$

If $a_k \leq \omega_j < a_{k-1}$ then $\beta_{jk}^+ = 1$

ElseIf $-a_{k-1} < \omega_j \leq -a_k$ then $\beta_{jk}^- = 1$

end

end

Return the classifier $\sum_{k=1}^{K} a_k(\beta_k^+ - \beta_k^-)^\top x + b$.

**Step 2.** Choose the optimal parameter vector using $\sum_{k=1}^{K} a_k(\beta_k^+ - \beta_k^-)^\top x + b$.

**Figure 5.1:** Pseudocode for the *SVM rounding* strategy.

Second, some of the decision variables in the MILP formulation (5.6)–(5.8) are fixed in advanced, and therefore eliminated. The pseudocode for this reduction strategy can be found in Figure 5.3. Using the grid corresponding to parameters $C$ and $a_1$, we first build the DILSVM$^{(1)}$ classifier yielding optimal parameters values $C^{(1)}$ and $a_1^{(1)}$. We then build the DILSVM$^{(2)}$ classifier using the MILP formulation (5.6)–(5.8) for $K = 2$, with $C = C^{(1)}$, $a_1 = a_1^{(1)}$ and the values of $a_2$ in the grid. In addition, we reduce the number of zero–one decision variables in the MILP formulation. In the current version of the *fixing* strategy, feature $j$ will have a strong positive rating in the DILSVM$^{(2)}$ classifier if that was the case in the DILSVM$^{(1)}$ classifier, and similarly for a strong negative rating. For the rest of features, $\beta_{jk}^+$ and $\beta_{jk}^-$, $k = 1, 2$, must be optimized.

As already mentioned, the *fixing* strategy can be extended to an arbitrary $K$, where we use the output of the DILSVM$^{(K')}$ classifier with $K' < K$.

## 5.4 Computational results

In this section we illustrate the quality of the DILSVM classifier in terms of accuracy and sparsity. As benchmark procedure we use the SVM, whose quality in terms of both criteria is reported in Table 5.1. To build the DILSVM classifier, we use four approaches, namely the MILP approach and the three reduction strategies. We will show that, considered in its full generality and with small values of $K$ ($K = 1, 2$) to ensure interpretability, the DILSVM$^{(K)}$ is competitive against the SVM in terms of accuracy, being substantially sparser than the latter. Inspired by the model in [48], we show in Table 5.2 results for the DILSVM$^{(1)}$ with fixed parameters $C = \infty$ and

**Step 1.** For each $(C, a_1, \ldots, a_K)$,

(i) Solve the LP relaxation of DILSVM$^{(K)}$ and obtain the (partial) optimal solution $(\alpha^+, \alpha^-)$.

(ii) For $j = 1, \ldots, d$

For $k = 1, \ldots, K$, set $\beta^+_{jk} = \beta^-_{jk} = 0$

Set $\mathscr{K} = \{1, \ldots, K\}$

while $(\mathscr{K} \neq \emptyset)$

Let $\bar{k}$ such that $\max\{\alpha^+_{j\bar{k}}, \alpha^-_{j\bar{k}}\} \geq \max\{\alpha^+_{jk}, \alpha^-_{jk}\}$, $\forall k \in \mathscr{K}$

Set $\beta^+_{j\bar{k}} = \texttt{rand}(\alpha^+_{j\bar{k}})$

If $\beta^+_{j\bar{k}} = 1$, set $\mathscr{K} = \emptyset$

Else

Set $\beta^-_{j\bar{k}} = \texttt{rand}(\alpha^-_{j\bar{k}})$

If $\beta^+_{j\bar{k}} = \beta^-_{j\bar{k}} = 0$, set $\mathscr{K} = \mathscr{K} \setminus \{\bar{k}\}$

Else $\mathscr{K} = \emptyset$

end

end

(iii) Return the classifier $\sum_{k=1}^{K} a_k(\beta^+_k - \beta^-_k)^\top x + b$.

**Step 2.** Choose the optimal parameter vector using $\sum_{k=1}^{K} a_k(\beta^+_k - \beta^-_k)^\top x + b$.

**Figure 5.2:** Pseudocode for the *randomized rounding* strategy.

$a_1 = 1$. Comparing the DILSVM$^{(1)}$ with it shows the relevance of tuning parameters. The results for the DILSVM$^{(1)}$ are reported in Table 5.2, where for each dataset and each criterion, we underline the best results across the three approaches. (Note that the *fixing* strategy only applies to $K \geq 2$, and therefore it is not present in Table 5.2.) Similarly, Table 5.3 reports the results for the four approaches to build the DILSVM$^{(2)}$ classifier.

Our experiments have been conducted on a PC with an Intel Core i7 processor, 16 Gb of RAM. We use the optimization engine `IBM-Cplex` v12.4, [98], for solving all optimization problems. We have set the time limit to 300 seconds, which is enough for most of the optimization problems we have solved. For the remaining ones, the classifiers derived in this way are of heuristic nature.

The construction of the DILSVM classifier calls for tuning some parameters, namely the tradeoff parameter $C$ as well as the rating levels. For that purpose, we use the tuning procedure in Figure 4.3 where $c_s = a_s$, $\forall s = 1, \ldots, K$. Following the usual approach, for the DILSVM$^{(1)}$, parameters $C$ and $a_1$ are tuned by inspecting a grid of the form $\frac{C}{n} \in \{10^{-6}, \ldots, 10^6\}$ and of the form $a_1 \in \{2^0, \ldots, 2^{10}\}$. For the DILSVM$^{(2)}$, $C$ and

**Step 1.** For each $(C, a_1)$,

(i) Solve the DILSVM$^{(1)}$ and obtain the (partial) optimal solution $(\alpha^+, \alpha^-, b)$.

(ii) Choose the optimal parameter vector $(C^{(1)}, a_1^{(1)})$ using $a_1(\alpha^+ - \alpha^-)^\top x + b$, and obtain $(\bar{\alpha}^+, \bar{\alpha}^-)$.

**Step 2.** For $C^{(1)}, a_1^{(1)}$, and for each $a_2$,

(i) For $j = 1, \ldots, d$

If $\bar{\alpha}^+_{j1} = 1$ then $\beta^+_{j1} = 1$ and $\beta^-_{j1} = \beta^+_{j2} = \beta^-_{j2} = 0$

ElseIf $\bar{\alpha}^-_{j1} = 1$ then $\beta^-_{j1} = 1$ and $\beta^+_{j1} = \beta^+_{j2} = \beta^-_{j2} = 0$

end

(ii) Solve the DILSVM$^{(2)}$ and return the classifier $\sum_{k=1}^{2} a_k(\beta_k^+ - \beta_k^-)^\top x + b$.

**Step 3.** Choose the optimal parameter vector using $\sum_{k=1}^{2} a_k(\beta_k^+ - \beta_k^-)^\top x + b$.

**Figure 5.3:** Pseudocode for the *fixing* strategy.

$a_1$ are tuned with the same grid, and $a_2 \in \{\frac{a_1}{2}, \frac{a_1}{2^2}\}$. In order to show the relevance of tuning parameters, results for the DILSVM$^{(1)}$ with $C = \infty$ and $a_1 = 1$ are reported in Section 5.4.1.

The quality in terms of accuracy and sparsity of our model is illustrated using 10 real-life large datasets whose description can be found in Table 4.1. The size of the training set $(n)$ is set as the closest $5 \cdot 10^{m-1}$ multiple to $|\Omega|/2$ where $|\Omega|$ is of $10^m$ order, see the fifth column of Table 4.1.

### 5.4.1 Results for the MILP approach

In this section we compare the quality of the DILSVM$^{(1)}$ and the DILSVM$^{(2)}$ against that of the SVM, where the DILSVM results are generated using the MILP formulation in Section 5.3.1. Quality will be measured in terms of two criteria, accuracy and sparsity. The ideal model would be that one achieving the highest values in both criteria. When, for a given criterion, the difference in quality between two approaches is 1 percentage point (p.p.) or below, we will say that both approaches are comparable under that criterion.

Recall that Table 5.1 reports the quality of the SVM, while the MILP approach results for the DILSVM$^{(1)}$ can be found in the second set of columns of Table 5.2, and the ones for the DILSVM$^{(2)}$ in the first set of columns of Table 5.3. For each dataset, we report the mean validation accuracy across the ten instances, as well as the standard deviation and the median. The same statistics are presented for the sparsity. Below we discuss mean values, but similar conclusions are derived if the median is used.

We start with the analysis of the mean accuracy, and show that the DILSVM is competitive against the SVM. We first compare with the mean accuracy of the SVM. For three datasets, `adult`, `german` and `careval`, the DILSVM$^{(1)}$ outperforms the SVM by 5.54, 2.36 and 1.16 p.p., respectively. For three datasets, `mushroom`, `gamma` and `shuttle`, the DILSVM$^{(1)}$ and the SVM are comparable. In three datasets, `ijcnn1`, `abalone` and `calhous`, the SVM outperforms the DILSVM$^{(1)}$ by 1.20, 1.24 and 1.61 p.p., respectively. In `cod-rna`, the DILSVM$^{(1)}$ is clearly inefficient compared to the SVM. For datasets such as `cod-rna`, the DILSVM$^{(1)}$ is too restrictive, and the accuracy may benefit from the additional flexibility built in by the DILSVM$^{(2)}$. An improvement of more than 1 p.p. can be observed from the DILSVM$^{(1)}$ to the DILSVM$^{(2)}$ in three datasets. The first one is `careval`, for which the DILSVM$^{(1)}$ is already better than the SVM, and the improvement on the SVM increases from 1.66 to 3.66 p.p. The second one is `cod-rna`, where now the difference in mean accuracy between the SVM and the DILSVM has been reduced from 15.69 to 3.22 p.p. The third one is `calhous`, where the SVM is better than the DILSVM$^{(1)}$, but now the DILSVM$^{(2)}$ has a comparable quality to the SVM. Across all datasets, the DILSVM$^{(2)}$ outperforms the SVM in three datasets, they are comparable in five datasets, while the SVM outperforms the DILSVM$^{(2)}$ in two datasets. Thus, the DILSVM$^{(2)}$ is competitive against the SVM in terms of mean accuracy. We now show the relevance of tuning parameters by comparing mean accuracy results between the DILSVM$^{(1)}$ and the DILSVM$^{(1)}$ with fixed parameters $C = \infty$ and $a_1 = 1$. Using this criterion, the fixed DILSVM$^{(1)}$ is not competitive and is outperformed by the DILSVM$^{(1)}$ in eight datasets, where the increase in mean accuracy achieved ranges between 2.80 and 18.41 p.p., while the mean accuracy of both methods is basically the same for `mushroom` and `ijcnn1`.

We now focus on the second criterion, and show that the DILSVM is the best in terms of mean sparsity. Indeed, for each dataset, the best model is the DILSVM$^{(1)}$, followed by the DILSVM$^{(2)}$, and the SVM being the worse. Note that the sparsity quality of the SVM is rather poor, being always fully dense (except for `german`), i.e., all features have nonzero coefficients, and therefore play a role in the classifier. In terms of mean sparsity, the DILSVM$^{(1)}$ with fixed parameters is clearly outperformed by the DILSVM$^{(1)}$, except for `cod-rna`, in which the mean sparsity is exactly the same for both methods. We now take a closer look at the mean sparsity of our model and show how useful the MILP approach is to determine a high number of irrelevant features which may make the classifier harder to interpret and also may negatively affect accuracy due to overfitting. The mean sparsity for the DILSVM$^{(1)}$ is always 50% or above except for `calhous`, with 31.25%, while for the DILSVM$^{(2)}$ the mean sparsity is always above 35% except for `calhous`, with 17.50%. Thus, except for `calhous`, at least half of the features are irrelevant in the DILSVM$^{(1)}$, while this becomes at least one third in the DILSVM$^{(2)}$. The results are even more encouraging for five of these datasets, where the mean sparsity of the DILSVM$^{(1)}$ is at least 70%, while the mean sparsity of the DILSVM$^{(2)}$ is at least 50%. In addition, in the `mushroom` dataset, both the DILSVM$^{(1)}$ and the DILSVM$^{(2)}$ report a 100% accuracy with a 91.60% mean sparsity.

In summary, the model we propose, the DILSVM, is competitive against the SVM in terms of accuracy, being substantially sparser than the latter. When faced with the choice between one or two rating levels, we should observe that the DILSVM$^{(1)}$ is, in general, more appealing: it has comparable accuracies to those of the DILSVM$^{(2)}$ in seven datasets, while the DILSVM$^{(1)}$ is at least as sparser as the DILSVM$^{(2)}$. Thus, one

rating level will, in general, suffice. In the remaining three datasets, including `cod-rna`, the DILSVM$^{(1)}$ underperforms in terms of accuracy, while the additional rating level available in the DILSVM$^{(2)}$ boosts the accuracy to a competitive level, at the cost of a lower sparsity.

The MILP approach to build the DILSVM yields attractive results in terms of both criteria against both benchmarks. That comes with a high computational cost though, where an MILP problem needs to be solved for each parameter vector. This can be illustrated by the median ratio across the ten datasets between the building time of the DILSVM and that of the SVM, which is equal to 210.91 for $K = 1$ and 429.60 for $K = 2$. When comparing with the fixed DILSVM$^{(1)}$, the median ratio is 361.18 for $K = 1$ and 1054.53 for $K = 2$. We should emphasize that this high computational effort only affects the phase of building the classifier (off-line). The evaluation phase (on-line), when new objects are to be classified, is even faster than with the SVM, since, due to the high sparsity of the classifier, fewer terms are computed. In addition, as shown in the next section, the three strategies proposed in Section 5.3.2 are able to reduce the time to build the DILSVM classifier. In terms of quality, while the sparsity is not compromised, the accuracy will depend on the magnitude of the time reduction.

### 5.4.2 Results for the reduction strategies

In this section we illustrate the quality of the three reduction strategies proposed in Section 5.3.2, as well as their building times. The accuracy and the sparsity results can be found in Tables 5.2 ($K = 1$) and 5.3 ($K = 2$), where we present the mean, the standard deviation and the median of both criteria. Clearly, there is a drop in accuracy with respect to the corresponding MILP approach. Below we show that these strategies behave well against the benchmark. As before, we start with the mean accuracy.

In the *RSVM* strategy, we round the feature scores of the SVM classifier using the rating levels in the grid. This is a cheap option, in which the optimization problems involved have only continuous decision variables. We now analyze the mean accuracy, and show that the *RSVM* strategy is dominated by the SVM. For $K = 1$, the *RSVM* strategy is clearly inefficient against the SVM, where the loss in accuracy is at least 1 p.p. in all datasets, while this becomes 47.97 p.p. in `mushroom` and 22.81 p.p. in `cod-rna`. The improvement of $K = 2$ is dramatic for `mushroom`, where the *RSVM* strategy becomes comparable to the SVM, also in `ijcnn1`, while in the remaining eight datasets this strategy is still not competitive against the SVM.

In the *RR* strategy, we reduce the computational cost of solving an MILP problem for each parameter vector. Instead, the *RR* strategy solves the LP relaxation of the DILSVM$^{(K)}$ and applies a randomized rounding to its optimal solution. Below, we show that the *RR* strategy yields mean accuracies close to those of the SVM. For $K = 1$, the loss in accuracy of the *RR* strategy compared to the SVM ranges from 0.51 to 17.73 p.p., and therefore, using this criterion, the *RR* strategy is dominated by the SVM. For $K = 2$, the *RR* strategy outperforms the SVM in `adult` by 4.15 p.p. and in `careval` by 3.71 p.p., for three datasets both methods are comparable, while in the remaining five the losses in accuracy (in p.p.) with respect to the SVM are 1.12 (`ijcnn1`), 1.87 (`shuttle`), 1.88 (`abalone`), 3.12 (`calhous`) and 13.76 (`cod-rna`). Except for `cod-rna`, we can conclude that the *RR* strategy with $K = 2$ and the SVM are comparable in terms of accuracy.

In the *fixing* strategy, we first build the DILSVM$^{(1)}$, and use its classifier to reduce both the number of parameter vectors to be inspected as well as the number of zero–one decision variables in the MILP. For each dataset, this strategy reduces the number of MILP problems to be solved from 286 (full grid inspected) to 2 (only grid for $a_2$ inspected), after solving the 143 MILP problems associated with the DILSVM$^{(1)}$. Roughly speaking, this halves the number of MILP problems to be solved. We now analyze the mean accuracy, and show that the *fixing* strategy yields results close to those of the SVM. Compared to the SVM, the *fixing* strategy outperforms the SVM in `adult` by 4.19 p.p. and in `careval` by 1.79 p.p., for three datasets both methods are comparable, while in the remaining five the SVM performs better. If we ignore `cod-rna` with a 12.54 p.p. loss, one can see that the improvement of the SVM for the remaining four datasets is below 1.36 p.p. Except for `cod-rna`, we can conclude that the *fixing* strategy has a comparable behavior to the SVM.

In terms of mean sparsity, the strategies clearly dominate the SVM, as the latter is always fully dense (except for `ijcnn1`). We now take a closer look at the mean sparsity of each strategy. For $K = 1$, the mean sparsity of the *RSVM* strategy is at least 50% for all datasets with eight datasets above 70%, while for $K = 2$ the mean sparsity is above 25% for all datasets with seven above 50%. For $K = 1$, the mean sparsity of the *RR* strategy is at least 65% for all datasets with eight datasets above 70%, while for $K = 2$ the mean sparsity is above 15% for all datasets with eight datasets above 50%. Finally, the sparsity of the *fixing* strategy is at least 20% for all datasets with five datasets above 50%.

We now illustrate the reduction in building time achieved by the strategies. As before, we measure the ratio between the building time of a given strategy and that of the SVM, and report the median ratio across the ten datasets. For the *RSVM* strategy, the rounding time is negligible for small values of $K$. Thus, for $K = 1, 2$, the ratio between the building time of the *RSVM* and the SVM is roughly equal to 1 for each dataset. For the *RR* strategy, when comparing with the SVM, the median ratio is equal to 10.32 for $K = 1$ and 34.41 for $K = 2$. Finally, the *fixing* strategy is designed to reduce the building time when $K = 2$ using the DILSVM$^{(1)}$ classifier, and the median ratios are very similar to the ones reported for the MILP approach with $K = 1$, namely 211.60 against the SVM.

In conclusion, we have presented three strategies of very diverse nature that are able to reduce the building time of the DILSVM classifier. When compared to the SVM, the *RR* and the *fixing* strategies are competitive in terms of mean accuracy, but less so is the *RSVM*, while the dominance of the three strategies in terms of sparsity is overwhelming.

## 5.5 Conclusions

In this chapter we propose the DILSVM$^{(K)}$ classifier, an SVM-type classifier in which there are $K$ possible levels of agreement of each feature with the positive class. We recommend small values of $K$, such as $K = 1$ and $K = 2$. In this case, our classifier enjoys two properties. First, it can be visualized as a collection of Likert scales, and therefore, since $K$ is small, the DILSVM$^{(K)}$ classifier gains in interpretability. Second, once the classifier has been built, the evaluation of the DILSVM$^{(K)}$ (i.e., classifying new objects) is at least as inexpensive as for the SVM: classifying new objects amounts

to evaluating a linear function. In addition, as shown by our computational experience, many coefficients are set to zero in both the DILSVM$^{(1)}$ and the DILSVM$^{(2)}$, while the SVM is fully dense.

The gain in visualization and sparsity achieved by the DILSVM is made without paying any price in accuracy. As our computational experience shows, the DILSVM is competitive against the SVM in terms of accuracy. Our classifier is much harder to obtain than the SVM, since an MILP problem is to be solved for each parameter vector, and we have illustrated that parameter tuning is crucial if we do not want to compromise accuracy. In terms of the number of parameters, there are now one ($K = 1$) or two ($K = 2$) more parameters than in the standard SVM to tune. In order to alleviate the computational burden, three reduction strategies have been proposed. Our computational tests show that we are able to preserve an accuracy comparable to the SVM and significantly better sparsities. This means that, at the expense of an increase in off-line computational cost, the DILSVM is able to extract easy-to-interpret information from datasets without sacrificing classification accuracy.

We conclude with three promising extensions of our approach. First, knowledge domain can also be incorporated into the model. Given a family of features, constraints of the type "at least one feature of this family should be selected" or "no more than one feature of this family should be chosen" simply lead to new linear constraints in the MILP formulation. Second, accuracy and interpretability are usually contradicting objectives. In this chapter we fix the number of rating levels, and aim at optimizing accuracy. It is natural to consider the problem of simultaneous optimization of both accuracy and interpretability, see for instance [91, 120]. This biobjective model deserves further analysis and testing. Third, our approach can also be extended to other linear classifiers, such as the classical Linear Discriminant Analysis [73] and the Logistic Regression [88], where building the new classifier yields Nonlinear Integer Programming problems [34]. Solving these nonlinear problems efficiently remains an important future challenge.

Table 5.1: Validation quality in % with $C/n \in \{10^{-6}, \ldots, 10^{6}\}$: the SVM.

| *Name* | *SVM* | | | | | |
|---|---|---|---|---|---|---|
| | Accuracy | | | Sparsity | | |
| | mean | std | med | mean | std | med |
| adult | 84.89 | 0.28 | 84.98 | 0.00 | 0.00 | 0.00 |
| mushroom | 99.98 | 0.04 | 100.00 | 0.00 | 0.00 | 0.00 |
| german | 72.96 | 2.37 | 72.60 | 4.12 | 1.05 | 3.97 |
| ijcnn1 | 91.37 | 0.36 | 91.33 | 0.00 | 0.00 | 0.00 |
| careval | 95.05 | 0.86 | 94.91 | 0.00 | 0.00 | 0.00 |
| gamma | 79.32 | 0.57 | 79.47 | 0.00 | 0.00 | 0.00 |
| abalone | 79.52 | 1.14 | 79.73 | 0.00 | 0.00 | 0.00 |
| shuttle | 98.17 | 0.08 | 98.15 | 0.00 | 0.00 | 0.00 |
| cod-rna | 93.86 | 0.12 | 93.89 | 0.00 | 0.00 | 0.00 |
| calhous | 83.59 | 0.29 | 83.59 | 0.00 | 0.00 | 0.00 |

Table 5.2: Validation quality in % with $C/n \in \{10^{-6}, \ldots, 10^{6}\}$: the MILP approach and the reduction strategies for the DILSVM$^{(1)}$.

| *Name* | *MILP* $C = \infty, a_1 = 1$ | | | | | | *MILP* $a_1 \in \{2^0, \ldots, 2^{10}\}$ | | | | | | *RSVM* $a_1 \in \{2^0, \ldots, 2^{10}\}$ | | | | | | *RR* $a_1 \in \{2^0, \ldots, 2^{10}\}$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Accuracy | | | Sparsity | | | Accuracy | | | Sparsity | | | Accuracy | | | Sparsity | | | Accuracy | | | Sparsity | | |
| | mean | std | med | mean | std | med | mean | std | med | mean | std | med | mean | std | med | mean | std | med | mean | std | med | mean | std | med |
| adult | 82.93 | 0.48 | 82.91 | 55.93 | 9.65 | 55.69 | 90.43 | 0.64 | 90.69 | 79.27 | 8.80 | 74.80 | 75.81 | 0.28 | 75.90 | 100.00 | 0.00 | 100.00 | 82.33 | 0.36 | 82.19 | 97.36 | 0.93 | 97.56 |
| mushroom | 99.98 | 0.04 | 100.00 | 49.08 | 3.32 | 50.42 | 100.00 | 0.00 | 100.00 | 91.60 | 0.00 | 91.60 | 52.01 | 1.05 | 51.87 | 100.00 | 0.00 | 100.00 | 99.47 | 0.23 | 99.47 | 97.56 | 2.27 | 98.32 |
| german | 72.52 | 2.14 | 72.40 | 51.43 | 7.31 | 50.00 | 75.32 | 5.70 | 73.20 | 70.95 | 12.17 | 69.84 | 69.24 | 3.00 | 69.20 | 98.89 | 2.84 | 100.00 | 69.28 | 2.72 | 68.80 | 93.02 | 6.49 | 94.05 |
| ijcnn1 | 90.18 | 0.36 | 90.03 | 81.82 | 22.27 | 100.00 | 90.17 | 0.36 | 90.03 | 100.00 | 0.00 | 100.00 | 90.18 | 0.36 | 90.03 | 98.18 | 3.64 | 100.00 | 90.17 | 0.37 | 90.04 | 96.59 | 4.22 | 100.00 |
| careval | 86.92 | 2.47 | 86.81 | 30.00 | 15.21 | 33.30 | 96.21 | 1.98 | 96.84 | 59.52 | 5.73 | 61.90 | 82.03 | 5.32 | 84.48 | 62.86 | 15.18 | 61.90 | 87.61 | 1.62 | 87.78 | 81.19 | 7.18 | 80.95 |
| gamma | 70.37 | 0.60 | 70.47 | 31.00 | 5.39 | 30.00 | 79.09 | 0.64 | 79.22 | 50.00 | 10.95 | 45.00 | 77.95 | 0.71 | 77.79 | 80.00 | 0.00 | 80.00 | 78.31 | 0.49 | 78.26 | 73.00 | 4.00 | 75.00 |
| abalone | 74.28 | 0.81 | 73.95 | 15.00 | 5.00 | 15.00 | 78.28 | 0.99 | 78.17 | 70.00 | 8.94 | 70.00 | 77.19 | 1.85 | 77.85 | 72.00 | 6.00 | 70.00 | 76.91 | 1.32 | 77.30 | 75.50 | 5.22 | 75.00 |
| shuttle | 80.83 | 1.26 | 80.63 | 15.55 | 7.37 | 16.66 | 97.32 | 0.11 | 97.33 | 63.33 | 5.09 | 66.67 | 84.30 | 5.09 | 82.76 | 75.56 | 9.69 | 77.78 | 93.75 | 1.77 | 92.81 | 68.33 | 7.88 | 66.67 |
| cod-rna | 66.56 | 0.38 | 66.58 | 87.50 | 0.00 | 87.50 | 78.17 | 0.28 | 78.20 | 87.50 | 0.00 | 87.50 | 71.05 | 3.09 | 71.35 | 88.75 | 3.75 | 87.50 | 76.13 | 1.51 | 75.70 | 93.75 | 0.00 | 93.75 |
| calhous | 63.57 | 2.47 | 63.48 | 0.00 | 0.00 | 0.00 | 81.98 | 0.50 | 81.78 | 31.25 | 8.39 | 37.50 | 78.60 | 2.00 | 79.06 | 50.00 | 0.00 | 50.00 | 78.17 | 1.17 | 78.16 | 65.00 | 5.73 | 68.75 |

Table 5.3: Validation quality in % with $C/n \in \{10^{-6}, \ldots, 10^{6}\}$: the MILP approach and the reduction strategies for the DILSVM$^{(2)}$.

| *Name* | *MILP* $a_1 \in \{2^0, \ldots, 2^{10}\}, a_2 \in \{a_1/2, a_1/2^2\}$ | | | | | | *RSVM* $a_1 \in \{2^0, \ldots, 2^{10}\}, a_2 \in \{a_1/2, a_1/2^2\}$ | | | | | | *RR* $a_1 \in \{2^0, \ldots, 2^{10}\}, a_2 \in \{a_1/2, a_1/2^2\}$ | | | | | | *fixing* $C^{(1)}, a_1^{(1)}, a_2 \in \{a_1^{(1)}/2, a_1^{(1)}/2^2\}$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Accuracy | | | Sparsity | | | Accuracy | | | Sparsity | | | Accuracy | | | Sparsity | | | Accuracy | | | Sparsity | | |
| | mean | std | med | mean | std | med | mean | std | med | mean | std | med | mean | std | med | mean | std | med | mean | std | med | mean | std | med |
| adult | 90.56 | 0.46 | 90.75 | 77.40 | 11.64 | 78.45 | 77.88 | 2.41 | 77.22 | 90.33 | 7.01 | 90.25 | 89.04 | 1.99 | 89.14 | 93.33 | 8.39 | 95.93 | 89.08 | 1.63 | 88.66 | 67.07 | 19.10 | 73.98 |
| mushroom | 100.00 | 0.00 | 100.00 | 91.60 | 0.00 | 91.60 | 99.41 | 0.14 | 99.42 | 95.80 | 0.00 | 95.80 | 100.00 | 0.00 | 100.00 | 100.00 | 0.00 | 100.00 | 100.00 | 0.00 | 100.00 | 91.60 | 0.00 | 91.60 |
| german | 75.96 | 5.32 | 74.00 | 55.87 | 22.06 | 55.56 | 70.92 | 3.38 | 71.00 | 57.78 | 20.21 | 47.62 | 73.00 | 4.51 | 72.20 | 54.76 | 20.59 | 46.03 | 71.60 | 2.26 | 71.60 | 35.56 | 10.56 | 36.51 |
| ijcnn1 | 90.18 | 0.36 | 90.03 | 95.46 | 13.63 | 100.00 | 90.46 | 0.45 | 90.51 | 65.91 | 7.67 | 70.46 | 90.25 | 0.33 | 90.14 | 70.45 | 4.66 | 72.73 | 90.17 | 0.36 | 90.03 | 100.00 | 0.00 | 100.00 |
| careval | 98.71 | 0.44 | 98.77 | 35.72 | 2.39 | 35.72 | 89.81 | 3.22 | 89.97 | 43.34 | 6.88 | 42.86 | 98.76 | 0.43 | 98.90 | 53.81 | 30.57 | 38.10 | 96.84 | 0.81 | 97.25 | 51.90 | 12.68 | 61.90 |
| gamma | 79.31 | 0.65 | 79.11 | 50.00 | 16.73 | 40.00 | 78.28 | 0.39 | 78.18 | 52.00 | 12.49 | 50.00 | 79.19 | 0.84 | 79.34 | 57.00 | 14.87 | 60.00 | 79.36 | 0.64 | 79.49 | 32.00 | 7.48 | 30.00 |
| abalone | 79.05 | 1.07 | 78.77 | 55.00 | 9.22 | 50.00 | 78.28 | 0.76 | 78.03 | 57.00 | 14.87 | 65.00 | 77.64 | 1.36 | 77.62 | 65.00 | 19.62 | 65.00 | 78.43 | 1.05 | 78.44 | 61.00 | 13.75 | 50.00 |
| shuttle | 97.59 | 0.27 | 97.51 | 45.55 | 13.57 | 44.44 | 92.16 | 3.77 | 92.16 | 43.33 | 20.76 | 33.33 | 96.30 | 0.83 | 96.29 | 43.33 | 27.87 | 33.33 | 97.32 | 0.09 | 97.34 | 47.78 | 15.76 | 50.00 |
| cod-rna | 90.64 | 0.88 | 90.57 | 42.50 | 11.46 | 50.00 | 73.94 | 5.88 | 74.03 | 82.50 | 8.29 | 81.25 | 80.10 | 4.60 | 82.46 | 62.50 | 12.50 | 62.50 | 81.32 | 2.18 | 80.25 | 46.25 | 11.25 | 50.00 |
| calhous | 83.03 | 0.24 | 83.09 | 17.50 | 6.12 | 12.50 | 80.14 | 0.84 | 80.23 | 25.00 | 19.36 | 12.50 | 80.47 | 0.64 | 80.50 | 17.50 | 6.12 | 12.50 | 82.25 | 0.44 | 82.12 | 20.00 | 6.12 | 25.00 |

Table 5.4: 3-point Likert scale for `german` dataset with the DILSVM.

The following features contribute to being a good customer:

| | | Strongly Agree | Neutral | Strongly Disagree |
|---|---|---|---|---|
| Status of existing checking account | ... < 0 DM | | × | |
| | 0 ≤ ... < 200 DM | | × | |
| | ... ≥ 200 DM | × | | |
| | No checking account | × | | |
| Credit duration | | | | × |
| Credit history | no credits taken/all credits paid back duly | | | × |
| | all credits at this bank paid back duly | | | × |
| | existing credits paid back duly till now | | × | |
| | delay in paying off in the past | | × | |
| | critical account/other credits existing (not at this bank) | × | | |
| Purpose | new car | | | × |
| | used car | | × | |
| | furniture/equipment | | × | |
| | radio/television | | × | |
| | domestic appliances | | | × |
| | repairs | | × | |
| | education | | | × |
| | vacation | | × | |
| | retraining | × | | |
| | business | | × | |
| | others | | × | |
| Credit amount | | | | × |
| Saving accounts | ... < 100 DM | | × | |
| | 100 ≤ ... < 500 | | × | |
| | 500 ≤ ... < 1000 | × | | |
| | ... ≥ 1000 | × | | |
| | unknown/no saving accounts | × | | |
| Duration of present employment | unemployed | | × | |
| | ... < 1 year | | × | |
| | 1 ≤ ... < 4 years | | × | |
| | 4 ≤ ... < 7 years | | × | |
| | ... ≥ 7 years | | × | |
| Installment rate in % of disposable income | | | | × |
| Personal status and sex | male (divorced/separated) | | | × |
| | female (divorced/separated/married) | | × | |
| | male (single) | | × | |
| | male (married/widowed) | | × | |
| | female (single) | | × | |
| Other debtors/guarantors | none | | × | |
| | co-applicant | | × | |
| | guarantor | × | | |
| Duration of present residence | | | × | |
| Property | real estate | | × | |
| | building society savings agreement/life insurance | | × | |
| | car/others | | × | |
| | unknown/no property | | | × |
| Age | | × | | |
| Other installment plans | bank | | × | |
| | stores | | × | |
| | none | | × | |
| Housing | rent | | × | |
| | own | | × | |
| | for free | × | | |
| Number of existing credits at this bank | | | | × |
| Job | unemployed/unskilled (nonresident) | | × | |
| | unskilled (resident) | | × | |
| | skilled employee/official | | × | |
| | management/self-employed/highly qualified employee officer | | × | |
| Number of people being liable to provide maintenance for | | | × | |
| Telephone | None | | × | |
| | Yes | | × | |
| Foreign worker | Yes | | × | |
| | No | × | | |

Table 5.5: 5-point Likert scale for `german` dataset with the DILSVM.

The following features contribute to being a good customer:

| | | Strongly Agree | Agree | Neutral | Disagree | Strongly Disagree |
|---|---|---|---|---|---|---|
| Status of existing checking account | $\ldots < 0$ DM | | | | × | |
| | $0 \leq \ldots < 200$ DM | | | × | | |
| | $\ldots \geq 200$ DM | × | | | | |
| | No checking account | × | | | | |
| Credit duration | | | | | | × |
| Credit history | no credits taken/all credits paid back duly | | | | | × |
| | all credits at this bank paid back duly | | | | | × |
| | existing credits paid back duly till now | | | × | | |
| | delay in paying off in the past | | × | | | |
| | critical account/other credits existing (not at this bank) | × | | | | |
| Purpose | new car | | | | | × |
| | used car | | | × | | |
| | furniture/equipment | | | | × | |
| | radio/television | | | | × | |
| | domestic appliances | | | | | × |
| | repairs | | | | × | |
| | education | | | | | × |
| | vacation | | | × | | |
| | retraining | × | | | | |
| | business | | | | × | |
| | others | | | × | | |
| Credit amount | | | | | | × |
| Saving accounts | $\ldots < 100$ DM | | × | | | |
| | $100 \leq \ldots < 500$ | | × | | | |
| | $500 \leq \ldots < 1000$ | × | | | | |
| | $\ldots \geq 1000$ | × | | | | |
| | unknown/no saving accounts | × | | | | |
| Duration of present employment | unemployed | | | × | | |
| | $\ldots < 1$ year | | | × | | |
| | $1 \leq \ldots < 4$ years | | | × | | |
| | $4 \leq \ldots < 7$ years | | × | | | |
| | $\ldots \geq 7$ years | | | × | | |
| Installment rate in % of disposable income | | | | | | × |
| Personal status and sex | male (divorced/separated) | | | | | × |
| | female (divorced/separated/married) | | | | × | |
| | male (single) | | | × | | |
| | male (married/widowed) | | | × | | |
| | female (single) | | | × | | |
| Other debtors/guarantors | none | | | × | | |
| | co-applicant | | | | × | |
| | guarantor | × | | | | |
| Duration of present residence | | | | × | | |
| Property | real estate | | | × | | |
| | building society savings agreement/life insurance | | | × | | |
| | car/others | | | × | | |
| | unknown/no property | | | | | × |
| Age | | × | | | | |
| Other installment plans | bank | | | × | | |
| | stores | | | × | | |
| | none | | × | | | |
| Housing | rent | | × | | | |
| | own | | × | | | |
| | for free | × | | | | |
| Number of existing credits at this bank | | | | | | × |
| Job | unemployed/unskilled (nonresident) | | | × | | |
| | unskilled (resident) | | | × | | |
| | skilled employee/official | | | × | | |
| | management/self-employed/highly qualified employee officer | | | × | | |
| Number of people being liable to provide maintenance for | | | | × | | |
| Telephone | None | | | × | | |
| | Yes | | | × | | |
| Foreign worker | Yes | | | × | | |
| | No | × | | | | |

# Chapter 6

# Clustering Categories in Support Vector Machines

Exploiting the structure of categorical features can benefit the process of building the classifier, obtaining less complex classifiers. This chapter [39] contributes to the literature on SVM and data mining, because it reduces the categorical complexity of the SVM classifier in the presence of categorical features without comprising accuracy. These advantages are achieved by defining a new methodology that lets categories cluster around their peers and builds an SVM classifier using the clustered dataset.

## 6.1 Introduction

In many applications of Supervised Classification, datasets are composed by a large number of features and/or objects, making it hard to both build the classifier and interpret the results. In this case, it is desirable to obtain a less complex classifier, which may make classification easier to handle and interpret, less prone to overfitting and computationally cheaper when classifying new objects [11, 38, 44, 113, 114, 125, 126]. The most popular strategy proposed in the literature to achieve this goal is feature selection [76, 81, 158], which aims at selecting the subset of most relevant features for classification while maintaining or improving accuracy and preventing the risk of overfitting. Feature selection reduces the number of features by means of all·or·nothing procedure. For categorical features, binarized as explained above, it simply ignores some categories of some features, and does not give valuable insight on the relationship between feature categories. These issues may imply a significant loss of information.

In this chapter, a methodology to reduce the complexity of the SVM classifier for datasets composed by categorical features, sometimes containing many categories, and occasionally continuous features, is proposed. This is done by clustering the different categories of each categorical feature into a given number of clusters, and then obtaining an SVM-type classifier for the clustered dataset. We call this the Cluster Support Vector Machines (CLSVM) methodology and we will refer to the CLSVM classifier.

As an illustration, let us consider the `german` dataset introduced in Chapter 4. In this dataset each object is originally composed by 20 features: 11 categorical features, binarized into 52 dummies, and 9 continuous features, finally obtaining 61 features in total. For this dataset, the SVM formulation with original data, hereafter denoted by $\text{SVM}^{\text{O}}$, gives a classifier, $(\omega)^\top x + (\omega')^\top x' + b = 0$, leading to a classification ac-

curacy of 76.67% and whose categorical score subvector $\omega'$ has 50 relevant features, i.e., $card(\{\omega'_{j'} \neq 0\}) = 50$. However, using the CLSVM methodology described in this chapter, where the categories of each categorical feature are grouped just into two clusters, the classification accuracy is increased to 80.00% while the CLSVM classifier uses $2 \times 11 = 22$ relevant dummies. In other words, the methodology proposed here allows one to obtain a much simpler classifier with an accuracy even higher than the original one. The clustering is shown in Figure 6.8, where we can see each categorical feature separated by a discontinuous line and each category from each categorical feature represented by a circle. The two clusters are distinguished by the coloring with dark grey and light grey circles. For instance, the categorical feature "property" originally had four categories, namely, "real estate", "building society savings agreement/life insurance", "car or other" and "unknown/no property". As we will see later, the three first categories, colored in dark grey, are those indicating the positive class, and will be grouped into one single cluster, against the category indicating the negative class, namely "unknown/no property".

In this chapter, four strategies to build the CLSVM classifier are proposed using different mathematical programming formulations. The first strategy proposed solves the SVM$^{\mathrm{O}}$ as initial step. Then, categories are clustered using the SVM$^{\mathrm{O}}$ scores and the CLSVM classifier consists of building an SVM classifier using the clustered values. For the second strategy a Mixed Integer Nonlinear Programming (MINLP) formulation of the same type as the SVM formulation is proposed, but in this case defining a score for each cluster of each categorical feature. The second strategy is based on solving the continuous relaxation of this MINLP formulation, a Quadratically Constrained Quadratic Programming (QCQP) formulation to find a clustering, and the CLSVM classifier consists of building again an SVM classifier using the clustered dataset. The third and fourth strategies are based on a Mixed Integer Quadratic Programming (MIQP) formulation derived from the MINLP formulation using the *big M* modeling trick to reformulate the nonlinear terms in the feasible region. The third strategy works similarly to the second one, but solves the continuous relaxation of the MIQP. The fourth strategy solves the MIQP formulation itself and obtains the clustering and the classifier at once.

In the computational results, the four strategies are compared against the SVM$^{\mathrm{O}}$ in ten real-life datasets using two quality criteria defined in Section 4.3, namely accuracy and categorical complexity of the classifier for the categorical data, see equation (4.8). We conclude from our experiments that the CLSVM achieves a comparable or even better accuracy than the SVM$^{\mathrm{O}}$ in nine of the ten datasets tested. In addition, the CLSVM methodology provides a reduction on the complexity of the classifier for the categorical data, while the SVM$^{\mathrm{O}}$ uses more dummy features for all the strategies and for all ten datasets.

The remainder of this chapter is organized as follows. In Section 6.2 we introduce the CLSVM methodology together with two mathematical programming formulations. Two theoretical results relevant to the formulations are presented. In Section 6.3 the four CLSVM strategies are presented. Section 6.4 is devoted to the computational experience, where the CLSVM classifier and the SVM$^{\mathrm{O}}$ classifier are compared using ten datasets. Finally, Section 6.5 contains a brief summary, final conclusions and some lines for future research.

## 6.2 The CLSVM methodology

In this section the CLSVM methodology is introduced. An MINLP formulation is presented for building the CLSVM classifier and some theoretical results for the formulation are stated and proved. Then, an MIQP formulation is derived from the MINLP one, using the *big M* modeling trick to reformulate the nonlinear terms in the feasible region. The theoretical results also hold for this formulation.

The CLSVM methodology is based on the SVM formulation, but takes into account the way categorical features are handled in the SVM (and other linear classifiers): splitting each feature into a series of 0-1 dummy features, the classifier assigns one score to each dummy feature, and thus to each value of the categorical feature. Instead, the CLSVM methodology lets categories cluster around their peers and builds an SVM classifier using the clustered dataset, which may reduce the number of relevant features. We will say that category $k$ from categorical feature $j'$ is relevant to the classifier if $\omega'_{j',k} \neq 0$. Let us focus now on categorical features. If a category is relevant to the classifier, we will say that category $k$ from feature $j'$ points towards the positive class if the score associated to the category is positive, i.e., if $\omega'_{j',k} > 0$. Analogously, if $\omega'_{j',k} < 0$ we will say that category $k$ from feature $j'$ points towards the negative class. The fact that a category points towards the positive (or negative) class means that it contributes to classify objects in the positive (or negative) class respectively, i.e., contributes to make $sign((\omega)^\top x_i + (\omega')^\top x'_i + b)$ equal to $+1$ $(-1)$.

First, we remind the standard SVM [42, 56, 152, 153], formulated as the following Quadratic Programming (QP) formulation with linear constraints:

$$\min_{\omega,\omega',b,\xi} \sum_{j=1}^{J}\sum_{k=1}^{K_j} \frac{(\omega_{j,k})^2}{2} + \sum_{j'=1}^{J'} \frac{(\omega'_{j'})^2}{2} + \frac{C}{n}\sum_{i=1}^{n} \xi_i \tag{6.1}$$

s.t. (SVM)

$$y_i(\sum_{j=1}^{J}\sum_{k=1}^{K_j} \omega_{j,k}\, x_{i,j,k} + (\omega')^\top x'_i + b) \geq 1 - \xi_i \qquad \forall i = 1,\ldots,n \tag{6.2}$$

$$\xi_i \geq 0 \qquad \forall i = 1,\ldots,n \tag{6.3}$$

$$\omega \in \mathbb{R}^{\sum_{j=1}^{J} K_j} \tag{6.4}$$

$$\omega' \in \mathbb{R}^{J'} \tag{6.5}$$

$$b \in \mathbb{R}, \tag{6.6}$$

where $\xi = (\xi_i)$ denotes the vector of deviation variables and $C$ is the tradeoff parameter that calls for tuning, see Section 4.4.

The methodology proposed in this chapter, the CLSVM, receives as input a dataset containing categorical and occasionally continuous features. We will denote by $L_{j'}$ the number of clusters in which the $K_{j'}$ dummies of categorical feature $j'$ are clustered. As a first step, the CLSVM performs a clustering for each categorical feature, defined by an assignment vector $z^*$, where $z^*_{j',k,\ell}$ is equal to 1 if category $k$ from feature $j'$ is assigned to the $\ell$-*th* cluster and 0 otherwise, for $j' = 1,\ldots,J'$, $k = 1,\ldots,K_{j'}$, $\ell = 1,\ldots,L_{j'}$. Then, the dataset is clustered according to $z^*$, see Figure 6.1, and an SVM-type classifier is constructed for the clustered dataset, given by $(\omega)^\top x + (\bar{\omega})^\top \bar{x} + b = 0$.

For categorical feature $j'$, the component $\bar{\omega}_{j',\ell}$ denotes the score for the $\ell$-*th* cluster, $j' = 1, \ldots, J',\ \ell = 1, \ldots, L_{j'}$. The pseudocode of the CLSVM methodology can be found in Figure 6.2. To avoid symmetry between clustering solutions, the first category of each categorical feature is always assigned to its first cluster.

For each $i \in \Omega$:

**Step 1.** Input:

- original object $(x_i, x_i', y_i)$, $x_i \in \mathbb{R}^J$, $x_i' \in \{0,1\}^{\sum_{j'=1}^{J'} K_{j'}}$.
- assignment vector $z^* \in \{0,1\}^{\sum_{j'=1}^{J'} L_{j'} K_{j'}}$
  where $\sum_{\ell=1}^{L_{j'}} z^*_{j',k,\ell} = 1,\ \ \forall j' = 1, \ldots, J'; \forall k = 1, \ldots, K_{j'}$.

**Step 2.** Output:

- clustered object $(x_i, \bar{x}_i, y_i)$, $x_i \in \mathbb{R}^J$, $\bar{x}_i \in \{0,1\}^{\sum_{j'=1}^{J'} L_{j'}}$
  where $\bar{x}_i = (\bar{x}_{i,1,1}, \ldots, \bar{x}_{i,J',L_{J'}})$ with $\bar{x}_{i,j',\ell} = \sum_{k=1}^{K_{j'}} z^*_{j',k,\ell} x'_{i,j',k}$.

**Figure 6.1:** Pseudocode for the clustered dataset defined by the assignment vector $z^*$.

Given a dataset $\Omega$:

**Step 1.** Find the assignment vector $z^*$, defining a clustering for the categorical features.

**Step 2.** Obtain the clustered dataset $\bar{\Omega}$ as in Figure 6.1.

**Step 3.** Find the *CLSVM* classifier for $\bar{\Omega}$, $(\omega)^\top x + (\bar{\omega})^\top \bar{x} + b = 0$.

**Figure 6.2:** Pseudocode for the CLSVM methodology.

### 6.2.1 Formulations for the CLSVM

In this section two different mathematical programming formulations are proposed for the CLSVM methodology, an MINLP and an MIQP formulations. The MIQP formulation is derived from the MINLP formulation using the *big M* modeling trick to reformulate the nonlinear terms in the feasible region.

First, we introduce the Cluster (CL) formulation, an MINLP formulation with nonlinear constraints and 0-1 decision variables. This formulation aims at finding a classifier, but at the same time clustering categorical feature $j'$ into $L_{j'}$ clusters, for each $j' = 1, \ldots, J'$. The CL is formulated as follows:

$$\min_{\omega,\bar{\omega},b,\xi,z} \sum_{j=1}^{J} \frac{(\omega_j)^2}{2} + \sum_{j'=1}^{J'} \sum_{\ell=1}^{L_{j'}} \frac{(\bar{\omega}_{j',\ell})^2}{2} + \frac{C}{n} \sum_{i=1}^{n} \xi_i \tag{6.7}$$

s.t. (CL)

$$y_i \left( (\omega)^\top x_i + \sum_{j'=1}^{J'} \sum_{\ell=1}^{L_{j'}} \bar{\omega}_{j',\ell} \sum_{k=1}^{K_{j'}} z_{j',k,\ell}\, x'_{i,j',k} + b \right) \geq 1 - \xi_i \qquad \forall i = 1, \ldots, n \tag{6.8}$$

$$\sum_{\ell=1}^{L_{j'}} z_{j',k,\ell} = 1 \qquad \forall j' = 1, \ldots, J'; \forall k = 1, \ldots, K_{j'} \tag{6.9}$$

$$\xi_i \geq 0 \qquad \forall i = 1, \ldots, n \tag{6.10}$$

$$z \in \{0,1\}^{\sum_{j'=1}^{J'} L_{j'} K_{j'}} \tag{6.11}$$

$$\bar{\omega} \in \mathbb{R}^{\sum_{j'=1}^{J'} L_{j'}} \tag{6.12}$$

$$\omega \in \mathbb{R}^{J} \tag{6.13}$$

$$b \in \mathbb{R}. \tag{6.14}$$

This formulation resembles the SVM formulation (6.1)-(6.6), and we will discuss their main differences. Here we have a score associated with each feature and each cluster, $\bar{\omega}_{j',\ell}$, as opposed to a score for each category, $\omega'_{j',k}$. With respect to the decision variables, we have $\sum_{j'=1}^{J'} L_{j'} K_{j'}$ new 0-1 variables, the number of components of the assignment vector $z$, but the number of continuous features associated with the score vector decreases from $\sum_{j'=1}^{J'} K_{j'}$ to $\sum_{j'=1}^{J'} L_{j'}$. Constraint (6.8) corresponds to constraint (4.2). Constraint (6.9) ensures that, given a categorical feature, each category is assigned to a unique cluster, which means that there are $\sum_{j'=1}^{J'} K_{j'}$ additional constraints to those in the SVM formulation.

We will say that a categorical feature $j'$ is irrelevant to the classifier if $\bar{\omega}_{j',\ell} = 0,\ \forall \ell = 1, \ldots, L_{j'}$. On the contrary, if the feature is relevant to the classifier, we will say that cluster $\ell$ from feature $j'$ points towards the positive class if the score associated to the cluster is positive, i.e., if $\bar{\omega}_{j',\ell} > 0$. Analogously, if $\bar{\omega}_{j',\ell} < 0$ we will say that cluster $\ell$ from feature $j'$ points towards the negative class. The effective use of the clusters by the CL formulation is stated in the following theoretical results.

**Proposition 6.1.** *For any optimal solution of CL, given a categorical feature $j^*$, if there exists $\ell^*$ such that $z_{j^*,k,\ell^*} = 1\ \forall k = 1, \ldots, K_{j^*}$, then $\bar{\omega}_{j^*,\ell} = 0\ \forall \ell = 1, \ldots, L_{j^*}$.*

*Proof:* The proposition will be proved by contradiction. Let $(\omega, \bar{\omega}, b, \xi, z)$ be an optimal solution of CL for which the desired property does not hold. For the case $\ell = \ell^*$, if $\bar{\omega}_{j^*,\ell^*} \neq 0$, then $(\omega^*, \bar{\omega}^*, b^*, \xi^*, z^*)$ obtained by setting $\bar{\omega}^*_{j^*,\ell^*} = 0$ and $b^* = b + \bar{\omega}_{j^*,\ell^*}$ is a feasible solution for (6.7)-(6.14) and has a smaller objective value, which contradicts the fact that the solution $(\omega, \bar{\omega}, b, \xi, z)$ is optimal.

Now we analyze the case $\ell \neq \ell^*$. If $\bar{\omega}_{j^*,\ell} \neq 0$, then $(\omega^*, \bar{\omega}^*, b^*, \xi^*, z^*)$ obtained by setting $\bar{\omega}^*_{j^*,\ell} = 0$ is a feasible solution for (6.7)-(6.14) and has a smaller objective value, which contradicts the fact that the solution $(\omega, \bar{\omega}, b, \xi, z)$ is optimal. □

From this proposition, we obtain:

**Corollary 6.2.** *Given a categorical feature, if all its categories belong to the same cluster, then the feature is irrelevant to the CLSVM classifier.*

The clustering given in the CL formulation for a categorical feature $j'$ with $L_{j'} = 2$, groups the categories into two clusters. It is easy to see that either the feature is

irrelevant or one of the clusters of the feature points towards the positive class while the other points towards the negative one.

**Proposition 6.3.** *If $L_{j'} = 2$, for a given $j'$, for any optimal solution of CL, it holds that:*

$$\bar{\omega}_{j',1} \cdot \bar{\omega}_{j',2} \leq 0. \tag{6.15}$$

*Proof:* The proposition will be proved by contradiction. Let $(\omega, \bar{\omega}, b, \xi, z)$ be an optimal solution of CL for which the desired property does not hold, i.e., $\bar{\omega}_{j',1} \cdot \bar{\omega}_{j',2} > 0$. Then $(\omega^*, \bar{\omega}^*, b^*, \xi^*, z^*)$ obtained by setting $\bar{\omega}^*_{j',1} = \frac{\bar{\omega}_{j',1} - \bar{\omega}_{j',2}}{2}$, $\bar{\omega}^*_{j',2} = \frac{\bar{\omega}_{j',2} - \bar{\omega}_{j',1}}{2}$ and $b^* = b + \frac{\bar{\omega}_{j',1} + \bar{\omega}_{j',2}}{2}$ satisfies (6.15), is a feasible solution for (6.7)-(6.14) and has a smaller objective value, which contradicts the fact that the solution $(\omega, \bar{\omega}, b, \xi, z)$ is optimal. □

Figure 6.8 illustrates the applicability of Proposition 6.3 for the dataset `german`, where the clustering gives the additional information of which cluster points towards the positive class or the negative class. We have assigned a dark gray coloring to clusters in which $\bar{\omega}_{j',\ell} > 0$ in the CLSVM classifier, and therefore, those clusters point towards the positive class; similarly, a light gray coloring is assigned to clusters in which $\bar{\omega}_{j',\ell} < 0$ in the CLSVM classifier, and therefore, those clusters point towards the negative class. For the four categories of feature "property", the two clusters are given by {"real estate", "building society savings agreement/life insurance", "car or other"} and {"unknown/no property"}. The categories of the first cluster point towards the positive class, while the category "unknown/no property" points towards the negative class.

Nonconvex nonlinear constraints such as (6.8) are known to be computationally difficult to deal with, e.g. [147]. Therefore, one may want to reformulate constraint (6.8) from the MINLP formulation in order to obtain an MIQP formulation where the nonlinear term of the product of variables $\bar{\omega}_{j',\ell} \sum_{k=1}^{K_{j'}} z_{j',k,\ell}\, x'_{i,j',k}$ in constraint (6.8) is reformulated by introducing new constraints, often indicated as *big M* or indicator constraints. This implies adding $\sum_{j'=1}^{J'} L_{j'} K_{j'}$ continuous variables, $\tilde{\omega}_{j',k,\ell}$, $j' = 1, \ldots, J'$, $k = 1, \ldots, K_{j'}$, $\ell = 1, \ldots, L_{j'}$, yielding

$$\min_{\omega,\bar{\omega},\tilde{\omega},b,\xi,z} \sum_{j=1}^{J} \frac{(\omega_j)^2}{2} + \sum_{j'=1}^{J'} \sum_{\ell=1}^{L_{j'}} \frac{(\bar{\omega}_{j',\ell})^2}{2} + \frac{C}{n} \sum_{i=1}^{n} \xi_i \tag{6.16}$$

s.t. (CL-bigM)

$$y_i \left( (\omega)^\top x_i + \sum_{j'=1}^{J'} \sum_{\ell=1}^{L_{j'}} \tilde{\omega}_{j',k(i),\ell} + b \right) \geq 1 - \xi_i \qquad \forall i = 1,\dots,n \tag{6.17}$$

$$\sum_{\ell=1}^{L_{j'}} z_{j',k,\ell} = 1 \qquad \forall k = 1,\dots,K_{j'},\ \forall j' = 1,\dots,J' \tag{6.18}$$

$$\tilde{\omega}_{j',k,\ell} \leq \bar{\omega}_{j',\ell} + M(1 - z_{j',k,\ell}) \qquad \forall k = 1,\dots,K_{j'},\ \forall \ell = 1,\dots,L_{j'},\ \forall j' = 1,\dots,J' \tag{6.19}$$

$$\tilde{\omega}_{j',k,\ell} \geq \bar{\omega}_{j',\ell} - M(1 - z_{j',k,\ell}) \qquad \forall k = 1,\dots,K_{j'},\ \forall \ell = 1,\dots,L_{j'},\ \forall j' = 1,\dots,J' \tag{6.20}$$

$$\tilde{\omega}_{j',k,\ell} \leq M\, z_{j',k,\ell} \qquad \forall k = 1,\dots,K_{j'},\ \forall \ell = 1,\dots,L_{j'},\ \forall j' = 1,\dots,J' \tag{6.21}$$

$$\tilde{\omega}_{j',k,\ell} \geq -M\, z_{j',k,\ell} \qquad \forall k = 1,\dots,K_{j'},\ \forall \ell = 1,\dots,L_{j'},\ \forall j' = 1,\dots,J' \tag{6.22}$$

$$\xi_i \geq 0 \qquad \forall i = 1,\dots,n \tag{6.23}$$

$$z \in \{0,1\}^{\sum_{j'=1}^{J'} L_{j'} K_{j'}} \tag{6.24}$$

$$\bar{\omega} \in \mathbb{R}^{\sum_{j'=1}^{J'} L_{j'}} \tag{6.25}$$

$$\omega \in \mathbb{R}^{J} \tag{6.26}$$

$$\tilde{\omega} \in \mathbb{R}^{\sum_{j'=1}^{J'} L_{j'} K_{j'}} \tag{6.27}$$

$$b \in \mathbb{R}. \tag{6.28}$$

We will compare this with the CL formulation. Both objective functions are exactly the same. The difference between the two formulations comes from the constraints, and the addition of $\sum_{j'=1}^{J'} L_{j'} K_{j'}$ new continuous variables. Constraint (6.17) is the corresponding to constraint (6.8). Here, the nonlinear term is replaced with the variable $\tilde{\omega}_{j',k(i),\ell}$, where $k(i)$ identifies the category in which object $i$ falls for feature $j'$. In order to reformulate constraint (6.8) as a collection of linear constraints, it is a very well-known modeling trick to use a 0-1 variable to control if constraint (6.8) is active or not, see [159]. Then, constraint (6.8) is reformulated as linear constraint (6.17), and $4 \cdot \sum_{j'=1}^{J'} L_{j'} K_{j'}$ more constraints are needed for the reformulation, (6.19)-(6.22), the so-called *big M* constraints, where $M > 0$ is a big enough constant.

Please note that Proposition 6.1, Proposition 6.3 and Corollary 6.2 also hold for the CL-bigM formulation, as it is a valid reformulation of the CL formulation.

## 6.3 Strategies for the CLSVM

In this section four different strategies are proposed to obtain the CLSVM classifier. The first, and natural, way to define a CLSVM classifier is by clustering the categories using the scores of the original SVM, the SVM$^{\mathrm{O}}$. This is a cheap strategy but underperforming in some cases in terms of accuracy, as we will see in the computational section. Three alternative strategies are proposed based on the two mathematical programming formulations introduced in Section 6.2, the CL and the CL-bigM.

In the remainder of this section, when describing the strategies, we will explain how to obtain the partial solution $(\omega, \bar{\omega}, b)$, which determines the CLSVM classifier, and the

assignment vector $z^*$. Then, the assignment vector $z^*$ performs a clustering for the original dataset, obtaining a clustered dataset, as shown in Figure 6.1.

The first strategy, the *centroid SVM* ($\text{SVM}^{\text{C}}$) strategy, is based on the $\text{SVM}^{\text{O}}$ scores. As initial step, the $\text{SVM}^{\text{O}}$ classifier is built for the original dataset, then the categories of categorical feature $j$ are clustered into $L_{j'}$ clusters by clustering the $\text{SVM}^{\text{O}}$ scores, for each $j'$. This is done by solving the minimum sum of squares clustering problem (MSSC), [86]. Given a categorical feature $j'$, MSSC clusters all the categories into $L_{j'}$ clusters such that the sum of the squared distance of the score of a category from the centroid of the cluster is minimized. The pseudocode of the MSSC problem can be found in Figure 6.3, where the $j'$ index has been dropped for the sake of clarity, and calligraphic font is used to denote sets, while regular font for their cardinality. After clustering the dataset, the CLSVM classifier builds an SVM classifier using the clustered dataset. The pseudocode of this strategy can be found in Figure 6.4.

Given a vector $\nu \in \mathbb{R}^K$ associated with a categorical feature with $K$ categories,

**Step 1.** Sort the values $\nu_k$, $k = 1, \dots, K$, increasingly, such that:

$$\nu_{(1)} \leq \nu_{(2)} \leq \ldots \leq \nu_{(K)}$$

**Step 2.** Set $\mathscr{I}_{i_0} = \emptyset$
Set $\mathscr{I}_{i_\ell} = \{1, \dots, i_\ell\}$
Set $\mathscr{I}_{i_L} = \{1, \dots, K\}$
Set $F(\mathscr{I}_{i_\ell}) = \sum_{i \in \mathscr{I}_{i_\ell}} \left( \nu_{(i)} - \frac{1}{I_{i_\ell}} \sum_{h \in \mathscr{I}_{i_\ell}} \nu_{(h)} \right)^2$
Solve

$$\min_{i_1, \dots, i_{L-1}} \sum_{\ell=1}^{L} F(\mathscr{I}_{i_\ell} \setminus \mathscr{I}_{i_{\ell-1}})$$

**Step 3.** Return the assignment vector $z^*$ defined by the $L$ clusters, determined by $\mathscr{I}_{i_1}, \dots, \mathscr{I}_{i_{L-1}}$.

**Figure 6.3:** Pseudocode for the MSSC problem.

For each $C$,

**Step 1.** Solve the *SVM*$^{\text{O}}$ and obtain the (partial) optimal solution $\omega'$.

**Step 2.** For each $j'$, cluster the $K_{j'}$ categories of feature $j'$ into $L_{j'}$ clusters solving the MSSC problem for $\nu = \omega'_{j'\cdot}$, obtaining the components from the assignment vector $z^*_{j'\cdot}$.

**Step 3.** Solve the *SVM* formulation with clustered data defined by $z^*$, and return this as the CLSVM classifier.

Choose the best $C$ using the CLSVM classifier.

**Figure 6.4:** Pseudocode for the $\text{SVM}^{\text{C}}$ strategy.

The second strategy, the *CL randomized rounding* ($\text{CL}^{\text{RR}}$) strategy, performs a randomized rounding, [132], to the fractional assignment vector returned by the continuous relaxation of the CL formulation. This is a QCQP formulation, where constraint (6.11) is relaxed to $z \in [0,1]^{\sum_{j'=1}^{J'} L_{j'} K_{j'}}$. The pseudocode of this reduction strategy can be found in Figure 6.5, where $\texttt{rand}(p)$ is a subroutine of random numbers generation,

returning the value 1 with probability $p$ and 0 otherwise.

For each $C$,

**Step 1.** (i) Solve the continuous relaxation of *CL* and obtain the (partial) optimal solution $z$.

(ii) Set $z^*_{j',k,\ell} = 0 \quad \forall k = 1,\dots,K_{j'}, \forall \ell = 1,\dots,L_{j'}, \forall j' = 1,\dots,J'$
For $j' = 1,\dots,J'$
For $k = 1,\dots,K_{j'}$
Set $\ell = 1$
while $(\ell < L_{j'})$
Set $z^*_{j',k,\ell} = \texttt{rand}(z_{j',k,\ell})$
If $z^*_{j',k,\ell} = 0$, set $\ell = \ell + 1$
Else $\ell = L_{j'}$
end
Set $z^*_{j',k,L_{j'}} = 1 - \sum_{\ell=1}^{L_{j'}-1} z^*_{j',k,\ell}$
end
end

(iii) Return the assignment vector $z^*$.

**Step 2.** Solve the *SVM* formulation with clustered data defined by $z^*$, and return this as the CLSVM classifier.

Choose the best $C$ using the CLSVM classifier

**Figure 6.5:** Pseudocode for the $\text{CL}^{\text{RR}}$ strategy.

The third strategy, the *CL-bigM randomized rounding* ($\text{CLM}^{\text{RR}}$) strategy is based on the randomized rounding of the partial solution of the continuous relaxation of the CL-bigM formulation. It is similar to the $\text{CL}^{\text{RR}}$ strategy, but with the difference that it solves the continuous relaxation of the CL-bigM formulation, where constraint (6.24) is relaxed to $z \in [0,1]^{\sum_{j'=1}^{J'} L_{j'} K_{j'}}$. The pseudocode of this strategy can be found in Figure 6.6.

The last strategy, the *CLM* strategy, is based on the CL-bigM formulation. Instead of solving the continuous relaxation, this strategy solves the CL-bigM formulation or returns the incumbent solution after a given time limit. In this case the incumbent solution gives the clustering and the classifier at once. The pseudocode of this strategy can be found in Figure 6.7. This is the most computationally expensive strategy, as it involves solving an MIQP formulation with *big M* constraints. However, the cost of the strategy is balanced with the computational results, as shown in Section 6.4.

Other strategies are possible and natural, and some were tested. For instance, we tried two strategies based on solving the CL formulation. These strategies solved to optimality the CL formulation or returned the incumbent solution after a given time limit. We tested the strategy for which the incumbent solution gave the clustering and the classifier at once. We also tested another one for which the assignment vector $z^*$ of the incumbent solution was used to cluster the dataset and an SVM was solved to find the classifier. These strategies are however computationally expensive as they involve solving MINLP formulations. The quality of these strategies is not reported in Section 6.4 since they were systematically outperformed by the strategies above.

For each $C$,

**Step 1.** (i) Solve the continuous relaxation of *CL-bigM* and obtain the (partial) optimal solution $z$.

(ii) Set $z^*_{j',k,\ell} = 0 \quad \forall k = 1,\dots,K_{j'}, \forall \ell = 1,\dots,L_{j'}, \forall j' = 1,\dots,J'$
For $j' = 1,\dots,J'$
For $k = 1,\dots,K_{j'}$
Set $\ell = 1$
while $(\ell < L_{j'})$
Set $z^*_{j',k,\ell} = \texttt{rand}(z_{j',k,\ell})$
If $z^*_{j',k,\ell} = 0$, set $\ell = \ell + 1$
Else $\ell = L_{j'}$
end
Set $z^*_{j',k,L_{j'}} = 1 - \sum_{\ell=1}^{L_{j'}-1} z^*_{j',k,\ell}$
end
end

(iii) Return the assignment vector $z^*$.

**Step 2.** Solve the *SVM* formulation with clustered data defined by $z^*$, and return this as the CLSVM classifier..

Choose the best $C$ using the CLSVM classifier

**Figure 6.6:** Pseudocode for the CLM$^{\mathrm{RR}}$ strategy.

For each $C$,

**Step 1.** Solve the *CL-bigM* and obtain the (partial) solution $(\omega, \bar{\omega}, b, z)$, the assignment vector and the classifier at once, and return this as the CLSVM classifier.

Choose the best $C$ using the CLSVM classifier

**Figure 6.7:** Pseudocode for the CLM strategy.

## 6.4 Computational results

In this section we illustrate the quality of the CLSVM methodology compared to the benchmark procedure, the SVM$^{\mathrm{O}}$, in terms of accuracy and complexity of the classifier associated with the categorical features. For the sake of clarity, we will refer to the complexity of the classifier associated with the categorical features as *complexity*. The complexity of the SVM$^{\mathrm{O}}$ classifier is given in equation (4.8) and the complexity of the CLSVM classifier is given by $\frac{card(\{\bar{\omega}_{j',\ell} \neq 0\})}{\sum_{j'=1}^{J'} K_{j'}} \cdot 100\%$. We will show that the CLSVM classifier is competitive against the SVM$^{\mathrm{O}}$ classifier in terms of accuracy and outperforms the SVM$^{\mathrm{O}}$ classifier in terms of complexity.

Our experiments have been conducted on a PC with an Intel Core i7 processor with 16 Gb of RAM for all strategies except for the CL$^{\mathrm{RR}}$ strategy, where the Neos Server is used, [58]. We use the optimization engine `IBM-Cplex` v12.5, [99], for solving the SVM formulation, the CL-bigM formulation and its continuous relaxation, and Ipopt, [155, 58], for the continuous relaxation of CL. We have fixed $M$=1000 on the CL-bigM formulation. Although most optimization problems are solved to optimality in a few

seconds, for the CL-bigM formulation the time limit is set to 300 seconds.

Following the usual approach, the parameter $C$ is tuned by inspecting a grid of the form $\frac{C}{n} \in \{10^{-6}, \ldots, 10^{6}\}$, see [42]. The quality in terms of accuracy and complexity of the CLSVM methodology is illustrated using ten real-life datasets from the UCI repository, [23]. A description of these datasets can be found in Tables 4.1 and 6.1. The size of the training set ($n$) is set as the closest $10^2$ multiple to $|\Omega|/3$ setting 5000 as the maximum in order to have running times below reasonable values, see the sixth column of Table 4.1. The second and third columns of Table 6.1 report the number of categorical and continuous features, respectively. Finally, the total number of categories and the number of categories per feature are reported. Computational results are presented in Section 6.4.1.

### 6.4.1 Results

In this section we compare the quality of the four strategies proposed to build the CLSVM classifier against that of the $\text{SVM}^{\text{O}}$ classifier in terms of accuracy and complexity of the classifier. When, for a given criterion, the difference in quality of two classifiers is below 1 percentage point (p.p.), we will say that both classifiers are comparable under such criterion.

Tables 6.2-6.5 report the mean validation accuracy as well as the standard deviation and the median across the ten reshuffles for the accuracy and complexity, where for each dataset and each criterion, we underline the best results accross all the strategies and the benchmark procedure. Results for the benchmark procedure, $\text{SVM}^{\text{O}}$, are reported in Table 6.2, for the $\text{SVM}^{\text{C}}$ strategy in Table 6.3, for the $\text{CL}^{\text{RR}}$ strategy in Table 6.4, and for the $\text{CLM}^{\text{RR}}$ and the CLM strategies in Table 6.5. The following conclusions can be drawn from our computational results for the mean values, but similar conclusions are derived if median values are analyzed.

We start with the accuracy. For seven datasets (`census income`, `mushrooms`, `coil 2000`, `abalone`, `molecular`, `solar-c`, `german`), at least one of the strategies is comparable to the $\text{SVM}^{\text{O}}$. For two datasets the $\text{SVM}^{\text{O}}$ is outperformed, by two strategies in `adult` and by one strategy in `australian`. In `adult`, the $\text{SVM}^{\text{C}}$ strategy and the $\text{CLM}^{\text{RR}}$ strategy outperform the $\text{SVM}^{\text{O}}$ by 3.65 p.p. and 4.18 p.p. respectively. In `australian`, the CLM strategy outperforms the $\text{SVM}^{\text{O}}$ in 1.26 p.p. For one dataset, `careval`, the $\text{SVM}^{\text{O}}$ achieves the best accuracy, where the difference with the CLSVM classifier is between 2.57 p.p., with the CLM strategy, and 13.94 p.p., with the $\text{SVM}^{\text{C}}$ strategy.

We now focus on the second criterion, namely, complexity. All strategies show a dramatic reduction on complexity of the classifier with respect to the categorical features. The minimum improvement over the $\text{SVM}^{\text{O}}$ is for the `coil 2000` dataset, of 8.12 p.p. For the remaining datasets, all strategies proposed for the CLSVM methodology outperform the $\text{SVM}^{\text{O}}$ at least by 30 p.p. For the first six datasets, (`census income`, `adult`, `mushrooms`, `coil 2000`, `abalone`, `molecular`), the CLM strategy achieves the lowest complexity. For the last four datasets (`careval`, `solar-c`, `german`, `australian`), the $\text{CLM}^{\text{RR}}$ strategy achieves the lowest complexity, reaching an improvement of 85.25 p.p. over the $\text{SVM}^{\text{O}}$.

In summary, the four strategies proposed for the CLSVM methodology are competitive against the $\text{SVM}^{\text{O}}$ in terms of accuracy, and clearly dominate in terms of complexity of the classifier. The $\text{SVM}^{\text{C}}$ and $\text{CLM}^{\text{RR}}$ strategies, have a computational

cost comparable to that of the benchmark procedure, $SVM^O$, as they only involve solving QP formulations. Then, for a small increase in the computational cost, one can obtain a more stable strategy, the $CL^{RR}$, solving QCQP formulations. Although the CLM strategy is the most computationally expensive strategy, as it involves solving difficult MIQP formulations with *big M* constraints, its cost is balanced with the computational results, as it is the strategy performing best accuracy results in three datasets (`careval`, `german`, `australian`) and best complexity results in six datasets (`census income`, `adult`, `mushrooms`, `coil 2000`, `abalone`, `molecular`).

As shown in Table 6.5, the quality of the CLM strategy suggests it could be improved for datasets with a large number of categories, such as `molecular`. Recall that to obtain running times below reasonable values, the time limit for this strategy is set to 300 seconds. Increasing the time limit to 3600 seconds for `molecular`, changes the mean accuracy from 51.92% to 93.70% and the median from 51.92% to 93.74%, which makes the CLM comparable to the $SVM^O$ in terms of accuracy for `molecular`. Therefore, increasing the running time may be an alternative for the CLM strategy when dealing with a large number of features.

## 6.5 Conclusions

In this chapter the CLSVM methodology is proposed, based on the SVM and performing a clustering for categorical features, letting categories cluster around their peers and building an SVM classifier using the clustered dataset. Four strategies are presented to build the CLSVM classifier by means of QCQP, MIQP and QP formulations. When using two clusters, the CLSVM classifier has a comparable accuracy to the $SVM^O$ classifier, in seven of the ten benchmark datasets. In the remaining three datasets, the CLSVM classifier outperforms the $SVM^O$ classifier in two datasets, and is outperformed in the other one. In terms of complexity of the classifier with respect to the categorical features, the CLSVM methodology shows a dramatic improvement over the $SVM^O$.

There are several interesting directions to extend the CLSVM methodology. First, knowledge domain [38, 114] can be incorporated into the methodology to build a set of comprehensible rules to facilitate interpretability. This can be done by adding new constraints to the formulations. For each categorical feature, the CLSVM creates a given number of clusters, hence, constraints implying that two categories must belong to the same cluster, or fixing the maximum (or minimum) number of categories that compose a cluster, can be easily added. Other natural constraints could contribute to interpretability. For instance, if categories are countries, one may want to impose some countries to be in the same cluster based on their geographic location.

Second, a sequential methodology could be designed to handle datasets containing a large number of categorical features. This can be done by running a CLSVM model for each feature, fixing a clustering for the feature, and then iteratively repeating the process for the remaining features. Different ways of choosing the order of features for the iterative process require extra analysis; for instance, one can choose the feature for which the CLSVM classifier has the best accuracy.

Third, the CLSVM methodology can be extended to handle continuous features as well. As the CLSVM aims at reducing the complexity of the classifier in the presence of categorical features, we have focused on benchmark datasets composed by categorical features and occasionally continuous features. However, for any dataset, a combined

methodology could be performed in order to transform continuous features into categorical ones, by applying the techniques from [36, 138], either binarizing or discretizing continuous features and then applying the CLSVM methodology. This extension deserves further study and testing.

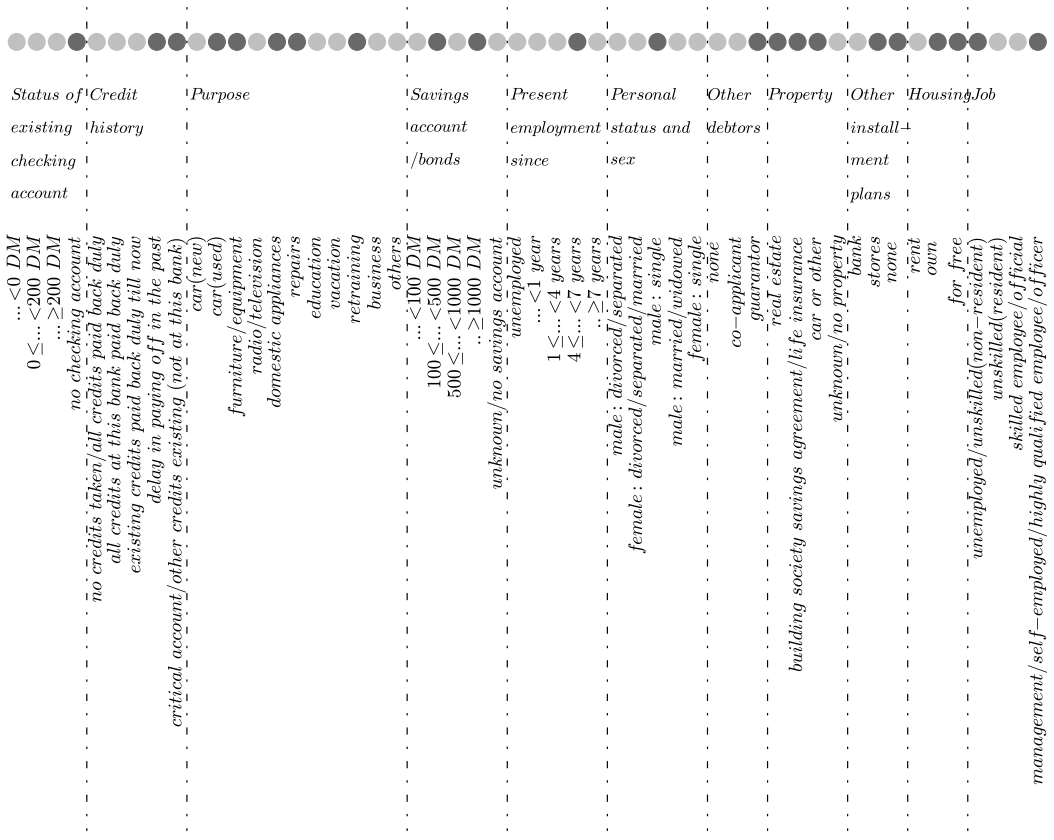


Figure 6.8: The CLSVM methodology for the `german` dataset.

Table 6.1: Datasets.

| Name | $J$ | $J'$ | $\sum_{j=1}^{J} K_j$ | $K_j$ |
|---|---|---|---|---|
| census income | 31 | 9 | 491 | 9,52,47,17,3,7,24,15,5,10,3,6,8,6,6, 50,38,8,9,8,9,3,3,5,42,42,42,5,3,3,3 |
| adult | 11 | 3 | 117 | 5,8,5,16,5,7,14,6,5,5,41 |
| mushrooms | 17 | 4 | 111 | 6,4,10,9,4,3,12,4,4,9,9,4,3,8,9,6,7 |
| coil 2000 | 5 | 80 | 77 | 41,6,10,10,10 |
| abalone | 1 | 7 | 3 | 3 |
| molecular | 60 | 0 | 480 | 8,8,8,... |
| careval | 6 | 0 | 21 | 4,4,4,3,3,3 |
| solar-c | 5 | 5 | 23 | 7,6,4,3,3 |
| german | 11 | 9 | 52 | 4,5,11,5,5,5,3,4,3,3,4 |
| australian | 4 | 10 | 29 | 3,14,9,3 |

Table 6.2: Accuracy and complexity results in % for the original SVM ($\text{SVM}^{\text{O}}$).

| *Name* | $\text{SVM}^{\text{O}}$ | | | | | |
|---|---|---|---|---|---|---|
| | Accuracy | | | Complexity | | |
| | mean | std | med | mean | std | med |
| census income | 94.90 | 0.00 | 94.90 | 63.14 | 0.00 | 63.14 |
| adult | 84.57 | 0.22 | 84.63 | 83.93 | 3.26 | 84.62 |
| mushrooms | 100.00 | 0.00 | 100.00 | 71.17 | 0.00 | 71.17 |
| coil 2000 | 100.00 | 0.00 | 100.00 | 1.30 | 0.00 | 1.30 |
| abalone | 79.87 | 1.18 | 79.72 | 100.00 | 0.00 | 100.00 |
| molecular | 94.22 | 0.80 | 94.04 | 57.04 | 3.12 | 58.54 |
| careval | 96.74 | 1.34 | 96.97 | 99.05 | 2.86 | 100.00 |
| solar-c | 83.53 | 1.23 | 83.46 | 52.17 | 34.51 | 69.57 |
| german | 74.60 | 2.71 | 75.66 | 94.62 | 1.88 | 95.19 |
| australian | 84.11 | 3.17 | 84.73 | 86.90 | 4.83 | 89.66 |

Table 6.3: Accuracy and complexity results in % for the $SVM^{C}$ strategy.

| *Name* | $SVM^{C}$ | | | | | |
|---|---|---|---|---|---|---|
| | Accuracy | | | Complexity | | |
| | mean | std | med | mean | std | med |
| census income | 94.85 | 0.00 | 94.85 | 10.18 | 0.00 | 10.18 |
| adult | 88.22 | 2.44 | 89.59 | 13.76 | 2.21 | 13.68 |
| mushrooms | 100.00 | 0.00 | 100.00 | 23.42 | 0.00 | 23.42 |
| coil 2000 | 100.00 | 0.00 | 100.00 | 1.30 | 0.00 | 1.30 |
| abalone | 79.90 | 1.05 | 79.44 | 66.67 | 0.00 | 66.67 |
| molecular | 93.94 | 0.73 | 93.84 | 0.00 | 0.00 | 0.00 |
| careval | 82.80 | 5.15 | 82.95 | 44.76 | 8.57 | 38.10 |
| solar-c | 83.61 | 1.38 | 83.46 | 12.17 | 10.43 | 8.70 |
| german | 74.80 | 2.36 | 75.00 | 42.31 | 0.00 | 42.31 |
| australian | 84.42 | 3.32 | 84.73 | 19.31 | 9.66 | 24.14 |

Table 6.4: Accuracy and complexity results in % for the $CL^{RR}$ strategy.

| *Name* | $CL^{RR}$ | | | | | |
|---|---|---|---|---|---|---|
| | Accuracy | | | Complexity | | |
| | mean | std | med | mean | std | med |
| census income | 94.84 | 0.04 | 94.82 | 8.31 | 0.33 | 8.55 |
| adult | 83.44 | 0.37 | 83.44 | 16.58 | 1.34 | 16.24 |
| mushrooms | 100.00 | 0.00 | 100.00 | 19.28 | 4.56 | 18.02 |
| coil 2000 | 100.00 | 0.00 | 100.00 | 1.30 | 0.00 | 1.30 |
| abalone | 79.86 | 1.02 | 79.44 | 66.67 | 0.00 | 66.67 |
| molecular | 93.40 | 1.28 | 93.53 | 24.87 | 0.19 | 25.00 |
| careval | 92.23 | 1.28 | 92.42 | 41.90 | 12.2 | 38.10 |
| solar-c | 83.83 | 1.08 | 83.46 | 5.22 | 11.14 | 0.00 |
| german | 74.60 | 3.12 | 74.00 | 38.27 | 4.59 | 41.35 |
| australian | 84.53 | 3.12 | 84.73 | 15.17 | 11.03 | 13.79 |

Table 6.5: Accuracy and complexity results in % for the $\text{CLM}^{\text{RR}}$ and the CLM strategies.

| *Name* | $\text{CLM}^{\text{RR}}$ | | | | | | CLM | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Accuracy | | | Complexity | | | Accuracy | | | Complexity | | |
| | mean | std | med | mean | std | med | mean | std | med | mean | std | med |
| census income | 94.40 | 0.04 | 94.37 | 8.55 | 1.62 | 8.35 | 94.37 | 0.00 | 94.37 | 0.00 | 0.00 | 0.00 |
| adult | 88.75 | 2.96 | 89.63 | 10.17 | 4.36 | 8.55 | 85.35 | 3.16 | 83.44 | 9.23 | 3.64 | 9.83 |
| mushrooms | 98.58 | 0.77 | 98.59 | 21.71 | 5.87 | 22.07 | 100.00 | 0.00 | 100.00 | 14.77 | 1.57 | 14.41 |
| coil 2000 | 100.00 | 0.00 | 100.00 | 1.30 | 0.00 | 1.30 | 100.00 | 0.00 | 100.00 | 1.30 | 0.00 | 1.30 |
| abalone | 79.87 | 0.96 | 79.66 | 66.67 | 0.00 | 66.67 | 79.65 | 1.25 | 79.29 | 66.67 | 0.00 | 66.67 |
| molecular | 88.04 | 1.68 | 88.38 | 22.71 | 0.90 | 22.50 | 51.92 | 0.00 | 51.92 | 0.00 | 0.00 | 0.00 |
| careval | 83.94 | 4.91 | 85.41 | 29.52 | 10.82 | 28.57 | 94.17 | 2.84 | 93.37 | 49.52 | 3.81 | 47.62 |
| solar-c | 83.76 | 1.02 | 83.46 | 0.87 | 2.61 | 0.00 | 83.61 | 1.38 | 83.46 | 11.31 | 7.83 | 8.70 |
| german | 72.53 | 3.77 | 71.66 | 36.92 | 4.28 | 36.54 | 75.60 | 3.01 | 76.34 | 42.31 | 0.00 | 42.31 |
| australian | 84.53 | 3.05 | 84.73 | 5.52 | 2.76 | 6.90 | 85.37 | 3.28 | 85.26 | 23.45 | 5.52 | 27.59 |

# Chapter 7

# Heuristic Approaches for Support Vector Machines with the Ramp Loss

SVM with the ramp loss is an interesting classification model due to its outperformance in terms of robustness compared to the SVM in spite of its difficulty, as it has been proved to be an NP-hard problem. This chapter [40] contributes to the literature on SVM because it illustrates the ability to handle datasets of much larger size than those previously addressed in the literature by proposing heuristics that exploit the nature of the SVM with the ramp loss.

## 7.1 Introduction

As stated in Chapter 4, the SVM [56, 152, 153], has proved to be one of the state-of-the-art methods for Supervised Classification. The SVM, in its typical form (4.1)-(4.5), finds the hyperplane $\omega^\top x + b = 0$ by minimizing the sum of the squared $L_2$ norm of the score vector $\omega$ and the so-called *hinge loss* function [42]. This convex loss function yields smooth convex optimization problems, in fact, convex quadratic, which have been addressed in the literature by a collection of competitive algorithms. However, more challenging optimization problems arise when solving the SVM with nonconvex loss functions, see e.g., [52, 125, 144, 161].

In this chapter, we are interested in the SVM with the so-called *ramp loss* function [52, 144]. From the computational perspective, a first attempt to study the SVM with the ramp loss is presented in [109], where the problem is formulated as a Mixed Integer Quadratic Programming (MIQP) problem, and datasets with up to $n = 100$ objects are solved with a commercial software package. In this model, objects are penalized in a different way depending on if they fall inside or outside the margin, i.e., if they fall between $\omega^\top x + b = -1$ and $\omega^\top x + b = 1$, see Figure 4.2. Misclassified objects that fall outside the margin have a fixed loss of 2, while objects that fall inside the margin have a continuous loss between 0 and 2. The state-of-the-art algorithm is given in [33],

where the ramp loss model, (RLM), is formulated as the following MIQP problem

$$\min_{\omega,b,\xi,z} \frac{1}{2}\sum_{j=1}^{d}\omega_j^2 + \frac{C}{n}\left(\sum_{i=1}^{n}\xi_i + 2\sum_{i=1}^{n} z_i\right)$$

s.t. (RLM)

$$\begin{aligned} y_i(\omega^\top x_i + b) &\geq 1 - \xi_i - Mz_i \qquad \forall i = 1,\dots,n \\ 0 \leq \xi_i &\leq 2 \qquad \forall i = 1,\dots,n \\ z &\in \{0,1\}^n \\ \omega &\in \mathbb{R}^d \\ b &\in \mathbb{R}, \end{aligned} \tag{7.1}$$

where $M > 0$ is a big enough constant, $\xi = (\xi_i)$ denotes the vector of deviation variables and $C$ is the tradeoff parameter that calls for tuning, see Section 4.4. For a given object $i$, the binary variable $z_i$ is equal to 1 if object $i$ misclassified outside the margin and 0 otherwise. See [33] for further details on this formulation, called there the SVMIP1(ramp), and [70, 156] for related models. This MIQP formulation has a quadratic term of dimension $d$ and $n$ binary variables, and is therefore challenging from the computational point of view. In [33], datasets with up to 500 objects are solved to optimality. In this chapter, we propose two heuristics for the RLM that can handle datasets of larger size, and compare them to the state-of-the-art algorithm. The first one is based on the continuous relaxation of the RLM, therefore, it is a cheap heuristic (its computation time is comparable to training an SVM). The second heuristic is based on training an SVM on a reduced dataset identified by an integer linear problem. At the expense of higher running times, and as our computational results will illustrate, this procedure behaves much better in terms of classification accuracy than the other two.

The remainder of the chapter is organized as follows. In Section 7.2, the two heuristics and the state-of-the-art algorithm for the RLM are described in more detail. In Section 7.3, we report our computational results using both synthetic and real-life datasets. We end this chapter in Section 7.4 with some conclusions and topics for future research.

## 7.2 Heuristic approaches

In this section, we describe three heuristics, heuristics 1, 2, 3, whose descriptions can be found in Figures 7.2–7.4, respectively. Heuristics 1 and 2 are our proposals while heuristic 3 is the state-of-the-art algorithm with a time limit [33]. Before describing the heuristics, we present in Figure 7.1 a procedure that exploits the nature of the RLM formulation to build feasible solutions starting from a partial solution $(\omega, b)$. In short, the RLM can be written as a two-stage problem, in which we first choose the classifier, defined by $\omega$ and $b$, and then fill in the variables $\xi$ and $z$.

Therefore, in the rest of this section, when describing the three heuristics, we will concentrate on explaining how to obtain the partial solution $(\omega, b)$ defining the classifier. The corresponding $\xi$ and $z$ will be derived using the fill-in procedure in Figure 7.1.

**Step 1.** For each $i$, fill in variables $\xi_i$ and $z_i$ as follows:

**Case I.** If $y_i(\omega^\top x_i + b) > 1$, then set $\xi_i = 0,\ z_i = 0$.

**Case II.** If $-1 \leq y_i(\omega^\top x_i + b) \leq 1$, then set $\xi_i = 1 - y_i(\omega^\top x_i + b),\ z_i = 0$.

**Case III.** If $y_i(\omega^\top x_i + b) < -1$, then set $\xi_i = 0,\ z_i = 1$.

**Step 2.** $(\omega,\ b,\ \xi,\ z)$ is a feasible solution for the RLM.

**Figure 7.1:** Fill-in procedure for the RLM.

We now describe heuristic 1, see Figure 7.2. In heuristic 1, we first construct the continuous relaxation of the RLM, where the integrality constraints (7.1) are replaced by $z \in [0,1]^n$, and return as (partial) solution $(\omega, b)$ to the RLM the optimal classifier of this relaxation. This is a cheap and naïve heuristic, which consists of solving a convex quadratic problem with linear constraints, and our computational results show that it performs well in the datasets tested.

**Step 1.** Solve the continuous relaxation of the RLM, yielding a (partial) solution $(\omega, b)$.

**Step 2.** Fill in $(\omega, b)$ as described in Figure 7.1.

**Figure 7.2:** Description of heuristic 1.

Heuristic 2 is based on the optimization of an integer linear problem, easier to solve than the RLM since neither the quadratic term $\frac{1}{2}\sum_{j=1}^{d}\omega_j^2$ nor the variables $\xi_i$ are present. Let us consider the Linear Separability Problem (LSP), which aims to find the minimum number of objects to be taken off to make the sets $\{x_i,\ y_i = 1\}$ and $\{x_i,\ y_i = -1\}$ linearly separable. For each object $i$, let us define the binary variable $\alpha_i$ taking the value 1 if object $i$ is removed, and 0 otherwise. Now, the LSP can be formulated as:

$$\min_{\omega,b,\alpha} \quad \sum_{i=1}^{n} \alpha_i$$

s.t. (LSP)

$$\begin{aligned} y_i(\omega^\top x_i + b) &\geq 1 - M\alpha_i \qquad \forall i = 1,\dots,n \\ \alpha &\in \{0,1\}^n \\ \omega &\in \mathbb{R}^d \\ b &\in \mathbb{R}, \end{aligned}$$

where $M$ is a big enough constant.

Heuristic 2 works as shown in Figure 7.3. First, a solution to the LSP is used to

define a reduced set, in which only objects with $\alpha_i = 0$ are considered. Second, an SVM is trained on the reduced set, yielding a (partial) solution $(\omega, b)$ to the RLM. Third, $(\omega, b)$ is used as initial solution in a branch and bound ($B\&B$) procedure for the RLM, which is truncated by a time limit.

Run for $tlim_2$ seconds:

**Step 1.** Solve the LSP and let $(\omega', b', \alpha')$ be the solution vector obtained.

**Step 2.** The solution to the LSP is used to define a reduced set, in which only objects with $\alpha'_i = 0$ are considered.

**Step 3.** An SVM is trained on the reduced set, yielding a (partial) solution $(\omega, b)$ to the RLM.

**Step 4.** Fill in $(\omega, b)$ as described in Figure 7.1.

**Step 5.** $(\omega, b, \xi, z)$ is used as initial solution in a $B\&B$ procedure.

**Figure 7.3:** Description of heuristic 2.

To end, we describe heuristic 3, see Figure 7.4. This heuristic is based on solving the RLM by a $B\&B$ procedure, enhanced with an initial solution, and possibly at each node with a heuristic procedure. In [33], two initial solutions are proposed, the so-called *zero solution* ($ini^{\omega=0}$) and the so-called *zero error solution* ($ini^{z=0}$). Please note that there is a third heuristic solution in [33]. Since it is computationally very expensive, and therefore, impractical for large datasets, it is not considered in our tests. The *zero solution* is derived by setting $\omega = 0$, and $b = 1$ if $card(\{i, y_i = 1\}) > card(\{i, y_i = -1\})$ and $b = -1$ otherwise. The *zero error solution* involves solving a quadratic problem consisting of the RLM with $z_i = 0 \ \ \forall i = 1, \ldots, n$. In [33], a heuristic procedure is applied at each node aiming at improving the best upper bound. This node improvement consists of constructing a feasible solution to the RLM by applying the fill-in procedure in Figure 7.1 to the classifier returned by the continuous relaxation. If the objective value of this new solution improves the best upper bound, this bound will be updated. Note that our heuristic 1 is a cheap heuristic where such a fill-in procedure is only applied at the root node. As for heuristic 2, heuristic 3 is run until a time limit is reached. Note that the three heuristics are matheuristics, [111], since their procedures require solving nontrivial optimization problems.

## 7.3 Computational results

This section is aimed to illustrate the quality of our heuristics, heuristics 1 and 2, with respect to heuristic 3.

Our experiments have been conducted on a PC with an Intel Core i7 processor, 16 Gb of RAM. We use the optimization engine `IBM-Cplex` v12.3, [97], for solving all optimization problems.

To illustrate the capacity of our heuristics to handle datasets of larger sizes than heuristic 3, we use both synthetic and real-life datasets. In [33], heuristic 3 was tested

Run for $tlim_3$ seconds:

**Step 1.** Let $NI \in \{0, 1\}$.

**Step 2.** Let $(\omega, b)$ be an initial (partial) solution.

**Step 3.** Fill in $(\omega, b)$ as described in Figure 7.1.

**Step 4.** $(\omega, b, z, \xi)$ is used as initial solution in a $B\&B$ procedure, where if $NI$=1, heuristic 1 is applied in each node with the aim of improving the current best feasible solution.

**Figure 7.4:** Description of heuristic 3.

on two synthetic datasets obtained using the `TypeA` and `TypeB` generators, and 11 real-life datasets from the UCI repository [23]. We use both the `TypeA` and `TypeB` generators (for $d$ equal to 2, 5 and 10), as well as the dataset `adult`, the only real-life one tested in [33] with a size larger than 5000 objects. In addition, we also report results on three other large UCI datasets, see [23, 45]. A description of these datasets can be found in Table 4.1, for which large sizes are used for the training set.

### 7.3.1 Parameter tuning

As customary in Supervised Learning, the RLM contains parameter $C$ in its formulation which needs to be tuned. Following the usual approach, 4.3, $C$ is tuned by inspecting a grid of 26 values of the form $C = 2^k$, such that $2^{-13} \leq \frac{C}{n} \leq 2^{13}$.

In order to make a fair comparison, overall time limits for both heuristics 2 and 3 should be the same. Heuristic 2 involves solving one LSP plus 26 RLMs, which are aborted when a time limit is exceeded. In our experiments we choose $tlim_{\text{LSP}} =$ 300 seconds for the LSP, and for each RLM $tlim_2$ is chosen as the closest integer to $\dfrac{d + \frac{n}{100}}{5}$. Heuristic 3 involves solving 26 RLMs, which are aborted after $tlim_3 = tlim_2 + \frac{tlim_{\text{LSP}}}{26}$.

### 7.3.2 Accuracy results

As in previous chapters, to obtain sharp estimates for the accuracy of the different heuristics, ten instances are run for each dataset. For each dataset and for each heuristic, Tables 7.1 and 7.2 report the mean validation accuracy across the ten instances, as well as the standard deviation and the median. When the difference in accuracy of two heuristics is below 1 percentage point (p.p.), we will say that both heuristics are comparable. For each dataset, we underline the best results across the heuristics.

We have a whole array of variants of heuristic 3 depending on the initial solution chosen and whether the node improvement procedure is applied. We can see that the simplest implementation of heuristic 3, where we start with the *zero solution* and no node improvement is applied, dominates the rest of the variants in terms of mean and median accuracy, except for `TypeA` (`d` = 2) and `gamma`. In any case, heuristic 2 outperforms any variant of heuristic 3. Therefore, and unless stated, when referring to

heuristic 3 we will use the results mentioned above.

The following conclusions can be drawn from our computational results. In `TypeA`, heuristic 2 outperforms heuristic 3, while heuristic 3 outperforms heuristic 1. The increase in mean accuracy shown by heuristic 2 compared to heuristic 3 is more pronounced for larger values of $d$, being equal to 19.68 p.p. for $d = 10$. A similar behaviour is observed for the median accuracy. In `TypeB`, heuristics 1 and 2 outperform heuristic 3. Heuristic 2 outperforms heuristic 1 in terms of median accuracy, where in terms of mean accuracy the same holds for $d = 2, 10$, while for $d = 5$ the mean accuracies are comparable. A closer look reveals that for larger $d$, any variant of heuristic 3 has a median accuracy of 50%, the one given by the *zero solution*. Heuristics 1 and 2 have a similar accuracy in the real-life datasets, and they clearly outperform heuristic 3 in three of the datasets. For the `gamma` dataset, the best mean accuracy of heuristic 3 is achieved by giving the best among the *zero solution* and the *zero error solution* at the root node, as well as applying the node improvement procedure. For the `gamma` dataset, the increase in mean accuracy (in p.p.) from heuristic 2 to heuristic 3 is then equal to 4.62. For the datasets `adult` and `codrna`, the increase in mean accuracy is equal to 5.41 and 12.67, respectively. Similar dominance is observed when using the median accuracy. For the last real-life dataset, `ijcnn1`, the behaviour in terms of accuracy of the three heuristics is similar. Taking a closer look at this dataset, one can observe that classes are highly unbalanced, see Table 4.1, and therefore the accuracy of the three is very similar to that of the *zero solution*. A finer analysis, taking into account not only the overall accuracy, but also the sensitivity and specificity, reveals significant differences between the heuristics. Indeed, as shown in Table 7.3, while heuristic 3 misclassifies all records in Class +1 (the class in minority), our procedures are slightly better for such class. If, instead of the overall average accuracy, we measure the (weighted) geometric mean of the accuracy in both classes, we see that our procedures clearly outperform heuristic 3. This shows that our heuristics can address, for a given time limit, larger datasets than heuristic 3.

## 7.4 Conclusions

In this chapter we show that a quick heuristic, based on solving the continuous relaxation of the RLM, is competitive against the state-of-the-art algorithm for the SVM with the ramp loss, the RLM. Much better results are obtained with our so-called heuristic 2, which involves solving an integer linear problem and a convex quadratic problem to obtain a good starting solution for the branch and bound procedure.

In order to solve problems of larger size, valid cuts strengthening the formulation would be extremely helpful. In this sense, [33] proposes the so-called geometric cuts, but also mentions that these cuts are only of interest in the trivial case of two-dimensional data. As done in [33], these cuts have not been employed in experiments. New and helpful cuts deserve further study.

Table 7.1: Validation sample accuracy in %.

| *Name* | *heuristic* 1 | | | *heuristic* 2 | | | *heuristic* 3, $NI = 0$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | $ini^{\omega=0}$ | | | $ini^{z=0}$ | | | $ini^{\omega=0}$ & $ini^{z=0}$ | | |
| | *mean* | *std* | *med* | *mean* | *std* | *med* | *mean* | *std* | *med* | *mean* | *std* | *med* | *mean* | *std* | *med* |
| `TypeA` ($d = 2$) | 68.36 | 18.62 | 76.54 | 83.78 | 0.80 | 83.84 | 72.03 | 14.60 | 82.93 | 56.11 | 7.45 | 51.47 | 73.95 | 12.82 | 83.04 |
| `TypeA` ($d = 5$) | 54.86 | 10.83 | 53.18 | 83.60 | 0.85 | 83.50 | 66.33 | 8.70 | 71.46 | 64.41 | 9.28 | 68.18 | 56.91 | 7.75 | 52.52 |
| `TypeA` ($d = 10$) | 51.46 | 10.12 | 50.26 | 79.79 | 10.51 | 82.98 | 60.11 | 6.83 | 62.98 | 57.08 | 7.24 | 55.53 | 56.90 | 7.05 | 55.53 |
| `TypeB` ($d = 2$) | 70.63 | 9.29 | 73.22 | 83.39 | 1.12 | 83.80 | 56.09 | 9.37 | 51.33 | 56.09 | 9.37 | 51.33 | 56.09 | 9.37 | 51.33 |
| `TypeB` ($d = 5$) | 70.71 | 5.41 | 71.98 | 70.50 | 14.44 | 76.88 | 52.08 | 5.14 | 50.00 | 52.08 | 5.14 | 50.00 | 52.08 | 5.14 | 50.00 |
| `TypeB` ($d = 10$) | 63.99 | 8.84 | 63.84 | 77.28 | 9.87 | 81.24 | 51.24 | 3.35 | 50.00 | 51.24 | 3.35 | 50.00 | 51.24 | 3.35 | 50.00 |
| `gamma` | 79.26 | 0.45 | 79.33 | 79.55 | 0.74 | 79.36 | 68.30 | 0.91 | 68.61 | 68.30 | 0.91 | 68.61 | 68.30 | 0.91 | 68.61 |
| `adult` | 84.91 | 0.30 | 84.98 | 84.91 | 0.29 | 84.92 | 79.50 | 1.62 | 78.73 | - | - | - | 79.50 | 1.62 | 78.73 |
| `codrna` | 93.88 | 0.13 | 93.90 | 91.11 | 7.86 | 93.71 | 78.44 | 2.99 | 78.37 | 78.44 | 2.99 | 78.37 | 78.41 | 3.03 | 78.37 |
| `ijcnn1` | 91.35 | 0.37 | 91.33 | 91.93 | 0.44 | 91.97 | 90.17 | 0.36 | 90.03 | 90.17 | 0.36 | 90.03 | 90.17 | 0.36 | 90.03 |

Table 7.2: Validation sample accuracy in %.

| *Name* | *heuristic* 3, $NI = 1$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $ini^{\omega=0}$ | | | $ini^{z=0}$ | | | $ini^{\omega=0}$ & $ini^{z=0}$ | | |
| | *mean* | *std* | *med* | *mean* | *std* | *med* | *mean* | *std* | *med* |
| `TypeA` ($d = 2$) | 54.90 | 5.68 | 51.47 | 56.11 | 7.45 | 51.47 | 54.90 | 5.68 | 51.47 |
| `TypeA` ($d = 5$) | 54.49 | 5.84 | 50.94 | 54.78 | 6.31 | 50.94 | 54.49 | 5.84 | 50.94 |
| `TypeA` ($d = 10$) | 53.10 | 4.75 | 50.00 | 53.28 | 5.04 | 50.00 | 53.10 | 4.75 | 50.00 |
| `TypeB` ($d = 2$) | 55.95 | 9.45 | 50.81 | 56.09 | 9.37 | 51.33 | 55.95 | 9.45 | 50.81 |
| `TypeB` ($d = 5$) | 51.86 | 5.18 | 50.00 | 52.08 | 5.14 | 50.00 | 51.86 | 5.18 | 50.00 |
| `TypeB` ($d = 10$) | 50.01 | 0.02 | 50.00 | 51.24 | 3.35 | 50.00 | 50.01 | 0.02 | 50.00 |
| `gamma` | 64.86 | 0.64 | 64.87 | 74.85 | 0.67 | 75.10 | 74.93 | 0.72 | 75.29 |
| `adult` | 75.81 | 0.28 | 75.90 | 79.30 | 1.54 | 78.73 | 77.93 | 0.42 | 78.09 |
| `codrna` | 66.57 | 0.27 | 66.62 | 78.43 | 3.00 | 78.37 | 66.57 | 0.27 | 66.62 |
| `ijcnn1` | 90.17 | 0.36 | 90.03 | 90.17 | 0.36 | 90.03 | 90.17 | 0.36 | 90.03 |

Table 7.3: Validation sample accuracy in % for `ijcnn1`.

| Heuristic | | Sensitivity | | | Specificity | | |
|---|---|---|---|---|---|---|---|
| | | *mean* | *std* | *med* | *mean* | *std* | *med* |
| 1 | | 18.81 | 2.29 | 18.22 | 99.25 | 0.17 | 99.24 |
| 2 | | 29.66 | 5.11 | 29.80 | 98.71 | 0.55 | 98.92 |
| 3,$NI = 0$ | $ini^{\omega=0}$ | 0.00 | 0.00 | 0.00 | 100.00 | 0.00 | 100.00 |
| | $ini^{z=0}$ | 0.00 | 0.00 | 0.00 | 100.00 | 0.00 | 100.00 |
| | $ini^{\omega=0}$ & $ini^{z=0}$ | 0.00 | 0.00 | 0.00 | 100.00 | 0.00 | 100.00 |
| 3,$NI = 1$ | $ini^{\omega=0}$ | 0.00 | 0.00 | 0.00 | 100.00 | 0.00 | 100.00 |
| | $ini^{z=0}$ | 0.00 | 0.00 | 0.00 | 100.00 | 0.00 | 100.00 |
| | $ini^{\omega=0}$ & $ini^{z=0}$ | 0.00 | 0.00 | 0.00 | 100.00 | 0.00 | 100.00 |

# Chapter 8

# Enhancing the Ramp Loss Model Formulation

This chapter, based on [16], contributes to the literature on SVM and Mathematical Programming because it shows that the ramp loss model formulation addressed in Chapter 7 can be solved much more efficiently by a Mixed Integer Nonlinear Programming reformulation with nonconvex constraints. We challenge the common practice of pursuing a linear reformulation for logical implications. This highlights that algorithmic features that are common in Global Optimization should be reconsidered for Mixed Integer Linear Programming as well.

## 8.1 Introduction

Let us consider the linear inequality

$$\alpha^T x \leq x_0, \tag{8.1}$$

in which both $x \in \mathbb{R}^d$ and $x_0 \in \mathbb{R}$ are variables, while $\alpha$ is a given $d$-dimensional vector. It is a very well-known modeling trick in Mixed Integer Linear Programming (MILP) to use a binary variable to control whether or not linear constraint (8.1) is active or not depending on other parts of the model or at the price of paying a penalty in the objective function. Then, the constraint is reformulated as the following *big M* or *indicator* constraint

$$\alpha^T x \leq x_0 + Mt, \tag{8.2}$$

where $t \in \{0, 1\}$ and $M$ is a big enough value that guarantees that the constraint is inactive if $t = 1$.

Although they provide a clean and flexible modeling tool to deal with nonlinearities and logical implications by staying within the MILP framework, it is well-known that indicator constraints present the drawback of having a weak continuous relaxation. Indeed, depending on the value $M$ and on the value attained by expression "$\alpha^T x - x_0$", very small (fractional) values of $t$ might be sufficient to satisfy the constraint. This leads to quality issues with a continuous relaxation value typically very far away from the mixed integer optimum, but, sometimes even more importantly, might lead to numerical issues, with the MILP solvers being unable to assert if a $t$ value below the

integer tolerance is in fact a true solution.

An alternative for logical implications that has been used in the Mixed Integer Nonlinear Programming (MINLP) literature for decades is provided by *complementary* reformulation

$$(\alpha^T x - x_0)\bar{t} \leq 0, \tag{8.3}$$

where $\bar{t} = 1-t$. However, (8.3) is a nonconvex constraint. Such a source of nonconvexity does not significantly complicate the solution of already nonconvex MINLP models arising, for example, in Chemical Engineering applications. In addition, numerical issues on the choice of the value of $M$ do not appear anymore, at least in the formulation. On the contrary, in the cases where those logical constraints were the only sources of nonconvexity, the common approach has always been that of using constraints (8.2) and MILP techniques.

In this chapter we challenge this common practice of pursuing a linear reformulation for logical implications. We do that by exposing that the Mixed Integer Quadratic Programming (MIQP) formulation of the ramp loss model (RLM) addressed in Chapter 7 can be solved much faster with the Global Optimization (GO) solver `Couenne` [55] using reformulation (8.3) than virtually any state-of-the-art commercial MIQP solver like `IBM-Cplex` [98], `Gurobi` [80] and `Xpress` [162]. This is quite counter-intuitive because, in general, convex MIQP problems admit more efficient solution techniques both in theory and in practice, especially by benefiting of virtually all machinery of MILP solvers.

By computationally analyzing one by one the major components of the GO solver and discussing their influence on the solving approach, we are able to shed some lights on the reasons of this unexpected success and to argue on how MIQP and MILP solvers could benefit from a tighter integration of the same components. One approach for performing such an integration is discussed at the end of the chapter.

The remainder of the chapter is organized as follows. In Section 8.2 we address the RLM, discussed in Chapter 7, the application we use as an example. In Section 8.3 we show the initial set of surprising computational results. In Section 8.4 we discuss why those results are surprising while in Section 8.5 we carefully analyze the reasons of the success of `Couenne` versus `IBM-Cplex`. In Section 8.6 we present an approach to enhance `IBM-Cplex` trying to mimic `Couenne`'s behaviour. Finally, some conclusions and future research are drawn in Section 8.7.

## 8.2 SVM with the ramp loss: An MIQP formulation

In this chapter, we are interested in the SVM with the ramp loss, addressed in Chapter 7. Let us remind the MIQP formulation of the RLM:

$$\min_{\omega,b,\xi,z} \frac{1}{2}\sum_{j=1}^{d}\omega_j^2 + \frac{C}{n}\left(\sum_{i=1}^{n}\xi_i + 2\sum_{i=1}^{n} z_i\right) \tag{8.4}$$

s.t. (RLM)

$$y_i(\omega^\top x_i + b) \geq 1 - \xi_i - Mz_i \qquad \forall i = 1, \ldots, n \tag{8.5}$$
$$0 \leq \xi_i \leq 2 \qquad \forall i = 1, \ldots, n \tag{8.6}$$
$$z \in \{0,1\}^n \tag{8.7}$$
$$\omega \in \mathbb{R}^d \tag{8.8}$$
$$b \in \mathbb{R}, \tag{8.9}$$

with $M > 0$ a big enough constant.

The appeal of model (8.4)–(8.9) relies in the fact that it can (potentially) be solved by a black-box MIQP solver, e.g., `IBM-Cplex` [98]. More precisely, objective function (8.4) is convex while constraints are linear, thus virtually all the very sophisticated and effective machinery for MILP problems can be applied. However, as stated in Chapter 7, the state-of-the-art algorithm given in [33] is able to solve to optimality only a quite limited number of instances although some problem-specific cutting planes and reductions are used to help the MILP solver, namely, `IBM-Cplex`. Essentially, this difficulty is due to the *big M* constraints (8.5) that make the continuous relaxation of model (8.4)–(8.9) very weak. Branching is effective for small problems but the almost-complete enumeration is very hard to do for instances of serious size. Cutting planes are not likely to solve the problem, although they would be specifically designed to face the *big M* issue, or, more precisely, the disjunctive nature of constraints (8.5).

**The Nonconvex Reformulation.** Motivated by the discussed difficulty of dealing with constraints (8.5), we propose the following alternative nonlinear, nonconvex, formulation of the RLM:

$$\min_{\omega,b,\xi,\bar{z}} \frac{1}{2}\sum_{j=1}^{d}\omega_j^2 + \frac{C}{n}\left(\sum_{i=1}^{n}\xi_i + 2\sum_{i=1}^{n}(1-\bar{z}_i)\right) \tag{8.10}$$

s.t.

$$(y_i(\omega^\top x_i + b) - 1 + \xi_i)\cdot \bar{z}_i \geq 0 \qquad \forall i = 1, \ldots, n \tag{8.11}$$
$$0 \leq \xi_i \leq 2 \qquad \forall i = 1, \ldots, n \tag{8.12}$$
$$\bar{z} \in \{0,1\}^n \tag{8.13}$$
$$\omega \in \mathbb{R}^d \tag{8.14}$$
$$b \in \mathbb{R}. \tag{8.15}$$

Binary variables $\bar{z}$'s are used as well in model (8.10)–(8.15) to disable constraints (8.11), which replace constraints (8.5), but are the complemented version of $z$ variables in the RLM, i.e., $\bar{z}_i = 1 - z_i$. Namely, $\bar{z}_i = 1$ forces the $i$-th constraint to be active, thus allowing a maximum violation for $\xi_i = 2$, while $\bar{z}_i = 0$ disables the constraint in a classical "complementary" way.

Of course, constraints (8.11) are responsible of the nonconvexity of the MINLP model (8.10)–(8.15). However, its continuous version obtained by simply replacing constraints (8.13) with $\bar{z} \in [0,1]^n$ is solved to (local) optimality by the Nonlinear Programming (NLP) solver `Ipopt` [101] providing a mixed binary solution that is very

accurate, and relatively quick to compute. Indeed, it is easy to prove that:

**Theorem 8.1.** *Any local optimal solution of the continuous version of model* (8.10)–(8.15) *is mixed binary.*

*Proof:* The proposition will be proved by contradiction. A local optimal solution $(\omega, b, \xi, \bar{z})$ is feasible. Thus, constraints (8.11) are satisfied. For any $i = 1, \ldots, n$, either $\bar{z}_i = 0$ (integer), or if $\bar{z}_i \in (0, 1)$, there exists an equivalent (still feasible) solution $(\omega, b, \xi, \bar{z}_1, \ldots, \bar{z}_{i-1}, 1, \bar{z}_{i+1}, \ldots, \bar{z}_n)$ with $\bar{z}_i = 1$ having smaller objective function. □

Theorem 8.1 above implies that the global optimal solution is mixed binary as well, thus solving the continuous version of the problem with a GO solver like `Couenne` [55] solves the overall problem to optimality. On the other hand, `Couenne` is an MINLP solver, hence it can handle integrality constraints on (a subset of) the variables. Thus, in the results presented in the next section we have retained integrality on variables $\bar{z}$'s.

## 8.3 A raw set of computational results

In Chapter 7, we focused on the robustness of the RLM as a Supervised Classification model. All experiments in Chapter 7 were performed following the usual approach (see Figure 4.3) of tuning the tradeoff parameter $C$ by inspecting a grid of values, i.e., solving (8.4)-(8.9) for different values of $C$. In this chapter, we aim at enhancing the ramp loss model formulation, therefore, we will focus on formulation (8.4)–(8.9) and its reformulation (8.10)–(8.15) for a fixed value of $C$. Hence in the computational experiments of this chapter we are not interested in classification accuracy (as we are always solving the same problem with different formulations), but on the formulations behaviour during optimization, such as computing time to reach the optimal value or number of nodes inspected during the branch and bound tree exploration.

We have performed an exploratory test for the nonconvex MINLP formulation proposed in Section 8.2. We consider only the artificial datasets proposed by [33], to be able to control the dataset and problem size. However, in this chapter we concentrate on a challenging subset of them (23 instances of size $n = 100$, $d = 2$, `TypeB`, see [33] for details) showing the surprising behavior that `Couenne` out-of-the-box performs better than `IBM-Cplex`. Table 8.1 reports the straightforward comparison. Computing times (time) in CPU seconds, number of nodes (nodes), percentage gap of the upper (ub) and lower (lb) bounds are reported, as well as the optimal value of each instance for future reference. A time limit of 1 hour is provided to each run and in case such a limit is reached the entry in the column "time" indicates a "×". For instances solved to optimality gaps are reported as "–".

The results of Table 8.1 are quite straightforward to interpret, with a strict dominance of `Couenne` with respect to `IBM-Cplex`. In the unique instance `Couenne` is unable to solve to optimality (instance 3) the issue is probably that the upper bound is not improved enough, thus being unable to propagate (see next section) and strengthen the formulation. Conversely, `IBM-Cplex` is always able to find the right upper bound (namely, the optimal solution value) but the lower bound value remains far from the optimal value, thus being unable to prove optimality.

Table 8.1 reports detailed numbers only for `IBM-Cplex` but we solved the convex MIQP model (8.4)–(8.9) with the convex MIQP solvers developed extending the

Table 8.1: Computational results for `Couenne` and `IBM-Cplex`. Instances of `TypeB` [33], $n = 100$, time limit of 1 hour, executed on Palmetto cluster [51].

| | | Couenne | | | | IBM-Cplex | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | % gap | | | | % gap | |
| | optimal value | time (sec.) | nodes | ub | lb | time (sec.) | nodes | ub | lb |
| 1 | 157,995.00 | 163.61 | 17,131 | – | – | 3,438.49 | 16,142,440 | – | – |
| 2 | 179,368.00 | 1,475.68 | 181,200 | – | – | × | 12,841,549 | – | 23.61 |
| 3 | 220,674.00 | × | 610,069 | 14.96 | 15.38 | × | 20,070,294 | – | 37.82 |
| 4 | 5,225.99 | 160.85 | 25,946 | – | – | × | 20,809,936 | – | 9.37 |
| 5 | 5,957.08 | 717.20 | 131,878 | – | – | × | 17,105,372 | – | 26.17 |
| 6 | 11,409,600.00 | 1,855.16 | 221,618 | – | – | × | 13,865,833 | – | 22.67 |
| 7 | 11,409,100.00 | 482.19 | 56,710 | – | – | × | 14,619,065 | – | 21.40 |
| 8 | 10,737,700.00 | 491.26 | 55,292 | – | – | × | 13,347,313 | – | 14.59 |
| 9 | 5,705,360.00 | 1,819.42 | 216,831 | – | – | × | 12,257,994 | – | 22.22 |
| 10 | 5,704,800.00 | 807.95 | 89,894 | – | – | × | 13,054,400 | – | 23.13 |
| 11 | 5,369,020.00 | 536.40 | 62,291 | – | – | × | 14,805,943 | – | 12.37 |
| 12 | 2,853,240.00 | 1,618.79 | 196,711 | – | – | × | 12,777,936 | – | 21.97 |
| 13 | 2,852,680.00 | 630.18 | 83,676 | – | – | × | 14,075,300 | – | 23.32 |
| 14 | 2,684,660.00 | 533.77 | 65,219 | – | – | × | 13,994,099 | – | 12.48 |
| 15 | 1,427,170.00 | 2,007.62 | 211,157 | – | – | × | 10,671,225 | – | 23.08 |
| 16 | 1,426,620.00 | 641.05 | 72,617 | – | – | × | 12,984,857 | – | 22.72 |
| 17 | 1,342,480.00 | 728.93 | 73,142 | – | – | × | 12,564,000 | – | 14.11 |
| 18 | 714,142.00 | 1,784.93 | 193,286 | – | – | × | 11,217,844 | – | 23.45 |
| 19 | 713,583.00 | 752.50 | 84,538 | – | – | × | 12,854,704 | – | 22.72 |
| 20 | 671,396.00 | 412.16 | 48,847 | – | – | × | 14,018,831 | – | 12.43 |
| 21 | 357,626.00 | 2,012.62 | 223,702 | – | – | × | 11,727,308 | – | 23.55 |
| 22 | 357,067.00 | 768.73 | 104,773 | – | – | × | 15,482,162 | – | 18.67 |
| 23 | 335,852.00 | 706.39 | 70,941 | – | – | × | 12,258,164 | – | 14.88 |

three major MILP solvers, namely `Gurobi` and `Xpress`, and `IBM-Cplex` itself. The three solvers behave very similarly in the considered instances, thus indicating that the weakness shown in Table 8.1 is structurally associated with solving the *big M* formulation, or, conversely, that solving the nonconvex formulation through a GO solver is effective.

## 8.4 Why are these results surprising?

Although, as anticipated in the introduction, convex MIQP solvers should be more effective than GO ones especially because they can exploit the very sophisticated MILP machinery, one can still argue that a comparison in performance between two different solution methods and computer codes is anyway hard to perform. However, digging into the way in which `Couenne` solves the problem leads to confirm the initial surprise.

**McCormick Linearization.** The first observation is that the way in which constraints (8.11) are managed by `Couenne` is through the classical McCormick linearization [115]. Namely, for $i = 1, \ldots, n$:

1. $\vartheta_i = y_i(\omega^\top x_i + b) - 1 + \xi_i$, with $\vartheta_i^L \leq \vartheta_i \leq \vartheta_i^U$
2. $u_i = \vartheta_i \bar{z}_i$.

Then, the product corresponding to each new variable $u_i$ is linearized as

$$\begin{aligned} u_i &\geq 0 && (8.16)\\ u_i &\geq \vartheta_i^L \, \bar{z}_i && (8.17)\\ u_i &\geq \vartheta_i + \vartheta_i^U \, \bar{z}_i - \vartheta_i^U && (8.18)\\ u_i &\leq \vartheta_i + \vartheta_i^L \, \bar{z}_i - \vartheta_i^L && (8.19)\\ u_i &\leq \vartheta_i^U \, \bar{z}_i && (8.20) \end{aligned}$$

again for $i = 1, \ldots, n$, where (8.16) is precisely (8.11). Essentially, setting $\bar{z}_i = 0$ again deactivates constraint $i$ by simply enforcing the loose $\vartheta_i \in [\vartheta_i^L, \vartheta_i^U]$, where $\vartheta_i^L$ plays the role of the *big M*.

In other words, `Couenne` initially builds a *big M* formulation itself, with the difference that a specific $\vartheta_i^L$ value for each $i$ is computed. Although such an internal computation is not responsible of the higher effectiveness of `Couenne` (typically `Couenne` is more conservative than the static values of $M$ used in the literature and especially by [33]), this is, in practice, not a negligible issue because a safe value of $M$ is not trivial to be determined a priori.

In the next section we will extensively discuss how McCormick inequalities are strengthened, as well as the bounds on $\vartheta$ variables. This will turn out crucial for `Couenne` but, at first this similarity confirms the surprise.

**Branching.** It is well-known that a major component of GO solvers is the iterative tightening of the convex (most of the time linear) relaxation of the nonconvex feasible region by branching on continuous variables (see, e.g., [17]). Another surprising fact here is that the default version of `Couenne` does not take advantage of this possibility and branches on the binary variables $\bar{z}$'s. Because everything is linear (after McCormick linearization) and the objective function convex, as soon as all binaries are fixed the problem is solved.

However, even better performance for `Couenne` could be obtained by branching on continuous variables. Namely, instructed to branch preferably on continuous variables, `Couenne` always selects $\vartheta$ variables, which clearly lead to additional bound tightening with respect to branch on binaries. Indeed, if in a given relaxation we have $\vartheta_i = c$, the two branches $\vartheta_i \leq c$ OR $\vartheta_i \geq c$ propagate as follows: (*i*) if $c < 0$ then $\vartheta_i \leq c$ implies $\bar{z}_i = 0$ OR (*ii*) if $c > 0$ then $\vartheta_i \geq c$ implies $\bar{z}_i = 1$. The results are reported in Table 8.2 and clearly show the computational advantage of this choice. Thus, again it is surprising that the default branching strategy of `Couenne` leads to an improvement over the sophisticated branching framework of `IBM-Cplex`.

**$L_1$ and $L_2$ norms.** A natural question is if the results reported in Table 8.1 are due to the somehow less sophisticated evolution of `IBM-Cplex` in its MIQP extension with respect to the MILP one. In order to answer this question we performed two sets of experiments in which the quadratic part of the objective function was replaced by its $L_1$ and $L_2$ norms on $\omega$, respectively. More precisely, in the case of the $L_1$ norm, the absolute value of $\omega$ is minimized (and linear constraints to deal with the absolute value are added). This results in a pure MILP once the *big M* constraints (8.5) (solved by `IBM-Cplex`) or a nonconvex MINLP with linear objective function (solved by `Couenne`) if constraints (8.11) are used instead. For the $L_2$ norm, instead, we replace the

Table 8.2: Computational results for `Couenne` default and `Couenne` branching emphasis on continuous variables. Instances of `TypeB` [33], $n = 100$, time limit of 1 hour, executed on Palmetto cluster [51].

| | | `Couenne` default | | | | `Couenne` continuous | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | % gap | | | | % gap | |
| | optimal value | time (sec.) | nodes | ub | lb | time (sec.) | nodes | ub | lb |
| 1 | 157,995.00 | 163.61 | 17,131 | – | – | 323.21 | 62,873 | – | – |
| 2 | 179,368.00 | 1,475.68 | 181,200 | – | – | 561.95 | 106,905 | – | – |
| 3 | 220,674.00 | × | 610,069 | 14.96 | 15.38 | 490.29 | 134,758 | – | – |
| 4 | 5,225.99 | 160.85 | 25,946 | – | – | 238.26 | 65,152 | – | – |
| 5 | 5,957.08 | 717.20 | 131,878 | – | – | 535.70 | 142,368 | – | – |
| 6 | 11,409,600.00 | 1,855.16 | 221,618 | – | – | 773.62 | 149,880 | – | – |
| 7 | 11,409,100.00 | 482.19 | 56,710 | – | – | 985.26 | 195,438 | – | – |
| 8 | 10,737,700.00 | 491.26 | 55,292 | – | – | 535.78 | 103,806 | – | – |
| 9 | 5,705,360.00 | 1,819.42 | 216,831 | – | – | 726.18 | 143,234 | – | – |
| 10 | 5,704,800.00 | 807.95 | 89,894 | – | – | 1,031.75 | 172,794 | – | – |
| 11 | 5,369,020.00 | 536.40 | 62,291 | – | – | 546.78 | 109,142 | – | – |
| 12 | 2,853,240.00 | 1,618.79 | 196,711 | – | – | 663.69 | 127,318 | – | – |
| 13 | 2,852,680.00 | 630.18 | 83,676 | – | – | 790.88 | 160,010 | – | – |
| 14 | 2,684,660.00 | 533.77 | 65,219 | – | – | 510.01 | 99,802 | – | – |
| 15 | 1,427,170.00 | 2,007.62 | 211,157 | – | – | 717.69 | 117,670 | – | – |
| 16 | 1,426,620.00 | 641.05 | 72,617 | – | – | 932.44 | 161,835 | – | – |
| 17 | 1,342,480.00 | 728.93 | 73,142 | – | – | 512.60 | 83,890 | – | – |
| 18 | 714,142.00 | 1,784.93 | 193,286 | – | – | 720.15 | 119,761 | – | – |
| 19 | 713,583.00 | 752.50 | 84,538 | – | – | 983.23 | 168,276 | – | – |
| 20 | 671,396.00 | 412.16 | 48,847 | – | – | 449.68 | 86,351 | – | – |
| 21 | 357,626.00 | 2,012.62 | 223,702 | – | – | 661.69 | 110,343 | – | – |
| 22 | 357,067.00 | 768.73 | 104,773 | – | – | 706.15 | 156,464 | – | – |
| 23 | 335,852.00 | 706.39 | 70,941 | – | – | 493.32 | 79,719 | – | – |

quadratic term with a variable $v$ and add the second-order-conic constraint $v^2 \geq ||\omega||_2^2$, thus obtaining a Mixed Integer Second Order Conic Programming (MISOCP) problem. For `Couenne` we simply use the square root of the current quadratic part and then move it as a constraint so as to obtain a Mixed Integer Quadratically Constrained Programming (MIQCP) problem. Again we use `IBM-Cplex` for the MISOCP problem (with constraints (8.5)) and `Couenne` for the nonconvex counterpart (with constraints (8.11)).

In both $L_1$ and $L_2$ versions of the RLM, `Couenne` continues achieving better results than `IBM-Cplex`.

## 8.5 Bound reduction in nonconvex MINLP problems

Bound reduction is a crucial tool in MINLP: it allows one to eliminate portions of the feasible set while guaranteeing that at least one optimal solution is retained. Although its origins can be traced back to Artificial Intelligence [60], it finds wide application in Constraint Programming and in solvers for both Nonlinear Optimization [117] and for MILP problems [4, 142].

Consider a generic optimization problem $\min\{f(x) : x \in X, \ell \leq x \leq u\}$, where $X \subset \mathbb{R}^n$ and $x, \ell, u \in \mathbb{R}^n$. Also, suppose an upper bound $U \in \mathbb{R} \cup \{+\infty\}$ on the objective function value of the optimal solution is available: if $U < +\infty$, then a feasible

solution $x \in X \cap [\ell, u]$ is available such that $f(x) = U$. Bound reduction attempts to find tighter lower bounds $\ell'_i > \ell_i$ and upper bounds $u'_i < u_i$. In general, a good upper bound is often the key to a strong bound reduction.

An ideal bound reduction procedure obtains bounds by exploiting the full problem structure:

$$\begin{array}{rcll} \ell'_i & = & \max & \{x_i : x \in X, \ell \leq x \leq u, f(x) \leq U\}, \\ u'_i & = & \min & \{x_i : x \in X, \ell \leq x \leq u, f(x) \leq U\}. \end{array} \tag{8.21}$$

However, the $2n$ optimization problems above can be as hard as the original MINLP itself, therefore this approach is impractical.

A fast bound reduction procedure, known as *Feasibility Based Bound Tightening* (FBBT), yields new bounds on a variable $x_i$ using bounds on other variables that are linked to $x_i$ through a constraint or the objective function. For instance, the constraint $x_1 x_2 \leq 4$ and the bounds $x_1 \geq 1, x_2 \geq 1$ yield new upper bounds $x_1 \leq 4, x_2 \leq 4$. An example that is closer to our application is the constraint $x_i^2 \leq u$, where $u \geq 0$, which obviously implies $x_i \in [-\sqrt{u}, \sqrt{u}]$.

A specialized version of this procedure applies to affine functions, and is commonly used in MILP solvers [4]. Consider the range constraint:

$$\ell_0 \leq \alpha_0 + \sum_{j=1}^{n} \alpha_j x_j \leq u_0.$$

Define $J^+ = \{j = 1, \ldots, n : \alpha_j > 0\}$ and $J^- = \{j = 1, \ldots, n : \alpha_j < 0\}$. Bounds $\ell_0, u_0$ on the expression imply new (possibly tighter) bounds $\ell'_j, u'_j$ on $x_j, j = 1, \ldots, n : \alpha_j \neq 0$:

$$\begin{array}{rrcl} \forall j : \alpha_j > 0, & \ell'_j & = & \frac{1}{\alpha_j}\left(\ell_0 - \left(\alpha_0 + \sum_{i \in J^+ \setminus \{j\}} \alpha_i u_i + \sum_{i \in J^-} \alpha_i \ell_i\right)\right), \\ & u'_j & = & \frac{1}{\alpha_j}\left(u_0 - \left(\alpha_0 + \sum_{i \in J^+ \setminus \{j\}} \alpha_i \ell_i + \sum_{i \in J^-} \alpha_i u_i\right)\right); \\ \forall j : \alpha_j < 0, & \ell'_j & = & \frac{1}{\alpha_j}\left(u_0 - \left(\alpha_0 + \sum_{i \in J^+} \alpha_i \ell_i + \sum_{i \in J^- \setminus \{j\}} \alpha_i u_i\right)\right), \\ & u'_j & = & \frac{1}{\alpha_j}\left(\ell_0 - \left(\alpha_0 + \sum_{i \in J^+} \alpha_i u_i + \sum_{i \in J^- \setminus \{j\}} \alpha_i \ell_i\right)\right). \end{array} \tag{8.22}$$

### 8.5.1 Applying bound reduction to model (8.10)-(8.15)

`Couenne` is a branch and bound solver for MINLP problems that uses, among others, several bound reduction techniques, including FBBT. At the beginning, `Couenne` runs a greedy rounding procedure to obtain a feasible solution of the problem, and hence an upper bound $U$. Applying FBBT using two rules mentioned above (for affine functions and for the square operator) yields tight bounds on $\omega_i$ at the root node of `Couenne`'s branch and bound tree. Consider the objective function of our problem:

$$\frac{1}{2}\sum_{j=1}^{d} \omega_j^2 + \frac{C}{n}\left(\sum_{i=1}^{n} \xi_i + 2\sum_{i=1}^{n}(1 - \bar{z}_i)\right),$$

which is bounded from above by $U$. Also, note that $\sum_{i=1}^{n} \xi_i + 2\sum_{i=1}^{n}(1 - \bar{z}_i)$ is nonnegative. Since $\sum_{j=1}^{d} \omega_j^2 \geq 0$ and $\sum_{i=1}^{n} \xi_i + 2\sum_{i=1}^{n}(1 - \bar{z}_i) \geq 0$, and also $\xi_i$ and $\bar{z}_i$ are nonnegative, we have

$$\omega_i \in \left[-\sqrt{2U}, \sqrt{2U}\right] \qquad \forall i = 1, \ldots, d.$$

A related observation concerns the constraint of our model. The tighter bounds on the $\omega_i$'s variables, which are initially unbounded per definition of the problem, does not seem to have an influence on the constraints (8.11), where the variable $b$ remains unbounded. This family of nonlinear constraints can be simplified to $\vartheta_i \bar{z}_i \geq 0$, with $\vartheta_i = (y_i(\omega^\top x_i + b) - 1 + \xi_i) \in [-\infty, +\infty]$ and $\bar{z}_i \in \{0, 1\}$, for all $i = 1, \ldots, n$. Due to the infinite bounds, this constraint does not admit a linear relaxation, which is useful for any MINLP solver to obtain a lower bound. When imposing fictitious bounds $[-M, M]$ on $\vartheta_i$, with large enough $M$, one gets the constraint $\vartheta_i \geq M(\bar{z}_i - 1)$, which yields an MILP formulation that is, however, impractical for large $n$.

In these cases, *probing* techniques can be of help. A probing bound reduction algorithm works as follows: impose a fictitious upper bound $\lambda_i \in (\ell_i, u_i)$ on a variable $x_i$, thereby restricting $x_i$ to $[\ell_i, \lambda_i]$. If the restricted problem can be proved (through FBBT, for example) to be infeasible or to have a lower bound that is above a cutoff $U$, then no feasible solution with better objective function value can be found in the restriction. Therefore, the new lower bound $x_i \geq \lambda_i$ is valid.

This procedure can be applied to tighten the upper bound as well, by imposing a fictitious lower bound $\mu_i \in (\ell_i, u_i)$. Although applying it to all variables is time consuming, it is especially useful for unbounded variables. Probing is a common tightening technique in MILP solvers [142] and MINLP [17, 147].

We will now describe the specific reductions that `Couenne` applied to the RLM, and we will computationally show that these reductions are crucial to obtain the results shown in Section 8.4.

### 8.5.2 Strengthening McCormick constraints

The coefficients of McCormick constraints are the lower and upper bounds on the variables involved, hence these constraints can be replaced by stronger ones if tighter bounds are available on the variables involved. `Couenne` does it automatically by means of a cut separator that is called at every branch and bound node, and only adds tighter McCormick cuts if they are violated by the Linear Programming (LP) problem solution available at that node.

Note that McCormick cuts are only useful if both variables $\vartheta_i$ and $\bar{z}_i$ are not fixed, as otherwise the constraint $u_i = \vartheta_i \bar{z}_i$ becomes linear. `Couenne` does not take into account the number of cuts separated up to a certain node, but rather leaves it to the branch and bound manager (`Cbc` in this case) to get rid of the redundant cuts.

While looking for reasons of `Couenne`'s performance, we have run an experiment where McCormick cuts were only added to the initial LP relaxation but excluded from separation at all nodes. The performance worsened dramatically on all instances, which indicates that the bound on the involved variables $u_i, \vartheta_i, \bar{z}_i$ is tightened and should be exploited.

### 8.5.3 Bound tightening

`Couenne` uses several techniques for bound tightening among those mentioned above. In the context of this problem, tightening is based on the following elements:

- the objective function, if a cutoff $U$ is available;
- the definition $u_i = \vartheta_i \bar{z}_i \geq 0$ and related constraint $u_i \geq 0$;

- the definition $\vartheta_i = (y_i(\omega^\top x_i + b) - 1 + \xi_i)$.

Propagation possibly generates new bounds on $\omega, b, \xi$, which are obtained through standard presolve procedures [4]. For the sake of clarity, we add them here and define $\alpha_{ij} = x_{ik}y_i$. Also, denote as $(y_i b)^L$ the lower bound on the expression $y_i b$ (and similar for $(y_i b)^U$). The bound reduction is as follows (clearly the bound is not set if a tighter one is already available):

$$
\begin{array}{lrcl}
\forall k : \alpha_{ik} > 0 & \omega_k^L & = & \frac{1}{\alpha_{ik}} \left( \vartheta_i^L - \sum_{j:\alpha_{ij}<0} \alpha_{ij}\omega_j^L - \sum_{j\neq k:\alpha_{ij}>0} \alpha_{ij}\omega_j^U - (y_i b)^U - \xi^U + 1 \right), \\
& \omega_k^U & = & \frac{1}{\alpha_{ik}} \left( \vartheta_i^U - \sum_{j:\alpha_{ij}<0} \alpha_{ij}\omega_j^U - \sum_{j\neq k:\alpha_{ij}>0} \alpha_{ij}\omega_j^L - (y_i b)^L - \xi^L + 1 \right), \\
\forall k : \alpha_{ik} < 0 & \omega_k^L & = & \frac{1}{\alpha_{ik}} \left( \vartheta_i^U - \sum_{j\neq k:\alpha_{ij}<0} \alpha_{ij}\omega_j^U - \sum_{j:\alpha_{ij}>0} \alpha_{ij}\omega_j^L - (y_i b)^L - \xi^L + 1 \right), \\
& \omega_k^U & = & \frac{1}{\alpha_{ik}} \left( \vartheta_i^L - \sum_{j\neq k:\alpha_{ij}<0} \alpha_{ij}\omega_j^L - \sum_{j:\alpha_{ij}>0} \alpha_{ij}\omega_j^U - (y_i b)^U - \xi^U + 1 \right), \\
\text{if } y_i > 0, & b^L & = & \frac{1}{y_i} \left( \vartheta_i^L - \sum_{j:\alpha_{ij}<0} \alpha_{ij}\omega_j^L - \sum_{j:\alpha_{ij}>0} \alpha_{ij}\omega_j^U - \xi^U + 1 \right), \\
& b^U & = & \frac{1}{y_i} \left( \vartheta_i^U - \sum_{j:\alpha_{ij}<0} \alpha_{ij}\omega_j^U - \sum_{j:\alpha_{ij}>0} \alpha_{ij}\omega_j^L - \xi^L + 1 \right), \\
\text{if } y_i < 0, & b^L & = & \frac{1}{y_i} \left( \vartheta_i^U - \sum_{j:\alpha_{ij}<0} \alpha_{ij}\omega_j^U - \sum_{j:\alpha_{ij}>0} \alpha_{ij}\omega_j^L - \xi^L + 1 \right), \\
& b^U & = & \frac{1}{y_i} \left( \vartheta_i^L - \sum_{j:\alpha_{ij}<0} \alpha_{ij}\omega_j^L - \sum_{j:\alpha_{ij}>0} \alpha_{ij}\omega_j^U - \xi^U + 1 \right), \\
& \xi^L & = & \vartheta_i^L - \sum_{j:\alpha_{ij}<0} \alpha_{ij}\omega_j^L - \sum_{j:\alpha_{ij}>0} \alpha_{ij}\omega_j^U - (y_i b)^U + 1, \\
& \xi^U & = & \vartheta_i^U - \sum_{j:\alpha_{ij}<0} \alpha_{ij}\omega_j^U - \sum_{j:\alpha_{ij}>0} \alpha_{ij}\omega_j^L - (y_i b)^L + 1.
\end{array}
$$

When solving an instance of our problem, tightening typically happens after obtaining a new integer feasible solution or after branching on a variable. We describe here the tightening steps we observed when solving a few instances of our problem while enforcing branching on the binary variables $\bar{z}_i$. Note that the sequence of tightening steps is often repeated to ensure a tight enough bound interval on all variables.

After branching on a binary variable $\bar{z}_i$, if $\bar{z}_i$ is fixed to 0 the lower bound on the objective function is increased, which may allow for extra tightening on the variables appearing in the objective or, if the lower bound is above the cutoff, for pruning the node. If $\bar{z}_i$ is instead fixed to 1, `Couenne` uses the new lower bound on $\vartheta_i$ to obtain a better bound on $\omega$ and $b$. When a new upper bound $U$ is found (`Couenne` finds a good one at the beginning), this triggers a tightening of $\omega$. No tightening is done on any $\bar{z}_i$ variable since all of them have the same coefficient in the objective.

Note that in both cases (branching on $\bar{z}_i$ and new bound $U$) the tightening of $\omega$ and $b$ above allows one to strengthen the McCormick inequalities, that are necessary to build a linear relaxation of the problem. To summarize, both branches and a new cutoff value allow to tighten the bounds on $\vartheta$, $\omega$, and $b$, and this in turn allows for strengthening McCormick inequalities at every node.

Also, disabling bound reduction in `Couenne` leads to a dramatic worsening of the performance. It appears therefore that both bound reduction and McCormick cuts are essential to solve these problems efficiently.

## 8.6 Enhancing MILP solvers: Iterative domain reduction

Inspired by the outperforming results of `Couenne` over `IBM-Cplex`, we have tried to exploit some of the MINLP tools to deal with the weak relaxations associated with *big M* constraints.

Iterative domain reduction can be seen as a preprocessing tool to enhance the behaviour of `IBM-Cplex`. An initial bound tightening is performed by solving a sequence of MILP problems to strengthen $\omega$ variables.

Let us denote by $P$ the set of feasible solutions of the RLM, by $Z(\omega_i, \xi, z)$ the objective value (8.4) of the solution $(\omega_i, \xi, z)$, and by $U$ the value of an upper bound on (8.4). Lower ($l_i$) and upper ($u_i$) bounds on $\omega_i$ are iteratively generated by solving the following MILP problems:

$$l_i = \min\{\omega_i : (w, \xi, z) \in P, Z(w, \xi, z) \leq U\}, \forall i = 1, \ldots, d, \quad (8.23)$$

$$u_i = \max\{\omega_i : (w, \xi, z) \in P, Z(w, \xi, z) \leq U\}, \forall i = 1, \ldots, d. \quad (8.24)$$

In order to obtain the set of feasible solutions $P$, the MILP problems, (8.23) and (8.24), are solved within a node limit. Please note that at each step, solutions from $P$ must satisfy new bounds on $\omega$. This iterative process allows `IBM-Cplex` internal preprocessing tools to propagate the current domain of the variables.

We have tested this approach on the 23 instances from Table 8.1. First, an initial upper bound $U$ is computed by solving the RLM with a node limit of $100k$ (plus 10 polish nodes). Then, for each lower and upper bound tightening, the MILP problems are solved within a node limit of $100k$. Finally, the RLM is solved with all new bounds on $\omega$ variables. The results are reported in Table 8.3. All instances are solved to optimality in an average time of 30 seconds and average number of nodes of $412k$.

## 8.7 Conclusions

In this chapter we have studied the RLM addressed in Chapter 7 from a computational point of view. We have shown that the nonconvex reformulation of so-called *big M* constraints and the consequent use of a general-purpose MINLP solver instead of an MIQP solver can lead, surprisingly, to faster computing times for the RLM. Through a careful analysis of `Couenne` features and components we have been able to isolate those that make a difference, namely aggressive bound tightening and iterative strengthening of the McCormick linearization. It is conceivable that similar reformulations tightened in the same way can be effective for other problems involving logical implications and disjunctions of this type.

More precisely, we have argued that sophisticated (nonconvex) MINLP tools might be very effective to face one of the most structural issues of MILP, which is dealing with the weak continuous relaxations associated with *big M* constraints. The biggest challenge at the moment is to export these techniques, or better, their more extensive use, in solvers like `IBM-Cplex`. One successful attempt is to involve the use of aggressive MILP computation as a preprocessing to tighten bounds (Section 8.6).

As an interesting approach to enhance the RLM formulation performance in `IBM-Cplex`, one can strengthen constraints (8.5) iteratively on the tree as explained in Section 8.5.2. Basically, any time one can tighten the upper and lower bounds on $\omega$ and $b$, then one can recompute the (smaller) value of $M$ in any constraint (8.5) so as to make it stronger. However, as discussed in the previous sections, the tightening of the bounds on $\omega$ and $b$ is obtained by branching and propagation, thus it is inherently local to the subtree rooted in the node it happens. Thus, strengthened versions of constraints

Table 8.3: Computational results for `IBM-Cplex` enhanced with iterative domain reduction (i.d.r.). Instances of `TypeB` [33], $n = 100$, executed on an Intel Xeon E3-1220V2 at 3.10 GHz.

| | | `IBM-Cplex` i.d.r. | | | |
|---|---|---|---|---|---|
| | | | | % gap | |
| | optimal value | time (sec.) | nodes | ub | lb |
| 1 | 157,995.00 | 29.83 | 370,978 | – | – |
| 2 | 179,368.00 | 40.83 | 506,624 | – | – |
| 3 | 220,674.00 | 29.91 | 506,414 | – | – |
| 4 | 5,225.99 | 31.48 | 334,833 | – | – |
| 5 | 5,957.08 | 31.87 | 441,770 | – | – |
| 6 | 11,409,600.00 | 35.73 | 507,300 | – | – |
| 7 | 11,409,100.00 | 25.61 | 374,911 | – | – |
| 8 | 10,737,700.00 | 25.23 | 342,383 | – | – |
| 9 | 5,705,360.00 | 42.99 | 506,984 | – | – |
| 10 | 5,704,800.00 | 27.07 | 373,788 | – | – |
| 11 | 5,369,020.00 | 25.38 | 348,722 | – | – |
| 12 | 2,853,240.00 | 32.82 | 486,515 | – | – |
| 13 | 2,852,680.00 | 24.54 | 374,144 | – | – |
| 14 | 2,684,660.00 | 25.18 | 342,767 | – | – |
| 15 | 1,427,170.00 | 35.99 | 507,397 | – | – |
| 16 | 1,426,620.00 | 29.28 | 372,975 | – | – |
| 17 | 1,342,480.00 | 25.15 | 342,465 | – | – |
| 18 | 714,142.00 | 32.96 | 506,965 | – | – |
| 19 | 713,583.00 | 29.62 | 372,701 | – | – |
| 20 | 671,396.00 | 28.61 | 354,818 | – | – |
| 21 | 357,626.00 | 36.08 | 506,453 | – | – |
| 22 | 357,067.00 | 27.71 | 375,582 | – | – |
| 23 | 335,852.00 | 24.12 | 331,407 | – | – |

(8.5) are added as local cuts within the branch and bound tree. This strategy has been implemented in a work-in-progress version with `IBM-Cplex` with encouraging results. This approach deserves further study and testing.

# List of Figures

# List of Tables

# Bibliography


[1] Data instances for arc routing problems. `www.uv.es/corberan/instancias.htm`.

[2] S. Ali and K.A. Smith. On learning algorithm selection for classification. *Applied Soft Computing*, 6(2):119–138, 2006.

[3] S. Alumur and B.Y. Kara. Network hub location problems: The state of the art. *European Journal of Operational Research*, 190(1):1–21, 2008.

[4] E. Andersen and K. Andersen. Presolving in linear programming. *Mathematical Programming*, 71:221–245, 1995.

[5] L. Antonie, K. Inwood, D.J. Lizotte, and J. Andrew Ross. Tracking people over time in 19th century Canada for longitudinal analysis. *Machine Learning*, 95(1):129–146, 2014.

[6] C. Apte. The big (data) dig. *OR/MS Today*, 30(1):24–29, 2003.

[7] S. Aral and D. Walker. Tie strength, embeddedness, and social influence: A large-scale networked experiment. *Management Science*, 60(6):1352–1370, 2014.

[8] N. Archak, A. Ghose, and P.G. Ipeirotis. Deriving the pricing power of product features by mining consumer reviews. *Management Science*, 57(8):1485–1509, 2011.

[9] I. Averbakh, O. Berman, D. Krass, J. Kalcsics, and S. Nickel. Cooperative covering problems on networks. *Networks*, 63(4):334–349, 2014.

[10] B. Baesens. *Analytics in a Big Data World: The essential guide to data science and its applications*. John Wiley & Sons, 2014.

[11] B. Baesens, R. Setiono, C. Mues, and J. Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, 49(3):312–329, 2003.

[12] M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser. *Network Routing. Handbooks in Operations Research and Management Science*. North-Holland, 1995.

[13] M. Barthélemy. Spatial networks. *Physics Reports*, 499(1-3):1–101, 2011.

[14] J.E. Beasley. OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41:1069–1072, 1990.

[15] L. Bello, R. Blanquero, and E. Carrizosa. On minimax-regret Huff location models. *Computers and Operations Research*, 38:90–97, 2011.

[16] P. Belotti, P. Bonami, M. Fischetti, A. Lodi, M. Monaci, A. Nogales-Gómez, and D. Salvagnin. On handling indicator constraints in Mixed-Integer Programming. Technical Report OR/13/1, revised OR/14/20, DEI, University of Bologna, 2014.

[17] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5):597–634, 2009.

[18] O. Berman, Z. Drezner, and D. Krass. Big segment small segment global optimization algorithm on networks. *Networks*, 58(1):1–11, 2011.

[19] O. Berman and D. Krass. Flow intercepting spatial interaction model: A new approach to optimal location of competitive facilities. *Location Science*, 6(1-4):41–65, 1998.

[20] O. Berman, R.C. Larson, and N. Fouska. Optimal location of discretionary service facilities. *Transportation Science*, 26(3):201–211, 1992.

[21] D.P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athenas Scientific, Belmont, Mass, 1998.

[22] D. Bertsimas, M.V. Bjarnadóttir, M.A. Kane, J.C. Kryder, R. Pandey, S. Vempala, and G. Wang. Algorithmic prediction of health-care costs. *Operations Research*, 56(6):1382–1392, 2008.

[23] C.L. Blake and C.J. Merz. UCI Repository of Machine Learning Databases. `http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html`, 1998. University of California, Irvine, Department of Information and Computer Sciences.

[24] R. Blanquero and E. Carrizosa. A D.C. biobjective location model. *Journal of Global Optimization*, 23(2):139–154, 2002.

[25] R. Blanquero and E. Carrizosa. Continuous location problems and big triangle small triangle: Constructing better bounds. *Journal of Global Optimization*, 45:389–402, 2009.

[26] R. Blanquero and E. Carrizosa. On covering methods for d.c. optimization. *Journal of Global Optimization*, 18:265–274, 2009.

[27] R. Blanquero and E. Carrizosa. Solving the median problem with continuous demand on a network. *Computational Optimization and Applications*, 56(3):723–734, 2013.

[28] R. Blanquero, E. Carrizosa, and B.G. -Tóth. Maximal covering location problems on networks with regional demand. *Optimization Online*, 2014. `http://www.optimization-online.org/DB_HTML/2014/09/4553.html`.

[29] R. Blanquero, E. Carrizosa, B.G. -Tóth, and A. Nogales-Gómez. $p$-facility Huff location problem on networks. *Optimization Online*, 2014. `http://www.optimization-online.org/DB_HTML/2014/10/4619.html`.

[30] R. Blanquero, E. Carrizosa, A. Nogales-Gómez, and F. Plastria. Single-facility Huff location problems on networks. *Annals of Operations Research*, 222(1):175–195, 2014.

[31] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308, 2006.

[32] Bonmin, v. 1.7.4. `https://projects.coin-or.org/Bonmin`.

[33] J.P. Brooks. Support vector machines with the ramp loss and the hard margin loss. *Operations Research*, 59(2):467–479, 2011.

[34] S. Burer and A.N. Letchford. Non-convex mixed-integer nonlinear programming: A survey. *Surveys in Operations Research and Management Science*, 17(2):97–106, 2012.

[35] E. Carrizosa, B. Martín-Barragán, and D. Romero Morales. Multi-group support vector machines with measurement costs: A biobjective approach. *Discrete Applied Mathematics*, 156:950–966, 2008.

[36] E. Carrizosa, B. Martín-Barragán, and D. Romero Morales. Binarized support vector machines. *INFORMS Journal on Computing*, 22(1):154–167, 2010.

[37] E. Carrizosa, B. Martín-Barragán, and D. Romero Morales. A nested heuristic for parameter tuning in support vector machines. *Computers and Operations Research*, 43:328–334, 2014.

[38] E. Carrizosa, A. Nogales-Gómez, and D. Romero Morales. Strongly agree or strongly disagree?: Rating features in Support Vector Machines. *Optimization Online*, 2013.
`http://www.optimization-online.org/DB_HTML/2013/10/4082.html`.

[39] E. Carrizosa, A. Nogales-Gómez, and D. Romero Morales. Clustering categories in Support Vector Machines. *Optimization Online*, 2014.
`http://www.optimization-online.org/DB_HTML/2014/06/4403.html`.

[40] E. Carrizosa, A. Nogales-Gómez, and D. Romero Morales. Heuristic approaches for support vector machines with the ramp loss. *Optimization Letters*, 8(3):1125–1135, 2014.

[41] E. Carrizosa and F. Plastria. Location of semi-obnoxious facilities. *Studies in Locational Analysis*, 12:1–27, 1999.

[42] E. Carrizosa and D. Romero Morales. Supervised classification and mathematical optimization. *Computers and Operations Research*, 40:150–165, 2013.

[43] M. Cecchini, H. Aytug, G.J. Koehler, and P. Pathak. Detecting management fraud in public companies. *Management Science*, 56(7):1146–1160, 2010.

[44] A. Chang, D. Bertsimas, and C. Rudin. An integer optimization approach to associative classification. In *Advances in Neural Information Processing Systems*, pages 269–277, 2012.

[45] C.C. Chang and C.J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:1–27, 2011.

[46] W.A. Chaovalitwongse, Y.-J. Fan, and R.C. Sachdeo. Novel optimization models for abnormal brain activity classification. *Operations Research*, 56(6):1450–1460, 2008.

[47] P.C. Chen, P. Hansen, B. Jaumard, and H. Tuy. Weber's problem with attraction and repulsion. *Journal of Regional Science*, 32(4):467–486, 1992.

[48] Y. Chevaleyre, F. Koriche, and J.-D. Zucker. Rounding methods for discrete linear classification. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 651–659, 2013.

[49] R.L. Church and R.S. Garfinkel. Locating an obnoxious facility on a network. *Transportation Science*, 12(2):107–118, 1978.

[50] R.L. Church and C. ReVelle. The maximal covering location problem. *Papers of the Regional Science Association*, 32(1):101–118, 1974.

[51] Palmetto cluster. `http://citi.clemson.edu/palmetto/`.

[52] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 201–208, 2006.

[53] A. Corberán and J.M. Sanchis. A branch & cut algorithm for the windy general routing problem and special cases. *Networks*, 49:245–257, 2007.

[54] G. Cornuejols, M. Fisher, and G.L. Nemhauser. On the uncapacitated location problem. In P.L. Hammer, E.L. Johnson, B.H. Korte, and G.L. Nemhauser, editors, *Studies in Integer Programming*, volume 1 of *Annals of Discrete Mathematics*, pages 163 – 177. Elsevier, 1977.

[55] Couenne, v. 0.4. `https://projects.coin-or.org/Couenne`.

[56] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[57] G. Cui, M.L. Wong, and H.-K. Lui. Machine learning for direct marketing response models: Bayesian networks with evolutionary programming. *Management Science*, 52(4):597–612, 2006.

[58] J. Czyzyk, M.P. Mesnier, and J.J. More. The NEOS Server. *IEEE Computational Science Engineering*, 5(3):68–75, 1998.

[59] M.S. Daskin. *Network and Discrete Location. Models, Algorithms and Applications*. Wiley, New York, 1995.

[60] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32(3):281–331, 1987.

[61] G. Dobson and U.S. Karmarkar. Competitive location on a network. *Operations Research*, 35(4):565–574, 1987.

[62] T. Drezner. Locating a single new facility among existing, unequally attractive facilities. *Journal of Regional Science*, 34(2):237–252, 1994.

[63] T. Drezner and Z. Drezner. Finding the optimal solution to the Huff competitive location model. *Computational Management Science*, 1:193–208, 2004.

[64] Z. Drezner. *Facility Location. A Survey of Applications and Methods.* Springer, New York, 1995.

[65] Z. Drezner, K. Klamroth, A. Schöbel, and G. Wesolowsky. The Weber problem. In H.W. Hamacher and Z. Drezner, editors, *Facility Location: Applications and Theory*, pages 1–36. Springer, 2001.

[66] Z. Drezner and A. Suzuki. The big triangle small triangle method for the solution of non-convex facility location problems. *Operations Research*, 52:128–35, 2004.

[67] H.A. Eiselt, G. Laporte, and J.-F. Thisse. Competitive location models: A framework and bibliography. *Transportation Science*, 27(1):44–54, 1993.

[68] E. Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46(1):48–60, 1990.

[69] E. Erkut and S. Neuman. Analytical models for locating undesirable facilities. *European Journal of Operational Research*, 40(3):275–291, 1989.

[70] S. Ertekin, L. Bottou, and C.L. Giles. Nonconvex online support vector machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):368–381, 2011.

[71] X. Fang, O.R. Liu Sheng, and P. Goes. When is the right time to refresh knowledge discovered from data? *Operations Research*, 61(1):32–44, 2013.

[72] J. Fernández, B. Pelegrín, F. Plastria, and B.G. -Tóth. Solving a Huff-like competitive location and design model for profit maximization in the plane. *European Journal of Operational Research*, 170:1274–87, 2007.

[73] R.A. Fisher. The use of multiple measurements in taxonomy problems. *Annals of Eugenics*, 7:179–188, 1936.

[74] J. Fliege, K. Kaparis, and B. Khosravi. Operations research in the space industry. *European Journal of Operational Research*, 217(2):233–240, 2012.

[75] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[76] G. Fung and O.L. Mangasarian. A feature selection Newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202, 2004.

[77] B.F. Gage, A.D. Waterman, W. Shannon, M. Boechler, M.W. Rich, and M.J. Radford. Validation of clinical classification schemes for predicting stroke: Results from the national registry of atrial fibrillation. *Journal of the American Medical Association*, 285(22):2864–2870, 2001.

[78] A. Ghosh, S. McLafferty, and C.S. Craig. Multifacility retail networks. In Z. Drezner, editor, *Facility Location. A Survey of Applications and Methods*, pages 301–330, New York, 1995. Springer.

[79] M. Golea and M. Marchand. On learning perceptrons with binary weights. *Neural Computation*, 5(5):767–782, 1993.

[80] Gurobi, v. 5.5.0. `http://www.gurobi.com`.

[81] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[82] S.L. Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12(3):450–459, 1964.

[83] S.L. Hakimi. On locating new facilities in a competitive environment. *European Journal of Operational Research*, 12(1):29–35, 1983.

[84] H.W. Hamacher and Z. Drezner. *Facility location: Applications and theory.* Springer, New York, 2002.

[85] H. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* MIT Press, 2001.

[86] P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Mathematical Programming*, 79(1-3):191–215, 1997.

[87] P. Hansen, D. Peeters, D. Richard, and J.F. Thisse. The minisum and minimax location problems revisited. *Operations Research*, 33(6):1251–1265, 1985.

[88] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, New York, 2001.

[89] D.S. Hochbaum, C.-N. Hsu, and Y.T. Yang. Ranking of multidimensional drug profiling data by fractional-adjusted bi-partitional scores. *Bioinformatics*, 28(12):i106–i114, 2012.

[90] M.J. Hodgson. A flow-capturing location-allocation model. *Geographical Analysis*, 22(3):270–279, 1990.

[91] J.N. Hooker and H.P. Williams. Combining equity and utilitarianism in a mathematical programming model. *Management Science*, 58(9):1682–1693, 2012.

[92] R. Horst and N.V. Thoai. Dc programming: Overview. *Journal of Optimization Theory and Applications*, 103:1–43, 1999.

[93] H. Hotelling. Stability in competition. *Economic Journal*, 39:41–57, 1929.

[94] Hubble Telescope. `http://hubblesite.org/`, Accessed: 11-11-2014.

[95] D.L. Huff. Defining and estimating a trading area. *Journal of Marketing*, 8:28–34, 1964.

[96] D.L. Huff. A programmed solution for approximating an optimum retail location. *Land Economics*, 8(42):293–303, 1966.

[97] IBM-Cplex, v. 12.3. `http://www-01.ibm.com/software/integration/optimization/cplex/`.

[98] IBM-Cplex, v. 12.4. `http://www-01.ibm.com/software/integration/optimization/cplex/`.

[99] IBM-Cplex, v. 12.5. `http://www-01.ibm.com/software/integration/optimization/cplex/`.

[100] IBM-Cplex, v. 12.6. `http://www-01.ibm.com/software/integration/optimization/cplex/`.

[101] Ipopt, v. 3.9.2. `http://projects.coin-or.org/Ipopt`.

[102] R.L. Keeney and R.S. Gregory. Selecting attributes to measure the achievement of objectives. *Operations Research*, 53(1):1–11, 2005.

[103] M. Labbé, D. Peeters, and J.F. Thisse. Location on Networks. In M.O. Ball et al., editor, *Handbooks in operations research and management science*, volume 8, pages 551–624, Amsterdam, 1995. Elsevier.

[104] E.L. Lawler and D.E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[105] B. Letham, C. Rudin, T.H. McCormick, and D. Madigan. Building interpretable classifiers with rules using bayesian analysis. Technical Report tr609, University of Washington, 2012.

[106] F. Li, Y. Yang, and E. Xing. From Lasso regression to feature vector machine. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18, pages 779–786. MIT Press, Cambridge, MA, 2006.

[107] G.Y.H. Lip, R. Nieuwlaat, R. Pisters, D.A. Lane, and H.J.G.M. Crijns. Refining clinical risk stratification for predicting stroke and thromboembolism in atrial fibrillation using a novel risk factor-based approach: The Euro heart survey on atrial fibrillation. *CHEST Journal*, 137(2):263–272, 2010.

[108] H. Liu, F. Hussain, C. Tan, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6(4):393–423, 2002.

[109] Y. Liu and Y. Wu. Optimizing $\psi$-learning via mixed integer programming. *Statistica Sinica*, 16:441–457, 2006.

[110] R.F. Love, J.G. Morris, and G.O. Wesolowsky. *Facilities location: Models and methods*. North Holland, New York, 1988.

[111] V. Maniezzo, T. Stützle, and S. Voss. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, volume 10 of *Annals of Information Systems*. Springer, 2009.

[112] V. Marianov and D. Serra. Location-allocation of multiple-server service centers with constrained queues or waiting times. *Annals of Operations Research*, 111:35–50, 2002.

[113] D. Martens, B. Baesens, T.V. Gestel, and J. Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, 183(3):1466–1476, 2007.

[114] D. Martens and F. Provost. Explaining data-driven document classifications. *MIS Quarterly*, 38(1):73–99, 2014.

[115] G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I - Convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.

[116] E. Melachrinoudis and Z. Xanthopulos. Semi-obnoxious single facility location in Euclidean space. *Computers and Operations Research*, 30(14):2191–2209, 2003.

[117] F. Messine. Deterministic global optimization using interval constraint propagation techniques. *RAIRO-RO*, 38(4):277–294, 2004.

[118] P.B. Mirchandani and R.L. Francis. *Discrete location theory*. Wiley, New York, 1990.

[119] A.J. Mumphrey, J.E. Seley, and J. Wolpert. A decision model for locating controversial facilities. *Journal of the American Institute of Planners*, 37(6):397–402, 1971.

[120] C. Müssel, L. Lausser, M. Maucher, and H.A. Kestler. Multi-objective parameter selection for classifiers. *Journal of Statistical Software*, 46(5):1–27, 2012.

[121] Y. Ohsawa and K. Tamura. Efficient location for a semi-obnoxious facility. *Annals of Operations Research*, 123(1-4):173–188, 2003.

[122] A. Okabe and M. Kitamura. A computational method for market area analysis on a network. *Location Science*, 5(3):198–198, 1997.

[123] A. Okabe and K.-I. Okunuki. A computational method for estimating the demand of retail stores on a street network and its implementation in GIS. *Transactions in GIS*, 5(3):209–220, 2001.

[124] K.-I. Okunuki and A. Okabe. Solving the Huff-based competitive location model on a network with link-based demand. *Annals of Operations Research*, 111(1-4):239–252, 2002.

[125] C. Orsenigo and C. Vercellis. Multivariate classification trees based on minimum features discrete support vector machines. *IMA Journal of Management Mathematics*, 14(3):221–234, 2003.

[126] C. Orsenigo and C. Vercellis. Discrete support vector decision trees via tabu search. *Computational Statistics and Data Analysis*, 47(2):311–322, 2004.

[127] P.H. Peeters and F. Plastria. Discretization results for the Huff and Pareto-Huff competitive location models on networks. *Top*, 6(2):247–260, 1998.

[128] F. Plastria. GBSSS: The generalized big square small square method for planar single-facility location. *European Journal of Operational Research*, 62(2):163 – 174, 1992.

[129] F. Plastria. Continuous location problems. In Z. Drezner, editor, *Facility Location. A Survey of Applications and Methods*, pages 225–262, New York, 1995. Springer.

[130] F. Plastria. Static competitive facility location: An overview of optimisation approaches. *European Journal of Operational Research*, 129(3):461–470, 2001.

[131] F. Provost and T. Fawcett. *Data Science for Business: What you need to know about data mining and data-analytic thinking*. O'Reilly Media, Inc., 2013.

[132] P. Raghavan and C.D. Tompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[133] G Reinelt. TSPLIB - A traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

[134] C.S. ReVelle, H.A. Eiselt, and M.S. Daskin. A bibliography for some fundamental problem categories in discrete location science. *European Journal of Operational Research*, 184(3):817–848, 2008.

[135] G. Ridgeway. The pitfalls of prediction. *National Institute of Justice Journal*, 271:34–40, 2013.

[136] S. Roksandić, E. Carrizosa, D. Urošević, and N. Mladenović. Solving multifacility Huff location models on networks using variable neighborhood search and multi-start local search metaheuristics. *Electronic Notes in Discrete Mathematics*, 39(0):121 – 128, 2012.

[137] D. Romero Morales, E. Carrizosa, and E. Conde. Semi-obnoxious location models: A global optimization approach. *European Journal of Operational Research*, 102(2):295–301, 1997.

[138] D. Romero Morales and J. Wang. Forecasting cancellation rates for services booking revenue management using data mining. *European Journal of Operational Research*, 202(2):554–562, 2010.

[139] V. Roth. The generalized LASSO. *IEEE Transactions on Neural Networks*, 15(1):16–28, 2004.

[140] C. Rudin and K.L. Wagstaff. Machine learning for science and society. *Machine Learning*, 95(1):1–9, 2014.

[141] M. Saar-Tsechansky, P. Melville, and F. Provost. Active feature-value acquisition. *Management Science*, 55(4):664–684, 2009.

[142] M.W.P. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, 6:445–454, 1994.

[143] D. Serra, C. ReVelle, and K. Rosing. Surviving in a competitive spatial market: The threshold capture model. *Journal of Regional Science*, 39(4):637–652, 1999.

[144] X. Shen, G.C. Tseng, X. Zhang, and W.H. Wong. On $\psi$-learning. *Journal of the American Statistical Association*, 98:724–734, 2003.

[145] P. Tambe. Big data investment, skills, and firm value. *Management Science*, 60(6):1452–1469, 2014.

[146] B.C. Tansel, R.L. Francis, and T.J. Lowe. State of the art – Location on networks: A survey. Part I: The p-center and p-median problems. *Management Science*, 29(4):482–497, 1983.

[147] M. Tawarmalani and N.V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, Boston MA, 2002.

[148] C. Toregas, R. Swain, C. ReVelle, and L. Bergman. The location of emergency service facilities. *Operations Research*, 19(6):1363–1373, 1971.

[149] P.D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.

[150] H. Tuy. *DC Optimization: Theory, Methods and Algorithms, Handbook of Global Optimization*. Kluwer Academic Publishers, Dordrecht, Holland, 1995.

[151] H. Tuy, F. Al-Khayyal, and F. Zhou. A d.c. optimization method for single facility location problems. *Journal of Global Optimization*, 7:209–227, 1995.

[152] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[153] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[154] P. Vlachos and M. Meyer. StatLib, 1989. `http://lib.stat.cmu.edu`.

[155] A. Wächter and L.T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[156] L. Wang, H. Jia, and J. Li. Training robust support vector machine with smooth ramp loss in the primal space. *Neurocomputing*, 71(13-15):3020–3025, 2008.

[157] A. Weber. *Über den standort der Industrien*. Tübingen, Germany, 1909. (Translation: On the location of industries, 1929, Chicago University Press, Chicago).

[158] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In *Proceedings of Neural Information Processing Systems*, pages 668–674, 2001.

[159] H.P. Williams. *Model building in Mathematical Programming.* Wiley, New York, 1985.

[160] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14:1–37, 2007.

[161] Y. Wu and Y. Liu. Robust truncated hinge loss support vector machines. *Journal of the American Statistical Association*, 102(479):974–983, 2007.

[162] Xpress, v. 7.6.
`http://www.fico.com/en/products/fico-xpress-optimization-suite`.

[163] Y.T. Yang, B. Fishbain, D.S. Hochbaum, E.B. Norman, and E. Swanberg. The supervised normalized cut method for detecting, classifying, and identifying special nuclear materials. *INFORMS Journal on Computing*, 26(1):45–58, 2014.