UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

Tinder for Cows

A constrained optimization problem

Dissertation presented by
Ninon DE GREEF , Grégory SCHOT

for obtaining the Master's degree in
Computer Science

Supervisor(s)
Pierre SCHAUS

Reader(s)
Kim MENS, Hélène VERHAEGHE , Fabienne DUPUIS

Academic year 2017-2018

Abstract

Cow milk is the basis product of the whole dairy industry. To be produced, dairy cows must give birth to calves. Those can be sold, or can stay in the herd for animal husbandry. It is interesting to select the bull that will impregnate a particular cow according to some criterion, like the consanguinity or the milk quality. Until now, it was done by hand and by humans.

Constrained programming is a programming paradigm that proves itself very useful in this situation. With a proper modelling of the problem, it can not only find solutions to the problem, finding suitable bulls for cows in a farm, but it can above all find the best.

This master thesis is not a theoretical problem. It is based on real data and answer the need for technical assistance to solve a problem. In this dissertation, we present the work that has been done in order to meet the required desires of professional.

Acknowledgment

First of all, we would like to thank our supervisor, Pierre Schaus and particularly Hélène Verhaeghe for their support, their help and the benevolence they showed toward us during the whole year.

We would also like to thank our families and friends for their interest in the subject, their support and presence during those three years at University.

Sebastien Combéfis's book *Latex, How to* has been particularly useful during the writing of this work, so we would like to thank him for it.

Last but not least, we would like to thank each other for the hard work, the great team that we form and the friendship that link us, only reinforced by this work.

Contents

Chapter 1

Introduction

1.1 Description of the problem

For a cow to produce milk, it must have given birth to a calf first. The calf can be sold for its meat or kept in the herd to grow. The female will produce milk when they are of age whilst the male can be kept for natural reproduction. Whether the outcome for the calf, it is interesting to select carefully the bull that will be the father in order to optimize some features of the calf, depending on the objective. The possible selection criterion are countless! A low consanguinity rate, high milk quality and quantity, price of the insemination doses, storage and packaging of the insemination doses, physical characteristics, facility of calving, race, ... Those are some examples of what can be taken into account during the selection process.

Nowadays, the selection process of the bulls is done by hand, with the help of the data that the farmers gather according to their needs. With a sufficient amount of data, this job could be automated. Moreover, the task could be optimized, giving the best results possible with the given data. It would be a huge improvement in the farmers' lives, saving them time, work and possibly headache over this recurrent problem. An automated process would also prevent inattention mistakes during the process.

1.2 Conditions

To automate the process, there are two conditions. First, we need a lot of digitized data. Second, the problem must be modelizable in term of constraints to be able to solve the problem with the Constrained Programming paradigm.

The first condition is not a problem: the farm we worked with have been collecting data about their cows for many years now. With their help, we could be able to use them at their greatest extent. The second condition is precisely the subject of this work. Is it possible to modelize the problem with numerical tools and solve it, providing results that are effectively useful in the real life? The answer is given in the following work.

1.3 Outline

The data we have been provided for are detailed in the chapter 3, where we specify the formats, explain what it represents in real life and what it could be used for. With the help of the farmers at the initiation of the subject of this thesis, we decided the constraints that were to be implemented. We explain them and how we modelized them in chapter 4. Having many constraints available to choose the bulls is good, but being able to decide whether to use a

constraint or not, or in other words, parameterize the model, is better. The different kind of parameterization we enabled are detailed in chapter 5. Chapter 6 validates our work by proving that our solution work as expected. Finally, this master thesis is concluded with chapter 7.

1.4 Repository

The code we produced as part of this project can be found on our repository. It will stay public for 6 months to allow our readers to have a look at the code in order to understand better and evaluate our work at its true value.
`https://bitbucket.org/greg020/tindercows/src/master/`

Chapter 2

State of the art

2.1 Constraint programming

Constraint programming (CP) is a very powerful programming paradigm capable of solving high combinatorial problems that is often used in the area of planning and scheduling. Rather than defining a various number of steps for the program to follow like in a classic imperative program, in constraint programming we specify the properties of the problem to be solved. Thus, CP is a form of declarative programming as it expresses the logic of a computation without describing its control flow.

A constrained program is composed of two entities: the model and the solver. They are two independent processes: a model can be applied to different searching techniques and a solver can be used for different problems and models. The search is done with general-purpose heuristics instead of problem-specific ones. The model is the representation of the problem in a declarative way with decision variables and constraints. It could be either an instance of a constrained satisfaction problem (CSP) or an instance of a constrained optimization problem (COP). The former aims at finding feasible solutions while the latter finds the best solution according to some minimum or maximum function(s). CSP and COP yield a natural representation of the problem so it is easier to model the problem and then solve it with existing solvers than try and design a custom solution using another search technique [1].

Let's take the well known Sudoku problem to illustrate CP. A Sudoku is a 9 x 9 grid like figure 2.1 to be filled with numbers from 1 to 9. Each number must appear exactly 9 times in the grid. Each line, column and each of the nine 3 x 3 subgrid must contain each number only once. Some cells already contain a value such that a Sudoku has only one solution. Although some can be tricky to solve by hand, taking tens of minutes, even the hardest Sudoku problems yields to a CSP solver in less than 0.1 seconds [1].

A Sudoku model is thus a CSP instance. A model contains decision variables, each having a domain, and constraints. The decision variables represent the decisions that we have to take in the real world; here, numbers to write in cells. A Sudoku is modelled with 81 variables, one for each square, where the fixed squares have a domain consisting of a single value. The domain of a decision variable is the set of possible values; here initially, the domain of a non fixed variable contains all of the nine numbers. Then, we must express the problem in term of constraints. For the Sudoku problem, only one type of constraint is needed: the constraint "All different" (alldiff) [2] [3]. It takes a set of decision variables and assert that their fixed values are all different. We need an alldiff constraint for each line, each column and each subgrid, thus 27 constraints. The model is done, we have described what we want and not how to obtain it. The decision variables of this model and their domain are represented in figure

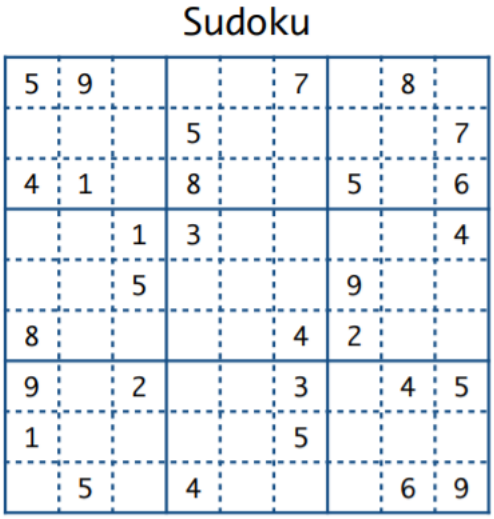


Figure 2.1: Example of a Sudoku grid

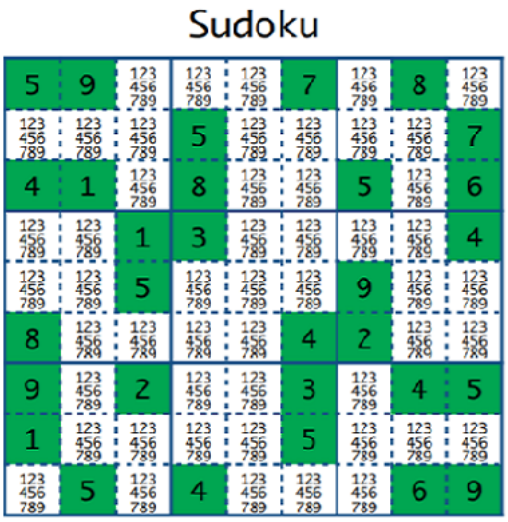


Figure 2.2: Representation of the decision variables

2.2. Green cells are the fixed ones and the numbers inside a cell are the possible values for that cell.

Once the model is done, it is given to the solver to be resolved. This may look like magic but it is actually smart brute-force. The solver combines systematic search and constraint satisfaction techniques [4]. Systematic search guarantee a complete search space. With the help of a search tree, whose nodes represent a partial solution and branches the assignation of a variable, the solver can use the generate-and-test technique. This technique consist of trying a possible value for a variable and find out if it leads to a solution or a dead-end if a constraint is violated, forcing the search to backtrack and try another assignation until the search space is completely explored, allowing the solver to return the optimal solution. To speed up the process and avoid backtracking as much as possible, the solver uses constraint satisfaction techniques such as arc and path consistency enforcing, variable ordering, branch and bound and other constraint optimisation techniques, symmetry breaking, etc. We are not going to detail those techniques here as this work is mainly about modelling.

Constraint programming differs from procedural programming in the way of reasoning. It sees a problem from a different angle. A procedural program is a succession of steps leading to a solution. Solving the Sudoku problem in a procedural way would actually be equivalent to rewriting the process of the solver, in a less efficient way. Finding an answer would require a complete scan of the grid with checks of the constraints at each step: not very efficient. It is easy for a human to scan and find rapidly obvious answers but not for a procedural program. Moreover, the resulting program might be hard to understand. The power of constraint programming lies in its ability to let the computer do the complex computations when it has been given the rules of the game. The programmer simply has to define those rules correctly and in terms intelligible for the used framework, in our case OscaR.

This works fine with games having strict rules. In the real world, it is not always simple to model the reality with constraints because we have a notion of priority. It is easy to end up with a too constrained model giving no results. How to find the hard constraints that caused the problem? How to relax the problem such that we have results? How to tell the model that a certain set of constraints is really important, and another set is less? We might have constraints

that we would like to see respected, but, not at the price of not having results at all. This is where the notion of hard and soft constraints is useful. Hard constraints are the one that must be respected in every case. By default, constraints are hard. Soft constraints on the contrary are constraints that can be violated in order to find results. In this project, we allowed the user to define himself which constraints must be hard or not! See section 5.1.2 for more information.

2.2 Scala

Scala is a multi-paradigm programming language. Its name comes from "Scalable language". It combines the object-oriented paradigm and the functional programming paradigm in one concise, high-level language. Scala's static types help avoid bugs in complex applications, and its JVM and JavaScript runtimes let you build high-performance systems with easy access to huge ecosystems of libraries [5]. It runs on the JVM, so Java and Scala stacks can be freely mixed for totally seamless integration. Java being the language we were the most familiar with, using Scala has been pretty easy for us and had the advantage of providing more advanced functionalities.

2.3 OscaR

Oscar is a Scala toolkit for solving Operation Research problems. The techniques currently available in OscaR are [6]:

- Constraint programming
- Constraint based local search
- Derivative free optimization
- Visualization

Obviously, this is the framework we used to model our problem and find solutions. This project is supported by those companies/institutions:

- UCLouvain and the research group of Pierre Schaus research lab.
- CETIC who develop and maintain the CBLS package and its satellite engines
- n-Side who use OscaR and allocate resources to improve it.
- YourKit, supporting this open source projects with its full-featured Java Profiler.

Many of the ideas implemented in OscaR come from the pioneering system/language Comet developed by Pascal Van Hentenryck and Laurent Michel. This system was the first to allow users full control to create complex hybridizations between CBLS/CP/LP components. A particularly desirable feature of Comet was that the models always looked nice, and consequently were easy to read and understand.

Another solver and API that inspired the creators of OscaR, because they used it in the past, is Ilog-Solver. This company published many important papers for the field that were implemented in many solvers [6].

Chapter 3

Data and web parser

For this master thesis, we worked with real data provided by a Belgian milking farm. Our solution is custom to their data format and to their needs. First, we describe the file data we received, what they represent in the real world and how we dealt with them. Then we explain the need for a web parser and how it works.

3.1 Description of the files

3.1.1 Births

The birth file gathers information about the births that happened in the farm for the last 28 years, the first record being about a birth that happened on the 3^{th} of October 1990. Thanks to that file, we were able to compute a graph of the genealogy of the cows in the actual herd, graph that we need for the consanguinity constraint. It gives information about the mother's calving abilities, the father's identity and about the calf itself. Figure 3.1 shows a quite representative subset of the file. Here is the description of each of the columns of the files. We explain what they represent, what kind of values can be found in each column and how it was useful for the task.

1. The first column is the birth ID. It can be a number or a number followed by a letter. IDs with a letter means that the cow gave birth to twins or triplets or more, depending on the number of lines we have for that birth. This situation can be observed in the figure 3.1 with the lines 38a and 38b. This birth ID is useful to identify a certain birth. It is used in the file mostly to link a cow with other birth that she gave, like the previous, the next one or even the first one (See points 4, 6 and 18 for more details).

37	-1	-17	/	1	/	Buisset Rouge	B Rouge	main petit veau	9033	3065	Buisset Rouge-1	PR	20/01/92	F		130	
38a	9038	4674	7	2	75	Bertha	Bell Robbie	seule jumeaux	9033	3039	Bertha-1	PN	30/01/92	F		137	1
38b	9038	4674	7	2	75	Bertha	Bell Robbie	seule jumeaux	9033	3040	Pietje-1	PN	30/01/92	F		133	1
39	-1	-5	18	2	78	Bella	Poulin	vt + véleuse				PB	08/02/92	M	V		1
40	9038	4673	11	2	88	310	Valiant	main	9033	3041	T-Steiny310	PN	09/02/92	M			
41	9042	4990	6	2	80	Elli	Poulin	véleuse	-1	7070	Elli-1	PB	10/02/92	F		138	
42	9038	4683	5	2	84	Ex-Gérardine	Junior	véleuse				PB	16/02/92	M	V		
43	9038	4680	15	2	86	Riquitta	Junior	césarienne				PB	20/02/92	M	V		1
44	9038	4681	12	2	89	880		véleuse				PN	21/02/92	M	V		
45	9042	4992	16	2	85	Anette	Astra	seule				PB	29/02/92	M	V		
46	-1	-3	10	2	/	Grande Rouge	Junior	vt				PB	02/03/92	M	V		1
47	9038	4685	2	2	79	Fernande	Laron	vt + véleuse				PB	08/03/92	M	V		1
48	9038	4682	13	2	83	133		véleuse				PN	13/03/92	M	V		
49	-1	-7	21	2	82	Anie	Laron	véleuse					18/03/92	M	V		
50	9038	4669	20	2	90	Suzanne	Laron	véleuse	9042	4991	Suzanne-1	PB	20/03/92	F		189	
51	9038	4670	9	2	87	Gérardine	Laron	véleuse	9033	3067	Gérardine-1	PB	21/04/92	F		184	
52	-1	-8	22	2	/	Fleurie	Laron	vt + véleuse				Blanc	28/04/92	F	MN		1

Figure 3.1: Subset of the "Births" file

2. The second column goes in pair with the third. Together, they form the mother's identification number. This ID is divided in two parts because of the legislation. One part itself can appear more than once but the combination of the two numbers is unique. Some numbers are negatives: those are not real data. Preprocessing has been done here by hand because the data were incomplete. Those cases are recognisable by the negative sign and still, each mother have a unique identification pair. The first line in the data subset figure 3.1 shows such a situation. The first parameter of the cows ID is referenced as IDa.

3. As previously said in point 2, this value represent the second half of the mother's identification number. It is referenced as IDb.

4. The fourth column in the file is a shortcut to identify the line of the previous birth of the mother. If the mother was a heifer before, like the first line of the figure 3.1, the slash sign (or nothing) appears instead of a birth ID (see point 1). This shortcut is mostly used with the handwritten version of the data. We didn't use it because we scanned every line of the file anyway.

5. The fifth column is a number representing the number of times the cow gave birth. If it is the mother's n birth, we should find $n-1$ birth line for that mother in the file before that line. Again, we did not use this column as it is only useful when the data are handled by hand on non-digital supports. However, we keep a count of the number of birth we find for that very cow, but its value is not based on this column.

6. Sixth column is similar to the fourth. It is a Birth ID, a shortcut to the next line birth of the mother if there is any. Otherwise, a slash or nothing appears. We ignore it.

7. Seventh column is the mother's name. It might be unknown and thus, empty. Female cows are identified by their double ID, not by their name, so this is not a problem.

8. Next is the father's name in short notation. Unlike the mother, the father IS identified with its name in the given data. If it's empty or "XXXXXX" or "?" appears, it means that the father is unknown. It also means that we cannot identify the father in any way because there are no other information about the father in the file. In some case, there are two names separated by a slash. It means we don't know exactly which one of the two is the father. This situation might happen when the farmer let his own bulls in the herd for natural reproduction.

9. Ninth column is a short comment about how the birth went, in French. This information is not useful for the task, thanks to the last column (see point 18).

10. The tenth and eleventh columns are similar to the second and third. Together, they form the calf's identification number. This part is referred as IDa. Again negative values can appear, meaning that some handmade preprocessing has been done. The complete ID might be unknown, as the farmer did not bother to register the identification number of calves that have been sold right away or born dead.

11. IDb of the calf.

12. Name of the calf. Usually, it is derived from the mother's name: for a female calf, a number would be added to the mother's name (like in the first line of the figure 3.1); for a male, a T would precede the mother's name. The name is not always stated, which is a problem for the males that were breeded and used for natural reproduction later on. The file "Lost names" helped us to palliate to this problem.

13. The calf's coat colour. PN stands for "Pie Noir", PR for "Pie Rouge" and PB for "Pie Bleu". This information is not useful for the task.

14. The birth date in the format dd/mm/yy. This information is always provided. It is useful for the selection of the cows, more detailed in the chapter 5.

15. The sex of the calf. F stands for "Female", M for "Male" and ? appears when the sex is unknown, when the farmer didn't bother to check in the case of a born dead celf for example. This is crucial information, as the cows and bulls are referred to in different ways.

16. Status of the calf. Could be V for "Vendu" (sold), MN for "Mort né" (born dead), DCD for "Décédé" (dead) or nothing when the status is unknown or none of the three. We use this information in order not to add data for born dead calves.

17. When the calf is a Female and she gave birth in the farm later on, this column contains the birth ID of her first birth. Just like column 4 and 6, it is a shortcut. We did not use it.

18. This column gives information about the difficulty of the calving. When it is set to 1, it means that the birth or the pregnancy has been difficult in some way. More information of often provided in the comment (see point 9). This value is used to learn the capacity of calving of the mother, needed for the calving difficulty constraint (see section 4.2).

As you can see in the description of the file, it provides a lot of information about the herd of the farm and the kinship links between the animals. We handled this file by extracting data line by line. We ended up with two dictionaries: one for the female and one for the male. We have to handle the two sex in a different way because their identification method (and thus the key in the dictionary) are not the same. The female are designated by their double ID and the male by their name. For each line, we extract data about the mother, or complete the data we already have. We also keep count of the number of birth she gave and the number of birth that were difficult. If its not born dead, we also add the calf to the dictionary corresponding to its gender. When its gender is unknown, we don't deal with it as we suppose the animal did not produce any descendants anyway, so it is not interesting for the genealogy graph. In the instance of the calf, we keep track of its parents, meaning its mother and zero, one or two fathers.

The genealogy graph is constructed at the end of the treatment of this file. The node objects are all children of the Bovine class, either representing a female or a male cow. The links between the nodes are representing the child-parent relation. Those links are unidirectional as we only have a reference from a child to its parent and not from a parent to its children.

This file contains data that have been gathered by hand in a non-numerical support before having been encoded. Unsurprisingly, some part of the data are missing. For example, after 2011, the calf names are not registered anymore. The female born after that can still be identified with their double IDs but it becomes impossible to keep tracks of the male calves anymore. It is a problem when those calves are used for reproduction later on. To overcome that, we have used the "Lost Names" file and the "Bull breeding" file.

3.1.2 Lost Names

The file we named "Lost Names" contains IDs and names of bulls that were in the farm at some point. Some of them were bought (their ID starts with -2) and others were born in the farm. With this file, we can fill the gap of the male calves that do not have their names registered in the birth file. We have to process it before the birth file so that, when we reach the birth line of the bull, the identification can be done straight away and all the data about the birth are gathered in the bull instance.

	I	II	III	IV	V
T-0316	Goodtime	?	?	?	?
T-Genisse	?	?	?	?	?
T-Steiny310	Valiant	?	?	?	?
T-Finette	Eblack	T-Steiny310	?	T-Genisse	?
T-7937	Toyota	Blitz	T-0316/T-Sabine	Bellwood	?
T-Miranda	Roucky	Goodtime	T-Steiny310	Admiration	?
T-Sabine	Rudolph	T-Facade	T-Steiny310	B Rouge	?
T-Facade	Conceal	?	?	?	?
T-9932	Blitz	Goodtime	Jubilant	T-Steiny310	B Rouge

Figure 3.2: A subset of the bull breeding file

3.1.3 Bull Breeding

This file contains part of the genealogy of the bulls that have been breeded in the farm. The figure 3.2 shows a subset of the file. Each row represent the genealogy of one bull: the one whose name appear on the first column. The column "I" is the father, "II" the maternal grandfather, "III" the maternal grandmother's father, and it goes like this up to five generations. The mothers are omitted here. With this file, we can add more kinship links into the genealogy graph. It is useful to compensate the absence of the calf names in the birth file. For example, T-7937 appears in the birth file only as a father and never as a calf, even if it was born in the farm so its birth line must be in the file.

So, the bulls whose genealogy are contained in the file often already exist in the system, because they have been seen in the birth file. Some of them already have a link to their mother. Other don't. For the latter case, we create what we call "ghost mom", a cow that we create with a invented id and without any other information. Those moms are created for the sole purpose to implement the kinship links that are provided by the file. Thus, our bull T-7937 will have a link to a ghost mom, even though its real mom probably exists in the file. We are just unable to merge the ghost mom and the real mom because of a lack of information. That is unfortunate but not such a problem for the consanguinity constrain. Here, we can assume that the ghost moms have a consanguinity rate of zero.

Again, the father can be unknown or there can be a certain uncertainty about the father, just as in the birth file. It is handled the same way.

3.1.4 Bull IA

This file is a list of the fathers that are not from the herd of the farm. Those bulls are referenced with their name in short notation, which means that they also have a full length name. The farmer did not bother to collect data about the genealogy of those reproducing bulls, as those data can be found online. He simply provided us with URL to the online description page of each of the bulls. The file is not complete though, some of the bulls' full name and URL are not provided.

We built and used a web parser to get those data online. See section 3.2.

3.1.5 Sample

The file named "Sample" is filled with data gathered from sampling the milk of the cows. Those data are the somatic cells rate in the format "10^3 SCC", the percentage of fat and the percentage

of proteins observed in each sampling. Those data are mandatory for the quality constraints (sections 4.3 and 4.4) and the global improvement constraints (sections 4.6 and 4.7.

3.1.6 Conformation

In this file, we learn more about the physique of the cows. Those data are gathered by hand, at the appreciation of the farmer. They just observed the cows and gave a note from the set $\{-2, -1, 0, 1, 2\}$ for each feature. When the values are missing, we assume the feature is neutral and we assign it the value 0. The negative and positive signs do not mean good or bad, they represent a value on a scale from "very short" to "very long" or "ugly" to "beautiful" or even "arched inwardly" to "arched outwardly" depending on the feature. The features are:

- Mammal system
- Udder depth
- Median suspension
- Front teat position
- Back teat position
- Teat length
- Limbs
- Foot angle
- Side view of the limbs
- Back view of the limbs

We also have the double IDs of the cows to identify them and the date on which the observation was made. A same cow might have been observed several time, appearing thus more than once in the file. In this case we keep the last data registered.

3.2 Web Parser

Unlike the collection of data about cows, that is under the farmer's responsibility, we cannot ask the user of our application to provide the data about the bulls that he doesn't breed. The task of collecting those data by hand would require a lot of time, might introduce encoding errors and would need constant updates to follow the evolution of the market. The market evolves, having younger bulls available for reproduction regularly. The constant improvement in the zootechnique characteristics of the bovines induce a change in the statistics of older bulls because many criterion are compared to a sample herd that improves over time. For example, an old bull whose statistics where considered good twenty years ago might not be interesting anymore compared to a younger bull whose genetics have been properly selected. Moreover, we need the genealogy of those bulls to check that no kinship link exists between the bull and the cow we want to inseminate. It is easy to get lost on the kin tree, so a by hand solution is not a good solution. It is interesting to automate this process by retrieving the data from a website.

The Canadian Dairy Network (CDN) is the generic website recommended by our farmer. Their mission is "to provide excellence and leadership in dairy herd improvement through an efficient information infrastructure and quality genetic evaluation services". The six mandates of CDN are the following:

- The calculation and publication of genetic evaluations for all dairy breeds in Canada as well as related services including the extension of information and research results.
- The coordination of industry-supported research in the area of dairy cattle improvement.
- The coordination of industry standards related to publishable lactation records, information required for genetic evaluations and Publication Code of Ethics.
- The ongoing development and maintenance of the Data Exchange System for the dairy cattle improvement industry.
- Act as a catalyst to identify opportunities and encourage closer collaboration among its member organizations, even to the point of integration.
- Act as a catalyst to identify opportunities and encourage closer collaboration among its member organizations, even to the point of integration.
- Provide services to industry partner organizations and international bodies that take advantage of core competencies and provide benefit to member organizations and/or Canadian dairy producers.

The website is quite old and steady. This is good and bad news: the good part is that the layout is very unlikely to change; the bad part is that the HTML code is not cooperative. In the next sections, we detail how the website is organized and how we built tools to fetch information from it, as well as the problems we met during the process.

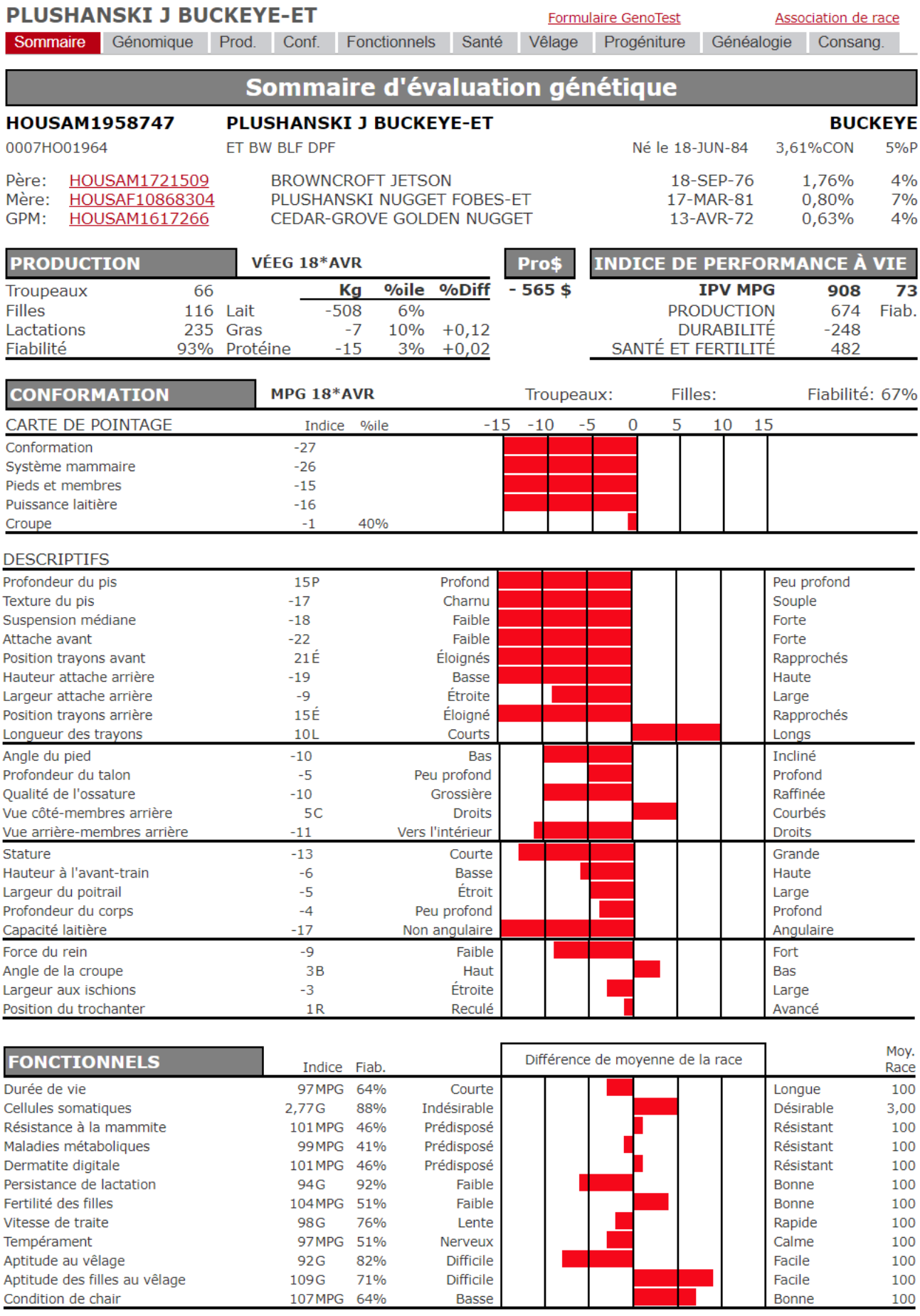
3.2.1 Organisation of a bull card on cdn.ca

CDN provides us with every information we need about the bulls' statistics and genealogy. Figure 3.3 shows the summary tab of a bovine card. The card is topped with the full name of the bovine, here "Plushanski J Buckeye-et". The summary tab starts with general information about the bovine such as their referencing number (an id specific to the website), their full and short name, birth date and links to the father, the mother and the maternal grandfather cards. Then the data are displayed in tables. We can find general data in the "carte de pointage" table, data about the physique of the animal in the "descriptifs" table, and data about the aptitudes in the "fonctionnels" table. We used some of those data according to the constrain we implemented.

The bovine card also contains a genealogy tab, as you can see in figure 3.4. The kin tree is horizontal with the fathers on the upper branch and the mothers on the bottom branch.

The HTML code is not the most cooperative one: every info we need is structured in table tags having little to no meta-data to help us identify easily the content of the table cell. A look into the HTML code is thus mandatory to find the good way to extract any data. For example, the referencing number, full and short names all have the class "animalHeadersubTitle" and nothing helps us to identify which one is what except their position. In some cases, we have to use the text that is printed to find the info we look for. The data we are looking for might be within the same tag, or the next tag. For example, to extract the birth date, we look for the keyword "Né" and extract the data from the sentence "Né le 18-JUN-84". For the somatic cells on the contrary, we look for the tag containing the keywords "Cellules somatiques" and the actual information is located in the following <td> tag.

Therefore there is no other way than to hard-code the way to access to the data we want. This raises a problem: our web parser would become obsolete if CDN decided to update the layout of the bovine cards or use a different appellation for some criterion. Fortunately, this is very unlikely to happen as the website hasn't changed in years.



PLUSHANSKI J BUCKEYE-ET

Formulaire GenoTest | Association de race

Sommaire | Génomique | Prod. | Conf. | Fonctionnels | Santé | Vêlage | Progéniture | Généalogie | Consang.

Sommaire d'évaluation génétique

HOUSAM1958747	**PLUSHANSKI J BUCKEYE-ET**			**BUCKEYE**
0007HO01964	ET BW BLF DPF	Né le 18-JUN-84	3,61%CON	5%P
Père: HOUSAM1721509	BROWNCROFT JETSON	18-SEP-76	1,76%	4%
Mère: HOUSAF10868304	PLUSHANSKI NUGGET FOBES-ET	17-MAR-81	0,80%	7%
GPM: HOUSAM1617266	CEDAR-GROVE GOLDEN NUGGET	13-AVR-72	0,63%	4%

PRODUCTION — VÉEG 18*AVR

			Kg	%ile	%Diff
Troupeaux	66				
Filles	116	Lait	-508	6%	
Lactations	235	Gras	-7	10%	+0,12
Fiabilité	93%	Protéine	-15	3%	+0,02

Pro$ - 565 $

INDICE DE PERFORMANCE À VIE

IPV MPG	**908**	**73**
PRODUCTION	674	Fiab.
DURABILITÉ	-248	
SANTÉ ET FERTILITÉ	482	

CONFORMATION — MPG 18*AVR — Troupeaux: — Filles: — Fiabilité: 67%

CARTE DE POINTAGE	Indice	%ile
Conformation	-27	
Système mammaire	-26	
Pieds et membres	-15	
Puissance laitière	-16	
Croupe	-1	40%

DESCRIPTIFS

Profondeur du pis	15P	Profond	Peu profond
Texture du pis	-17	Charnu	Souple
Suspension médiane	-18	Faible	Forte
Attache avant	-22	Faible	Forte
Position trayons avant	21É	Éloignés	Rapprochés
Hauteur attache arrière	-19	Basse	Haute
Largeur attache arrière	-9	Étroite	Large
Position trayons arrière	15É	Éloigné	Rapprochés
Longueur des trayons	10L	Courts	Longs
Angle du pied	-10	Bas	Incliné
Profondeur du talon	-5	Peu profond	Profond
Qualité de l'ossature	-10	Grossière	Raffinée
Vue côté-membres arrière	5C	Droits	Courbés
Vue arrière-membres arrière	-11	Vers l'intérieur	Droits
Stature	-13	Courte	Grande
Hauteur à l'avant-train	-6	Basse	Haute
Largeur du poitrail	-5	Étroit	Large
Profondeur du corps	-4	Peu profond	Profond
Capacité laitière	-17	Non angulaire	Angulaire
Force du rein	-9	Faible	Fort
Angle de la croupe	3B	Haut	Bas
Largeur aux ischions	-3	Étroite	Large
Position du trochanter	1R	Reculé	Avancé

FONCTIONNELS

	Indice	Fiab.		Différence de moyenne de la race		Moy. Race
Durée de vie	97MPG	64%	Courte		Longue	100
Cellules somatiques	2,77G	88%	Indésirable		Désirable	3,00
Résistance à la mammite	101MPG	46%	Prédisposé		Résistant	100
Maladies métaboliques	99MPG	41%	Prédisposé		Résistant	100
Dermatite digitale	101MPG	46%	Prédisposé		Résistant	100
Persistance de lactation	94G	92%	Faible		Bonne	100
Fertilité des filles	104MPG	51%	Faible		Bonne	100
Vitesse de traite	98G	76%	Lente		Rapide	100
Tempérament	97MPG	51%	Nerveux		Calme	100
Aptitude au vêlage	92G	82%	Difficile		Facile	100
Aptitude des filles au vêlage	109G	71%	Difficile		Facile	100
Condition de chair	107MPG	64%	Basse		Bonne	100

Figure 3.3: Example of a bull card summary tab on CDN.ca

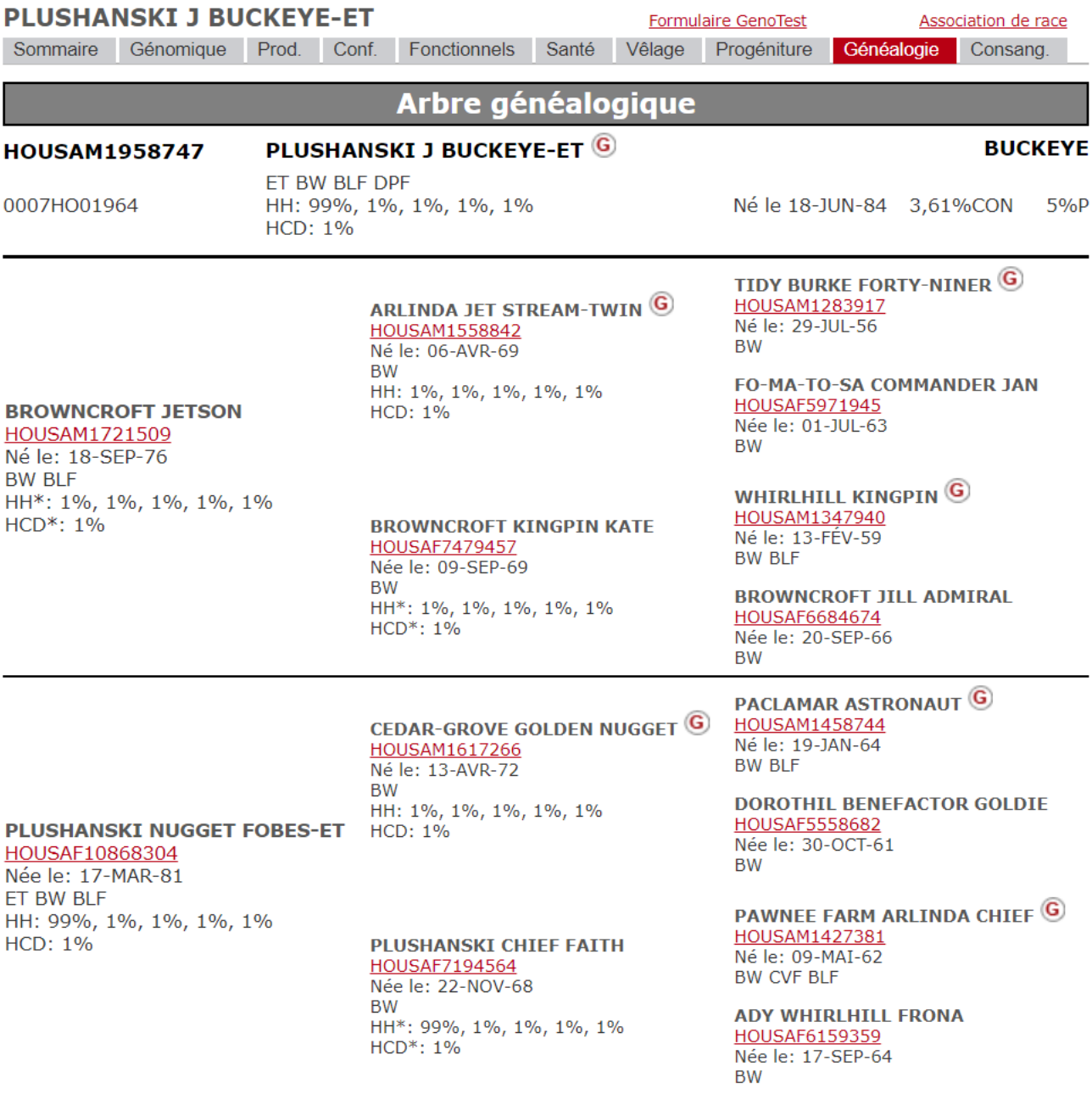


Figure 3.4: Example of a bull card genealogy tab on CDN.ca

3.2.2 Operation

The idea behind the web parser is to download the raw HTML code and search the obtained text to extract the information we need. In practice, we used a HTML Cleaner library [7] to achieve our goal. In this section, we present the different kind of parsers we implemented according to their usage. We also develop how they work.

Extract info parser

The extract info parser is the parser that help us extract information in the summary tab of the bovine card. It contains tree dictionaries: one for the elements that are going to get extracted with a class attribute in an HTML tag, one for those that use directly the text in between an HTML tag and one for the results. The keys are the keyword used to describe what we are looking for (an ID, a birth date, an info about somatic cells, etc.). The two first one have a KeyWordInfo object as values while the latter has a string representing the value we were looking for in the HTML code. KeyWordInfo is a class used to encode the information about how to extract one particular information from the website. It contains the current step initialised to -1, the keyword (either the class value or the text), the number of tag to ignore before the one we are looking for and a Boolean to know whether the element must be strictly equal to the keyword or just contain it. In the case of the keyword "Né", it is part of a sentence that contain the birth date, it is not a title. Hence, when we look for that keyword, we want to find text containing it but not strictly equal to it.

The part where those KeyWordInfo are created is hard-coded. We didn't want to let the user of our application fill those data himself because it require a technical background to fully understand how to find those data. After this creation, the HTMLCleaner library is used to extract the <td> tags from the HTML code of the page designated by the URL given. We then filter the list of tags to delete tags that simply display a space (" " in HTML). We run through the list of tags and look for the keywords. The treatment is not exactly the same if we are looking for a keyword as value to a class attribute or as text but it is very similar. We compare the value of the class attribute (resp. the text contained in the tag) with the keyword (it must be equal or just contained, according to the Boolean value in KeyWordInfo) and if it match, we set the current step to the goal number of steps. If it reaches zero then we have found the data we were looking for! We register the string data in the result dictionary with the corresponding key and delete the KeyWordInfo from its dictionary. If on the contrary the current step is bigger than zero we decrease it and continue to the next tag. When the elem and class dictionaries are empty, we have found all the data we needed.

We return a new object of the class "WebParsingResult" taking the result dictionary in parameter. This class handles the raw string we extracted and refactor them so that we can use them directly on the bovine object. We noticed that some of the bovine did not have a name in short notation in their card. Because of that, we extract the name in long notation from the card and use it when the short name is missing.

Bull Web Parser

In this parser, we parse the genealogy of the bull. We change the URL to access to the genealogy tab, then extract the <td> tags from the HTML code with the HTMLCleaner library [7]. We identify the one representing a parent on the kin tree (identified with the class attribute, either "animalUnderline" or "animalSmallText"). To identify which tag represent which parent, we have to rely on the position of each tag to find out the sex, so that we can create an instance of the right kind and which parent is in the kin tree. We look for individual information by using the extract info parser with the partial link given in an <a> tag.

3.2.3 Warnings and problems we met

We encountered a problem while working with the CDN website. It is protected against denial-of-service attacks. DoS attacks are cyber-attacks that aim at flooding an online resource so that it is not not available for its legit users anymore. It is performed by sending superfluous requests to the service in an attempt to overload the system and prevent legitimate requests to be answered. DoS attacks are performed from a single source contrarily to Distributed DoS attacks (DDoS). The former can be prevented by blocking the access to a single user whose behaviour is suspicious, and this is exactly what happened to us. The automatic process to recover the genealogy of the bull necessitate a lot of navigation on the Canadian Dairy Network and this action seemed suspicious, so we lost access to the website after some time.

The solution to get around this security is to change the IP address of the router, or partition the recovery of the data in a way that would not trigger the security. We chose the latter because it is less technical. It thus takes us more time to obtain the data than we expected, but this is not really a problem.

3.3 Storage of the data

We chose not to use a Database Management System (DBMS) for this project. The main reason behind this decision it that it was not necessary. We simply represented the data with class objects and use instances of those classes as data. They are stored in dictionaries with their keys (name for the bulls, IDs for the cows), which makes the retrieval with the key easy and fast.

We have modelled the animals through a super class called "Bovine" and two classes extending it, "Bull" and "Cow". The super class contains all the information that are common to the two genders: double IDs, name, birth date, coat colour, the conformation data, the somatic cell count in the format "10^3 SCC", and the final rates of fat and protein. It also contains the links to the mother and the father(s). The gender specific classes are used to register information that are specific to the gender. For the female, we find the number of calving and the count of difficult calving. We also find helpers to compute the final rates of fat and protein, being a mean of all the rates extracted from the "Sample" file. For the males, we have their calving quality rate and fat and protein indexed percentages. Each of the classes also have a custom toString method as well as other useful methods. Storing data as instances of those classes works fine.

Nonetheless, reloading and process again every file at each use of the program is redundant and not performing at all. It would be a huge problem with the retrieval of the bull data, due to the protection of the Canadian Dairy Network against DoS attack. To avoid that, we serialized our data.

Serialization is the process of translating data structures or object state into a format that can be stored (for example, in a file or memory buffer) or transmitted (for example, across a network connection link) and reconstructed later (possibly in a different computer environment). When the resulting series of bits is reread according to the serialization format, it can be used to create a semantically identical clone of the original object [8]. In Java (and thus Scala), the Serializable interface must be implemented by the classes that we want to serialize. Note that every subclass of a class that implement the Serializable interface are also serializable. The interface has no methods or fields and serves only to identify the semantics of being serializable [9]. In this project, the Bovine super class is serializable so is the entire database.

Chapter 4

Constrained model

4.1 Central constraint - Consanguinity

4.1.1 In the real world

As stated in [10], consanguinity can be defined as "the probability that two kin alleles of an individual, located on the same locus, are identical by descendants". Dairy bovines have thirty pairs of chromosomes and each gene have a designated place, called locus, on each chromosome. Many shapes exist for each gene, called allele, which will be expressed as an alternative shape of a characteristic. When an animal has two identical alleles, it is considered heterozygous for that gene whilst those with different alleles for the gene are called homozygous. Animals with common ancestors have a greater probability to have inherited an identical allele than those having unrelated parents (see figure 4.1). Consanguinity represents the probability that a beast inherited a same gene from its parents. It increases when the parents have common ancestors. It has been proven that the increase of consanguinity tends to reduce the zoo-technique [1] capacities of the individual. Consanguinity is thus something we want to avoid.

In the field of breeding, a genetic selection is performed when selecting the best male and female as the parents of the next generation. The consequences are such that the next generation is bounded to have a kinship relation with the elite animals of today. The solution is to look for a good trade-off between the consanguinity and the genetic gain. The probability of consanguinity decreases with generations. We used the figure 4.2 to compute the consanguinity probability of our animals. Consanguinity is identified whenever a same animal appears in the genealogy of both parents. The rate depends on the level on which we find this animal on each branch. The columns in the table of the figure 4.2 represents one of the parent's genealogy levels and the rows the genealogy levels of the other parent. The table is symmetrical. Level I represents the direct parents, level II the grandparents and it goes like this until the fifth generation at level V.

The goal of this constraint is to limit the consanguinity rate of the calves that are going to be produced when the bull X is selected to inseminate the cow Y, and such for the whole lot of reproducing cows in the herd and for a selection of bulls.

4.1.2 Computation of the consanguinity rate of a pair cow-bull

The first step to modelize this constraint is to compute the consanguinity rate of an hypothetical animal, resulting of the mating of a cow c and a bull b. To do so, we need the kin trees of the parents to explore their genealogy. Thanks to the given data detailed in chapter 3, we were able

[1] Zootechny : the scientific art of maintaining and improving animals under domestication including breeding, genetics, nutrition, and housing : the technology of animal husbandry

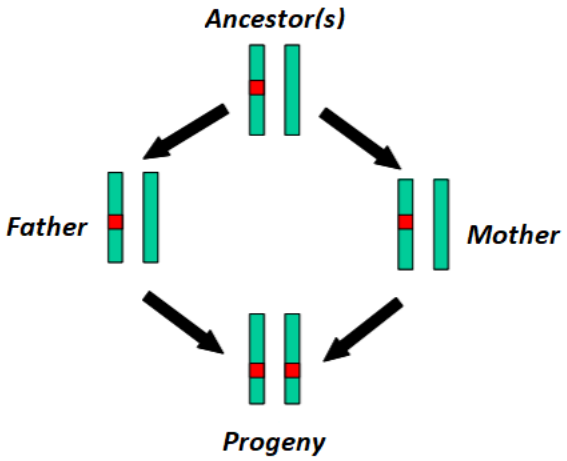


Figure 4.1: Heredity of an allele with common ancestors (from [10])

N° de la génération	I	II	III	IV	V
I	/	25%	12,5%	6,25%	3,125%
II	25%	12,5%	6,25%	3,125%	1,5625%
III	12,5%	6,25%	3,125%	1,5625%	0,7813%
IV	6,25%	3,125%	1,5625%	0,7813%	0,3906%
V	3,125%	1,5625%	0,7813%	0,3906%	0,1953%

Figure 4.2: Consanguinity rates according to the generation level (from [11])

to construct a graph structure in which every animal has references to its parents if they are known.

We explore the kin tree in a Breadth First Search (BFS), level by level, each level representing a generation. Whenever we find a beast that has already been seen in the graph, or in more technical words we find a cycle in the genealogy graph of the hypothetical calf, we have found consanguinity. We compute its rate with the help of figure 4.2, depending on the level on which the guilty bovine appears. In figure 4.3, Marguerite has a rate of 1.5625% and T-Bruno has a rate of 6.25%. The final consanguinity rate of the hypothetical calf is the sum of all the consanguinity rates found in its genealogy graph, here 7.8125%. When consanguinity is found, we stop exploring the genealogy of the guilty bovine, supposing that it has a consanguinity rate of zero. In figure 4.3, the nodes in red are the nodes that are never explored because consanguinity has been found regarding their descendant. The exploration of the tree is stopped at the fifth generation because the consanguinity rates are negligible above that.

Deal with the uncertainty about the father

As stated in the section 3.1.1, we are sometimes unsure about the identity of the father of an animal. How to deal with this uncertainty when we compute the consanguinity rate? By being pessimistic. We explore the genealogy of each of the potential fathers, looking for consanguinity and we use the worst rate found for the computation of the final sum.

4.1.3 Strategies

Different strategies can be applied when we look for the best consanguinity rate for a group of bovines composed of C cows and B bulls.

Average The objective is to minimize the average cost of the herd. The sum of the costs associated with each match is to be minimized to reach that goal. With this strategy, some really bad

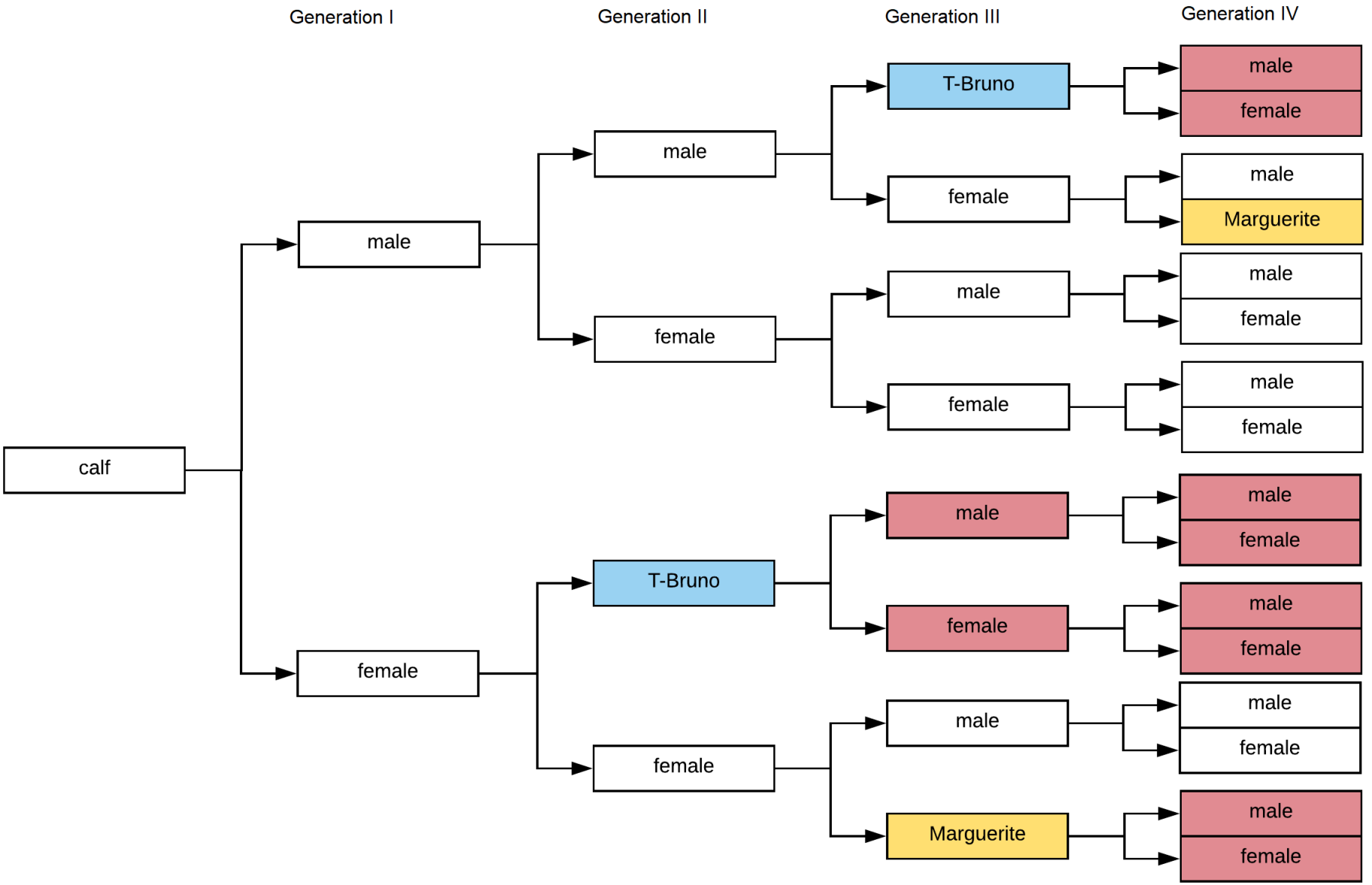


Figure 4.3: Kin tree of a hypothetical calf with two cases of consanguinity

consanguinity rates might be accepted in a solution, as long as the average is the minimum. To have a guarantee that no rates exceed some value, one of the other strategy must be used.

Threshold A threshold of the worst level acceptable is decided. Every bull exceeding this rate with any of the cows is removed from the domain of that cow. With this strategy, we have a guarantee that each of the bulls left in the domain of the cows is acceptable regarding the consanguinity rate. The objective after that is to minimize the sum of the costs of the cows, just like in the "Average" strategy. This strategy enable the "hard" mode of the consanguinity constraint, as some values might be discarded from the domains.

Worst The match having the worst consanguinity rate is used to summarize the total cost of the herd. By minimization, we assert that no rate will exceed this value. This is the soft version of the "Threshold" strategy, as no value is rejected from the domains.

See section 5.1.2 for more information about the hard and soft constraints.

4.1.4 Modelization

We have:

- C the set of cows
- B the set of bulls
- A matrix of size $|C| * |B|$ filled with the consanguinity rates. Cell $cell_{cb}$ contains the value of the rate of the pair (c, b) with $c \in C$ and $b \in B$.
- M the maximum number of selected bulls from the total set of bulls.
- A parameter $Strategy$ taking its value in the set $\{"Average", "Worst", "Threshold"\}$. See section 4.1.3 for more information.

We modelize the problem like this:

- V the set of $|C|$ variables, each of them representing a cow from the set of cows. The initial domain $dom(v)$ of variable $v \in V$ is the range $[1, |B|]$, each of the values representing one of the bull $b \in B$.
- $|C|$ intermediate variables i_v that represent the cost associated with each cow depending on its value, the bull that has been chosen for it.
- A variable $TotalCost$ representing the consanguinity rate of the herd. The way its value is computed depends on the chosen strategy (see section 4.1.3).

The constraints are:

1. $|C|$ constraint element [12] [13], linking the consanguinity cost found in $cell_{cb}$ with the intermediate variable i_v, where c is the cow represented by the variable v and b the bull represented by the value of the variable v.
2. $Strategy \neq "Worst" \implies \sum_{v \in V} i_v = TotalCost$. A constraint sum [14] [15] states that the variable $TotalCost$ must be equal to the sum of the intermediate variables when the strategy is "Average" or "Threshold".
3. $Strategy = "Worst" \implies TotalCost = \texttt{max}(i_v)$. When the strategy is "Worst", we summarize the cost of the herd by the worst consanguinity rate, the greatest.

4. $|\{dom(v_1) \cup dom(v_2) \cup ... \cup dom(v_n)\}| \leq M, \forall v_1, v_2, ..., v_n \in V$. This constraint is also known as the AtMostNValue constraint [16] [17]. The size of the union of the domains must be smaller or equal to M. With this constraint, we assert that the number of bulls selected does not surpass the given number M.

The $TotalCost$ variable is to be minimized in order to optimize the consanguinity rate according to the strategy.

4.2 Security constraint - Difficulty of calving

4.2.1 In the real world

A lot of birth happen in the farm every year. It is better when the pregnancy goes smoothly as well as the birth because it costs less. The mother and/or the calf could die, which is an evident loss of money, but difficult calving might also induce veterinary costs, and even a reduced herd fertility due to the longer time required by the mother to conceive again, as stated in [18]. We know that the ability of calving might be genetic. Inadequate size of birth canals and excessive calf size at birth are influencing factors. Hence, large pelvic areas are considered better for calving. Others causes of calving difficulties include:

- abnormal calf presentation
- obstruction of the birth canal by fat deposit
- constriction of the birth canal at the vagina, vulva or cervix
- weak labour or poor muscles tone in cows that are either very thin or too fat
- multiple births like twins or triplets

We need data about each of the bovines aptitudes in order to do some computation and minimize the calving difficulty of the future calves we are going to breed. With the cows of the herd, this value is computed from experience. As stated in section 3.1.1, we have records of the births that are considered "difficult". For the heifers, the default value of difficult calving is one. For the bulls that were breeded in the farm, we simply assign them the rate of their mother. For the reproducing bulls, CDN provides us with this information under the name "Aptitude au vêlage" in the "fonctionnels" table. The mean value is always 100 so bulls whose value is below the mean demonstrate calving difficulties and those having it above the mean don't.

4.2.2 Modelization

We model the difficulty of calving with a cost associated with each pair cow-bull. If the father is considered easy then the cost is null. Otherwise, the cost is based on the calving difficulty of the cow and its mother's.

We have :

- C the set of cows
- B the set of bulls
- $f(c)$ the number of difficult calving of the cow, by default 1 when the cow is a heifer. $c \in C$
- $g(b)$ the difficulty of calving of the bull $b \in B$. Equals either "DIFFICULT" or "EASY"

- $mother(c)$ representing the mother of the cow $c \in C$
- A matrix of size $|C| * |B|$ filled with the difficulty of calving value. Cell $cell_{cb}$ contains the value of the pair (c, b) with $c \in C$ and $b \in B$.

The formula to compute the difficulty of calving cost of the pair (c, b):

$$\texttt{difficultyOfCalving(c, b)} = \begin{cases} 0 & \texttt{if } g(b) = \texttt{ EASY} \\ \frac{f(c)+f(mother(c))}{2} & \texttt{if } g(b) = \texttt{ DIFFICULT} \end{cases}$$

We modelize the problem like this:

- $|C|$ intermediate variables i_v that represent the cost associated with each cow depending on its value, the bull that has been chosen for it.
- A variable $TotalCost$ representing the sum of the difficulty of calving values for each cow.

The constraints are:

1. $|C|$ constraint element [12] [13], linking the difficulty of calving cost found in $cell_{cb}$ with the intermediate variable i_v, where c is the cow represented by the variable v and b the bull represented by the value of the variable v.
2. $\sum_{v \in V} i_v = TotalCost$. A constraint sum [14] [15] states that the variable $TotalCost$ must be equal to the sum of the intermediate variables.

The $TotalCost$ variable is to be minimized in order to optimize the facility of calving of the herd.

4.3 Quality of milk constraints - Somatic cells

4.3.1 In the real world

A milk of good quality is the key for profitability in the dairy industry. To evaluate the quality a count of the somatic cells, a good quality indicator, is performed on the milk before the sale. It reflects the general health of the herd.

Somatic cells are composed of various type of white blood cells that aim at protecting the cow in case of infections, increasing in number to fight it when necessary. A high number of somatic cells is thus a warning that something is wrong [19].

Somatic cells are expressed in number of cell by millilitre. A milk delivery is acceptable if the somatic cell rate is between 0 and 400 000 cells/mL. Beyond that limit, there are penalties. For the cows of the herd, the mandatory data are provided by the file "Sample" (see section **??**). For the reproducing bulls, the information is extracted with the web parser (see section 3.2).

There exists different formats of data. The farmer uses the " 10^3 SCC" format. SCC stands for "Somatic Cells Count" [20]. To convert the value c, obtained in the given "Sample" file (see section 3.1.5), to the official "SCC" format, we need to use this equivalence equation:

$$c' = 10^3 c$$

CDN uses another format called "Somatic Cells Score" or "SCS". The converting equation from one format to the other is the following:

	$\frac{c+b}{2} \leq 300$	$\frac{c+b}{2} \in]300; 400]$	$\frac{c+b}{2} \geq 400$
$c \in [0; 400]$	0	$\left(\frac{\frac{c+b}{2}-300}{500}\right)$	$\texttt{min}\left(4, \frac{\frac{c+b}{2}-300}{500}\right)$
$c > 400$	0	$\texttt{min}\left(4, \frac{\frac{c+b}{2}-300}{500}\right)$	$\texttt{min}\left(4, \frac{\frac{c+b}{2}-300}{500}\right)$

Table 4.1: Costs of the somatic cell constraint when the constraint is soft

	$\frac{c+b}{2} \leq 300$	$\frac{c+b}{2} \in]300; 400]$	$\frac{c+b}{2} \geq 400$
$c \leq 300$	0	$\perp$	$\perp$
$c \in]300; 400]$	0	$\left(\frac{\frac{c+b}{2}-300}{500}\right)$	$\perp$
$c > 400$	0	$\texttt{min}\left(4, \frac{\frac{c+b}{2}-300}{500}\right)$	$\texttt{min}\left(4, \frac{\frac{c+b}{2}-300}{500}\right)$

Table 4.2: Costs of the somatic cell constraint when the constraint is hard

$$SCS = \log_2\left(\frac{SCC}{100000}\right) + 3$$

Or equivalently:

$$SCC = 100000 * 2^{SCS-3}$$
$$10^3 SCC = 100 * 2^{SCS-3}$$

We used the latter to convert the data fetched by the web parser, which means the data of the bulls, into the custom SCC format, the 10^3 SCC.

4.3.2 Modelization

In practice, we compute the hypothetical somatic cell rate of the future calf of a pair cow-bull by computing the mean of the rates of the two bovines. We want to keep it as low as possible. To do so, we have different kind of penalties according to the rate of the female. We have c the somatic cell rate of the cow and b the bull's, both in 10^3 SCC. Another criterion deciding of the penalties is the importance we accord to this constraint. It may be super important to take count of the somatic cells, then the constraint must be hard and the penalties used are detailed in the table 4.2. The $\perp$ sign means that the bull b must be discarded for the cow c because the hard constraint won't allow such combination. The soft constraint on the other hand is more accepting of bad results, in the case where we want to accord some attention to the somatic cell counts but not to the point where we risk to not be able to find any solution. Table 4.1 shows the penalties in such a situation.

We modelize the problem like this:

- $|C|$ intermediate variables i_v that represent the somatic cell cost associated with each cow depending on its value, the bull that has been chosen for it.
- A variable $TotalCost$ representing the sum of the somatic cell rates for each cow.

The constraints are:

1. $|C|$ constraint element [12] [13], linking the penalties found in table 4.1 if the constraint is soft, in table 4.2 if the constraint is hard, with the intermediate variable i_v, where c is the cow represented by the variable v and b the bull represented by the value of the variable v.

	b < 0	b ≥ 0
c < M	4 / ⊥	0
c ≥ M	2	0

Table 4.3: Costs of the protein and fat constraints

2. $\sum_{v \in V} i_v = TotalCost$. A constraint sum [14] [15] states that the variable $TotalCost$ must be equal to the sum of the intermediate variables.

The $TotalCost$ variable is to be minimized in order to optimize the somatic cell rate of the herd.

4.4 Quality of milk constraints - Proteins and fat

4.4.1 In the real world

The nutritious quality of the milk can be partially expressed with the protein and fat rates. Here, we hope to at least hold the quality we had before. We have the rates of our cows, given by the file "Sample" (see section 3.1.5). For the bulls, that obviously don't produce milk, the rates extracted from CDN describe the difference between the rates of the daughters of the bull and the average rate of a sample herd. If the daughters produce less than the rate observed, the difference is negative, otherwise positive.

Again, we associate costs with each cow-bull pair according to their rates. C is the set of cows we have in the herd. Here, the cow $c \in C$ is compared to the mean rate of the herd $Mean$. Table 4.3 shows how to compute the costs. When the $\perp$ sign appears, it means that the solution must reject the bull for the cow c whenever the constraint is hard. With soft constraint, we try and minimize the sum of the costs for each cow.

4.4.2 Modelization

4.5 Conformation constraints

Conformation constraints focus on the physique of the cows because some features are disadvantageous. The milking process is completely automated, so we need to make sure the cows are fit for the machines. Characteristics of the udder, the teat, the legs have their importance. The milking process is done on a rotating platform called a carousel where the milker automatically plugs itself on the udder of the cow. It is thus important that the legs of the cows allow the passage of the milker. Figure 4.4 shows different observations for the side and back view of the limbs. Cow-hocked bovines are a problem to be milked with the carousel.

Each of the features observed in the Conformation file (see section 3.1.6) has its importance. During the selection of the best bull for a particular cow, we aim at reaching some balance in each of the features. For example, a cow having a good mammal system could be inseminate with any bull without taking this criterion into consideration. On the contrary, if the mammal system is bad, it is best to match it with a bull whose mammal system score is high, with the hope that it will balance the problem for the calf.

We have been given the penalty to associate with the every match according to the scores of each of the bovines. Those can be found on the next sections.

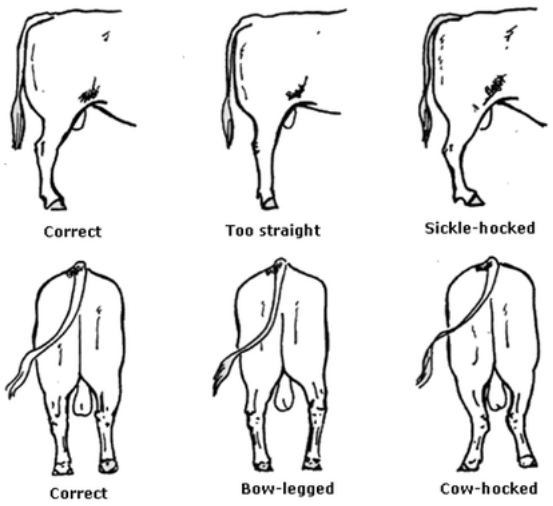


Figure 4.4: Side and back views of the limbs [21]

4.5.1 Mammal system

Let's have a cow and a bull. We associate a penalty when the combination of the scores of the mammal system is not good. Here, c represents the value of the mammal system of the cow, b the value of the bull. The penalties are positive or nil integers. In the table 4.4, when the $\perp$ sign appears, it means that the bull must be discarded from the domain of the variable representing the cow when the constraint is set to "hard" mode. When the "soft" mode is set, only the penalties are used. The objective is to minimize the sum of the penalties for each couple of the solution.

	$b \leq -5$	$b \in]-5;0]$	$b = 0$	$b \in]0;5[$	$b \geq 5$
$c = -2$	4 / $\perp$	3 / $\perp$	2	1	0
$c = -1$	3 / $\perp$	2	1	0	0
$c = 0$	2	1	0	0	0
$c = 1$	1	0	0	0	0
$c = 2$	0	0	0	0	0

Table 4.4: Costs of the mammal system constraint

4.5.2 Udder depth

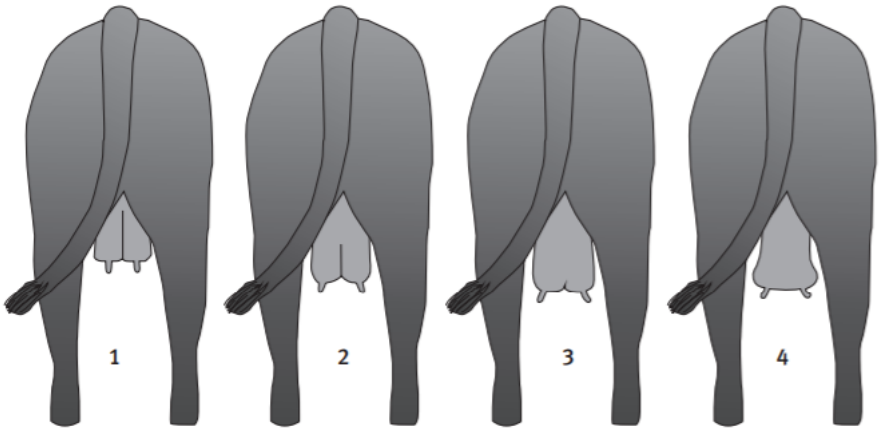
Let's have a cow and a bull. We associate a penalty when the combination of the scores of the udder depth is not good. Here, c represents the value of the udder depth of the cow, b the value of the bull. The penalties are positive or nil integers. In the table 4.5, when the $\perp$ sign appears, it means that the bull must be discarded from the domain of the variable representing the cow when the constraint is set to "hard" mode. When the "soft" mode is set, only the penalties are used. The objective is to minimize the sum of the penalties for each couple of the solution.

4.5.3 Median suspension

"Median suspension" refers to the median suspensory ligament providing various level of udder support, as seen in figure 4.5. Note that the number related to each figure does not correspond to the score attributed by our farmer.

	$b \leq -5$	$b \in]-5;0]$	$b = 0$	$b \in]0;5[$	$b \geq 5$
$c = -2$	4 / $\perp$	3 / $\perp$	2	1	0
$c = -1$	3 / $\perp$	2	1	0	0
$c = 0$	1	0	0	0	1
$c = 1$	0	0	1	2	3 / $\perp$
$c = 2$	0	1	2	3 / $\perp$	4 / $\perp$

Table 4.5: Costs of the udder depth constraint



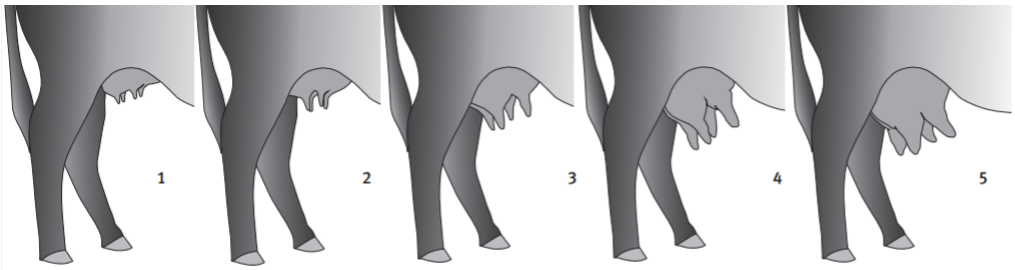
Drawing 1: Prominent median suspensory ligament holds the udder tight to the body cavity. Teats are suspended perpendicular to the ground.
Drawing 2: Intermediately prominent median suspensory ligament suspends the udder farther from the body cavity. The udder is suspended about level with the hock and almost perpendicular to the ground.
Drawing 3: With a very weak median suspensory ligament, the udder and teats are suspended below the hock. When the udder and teats are engorged with milk, teats splay outward.
Drawing 4: The median suspensory ligament is absent. The udder and teats are suspended below the hocks. The udder balloons and teats splay outward.

Figure 4.5: The effect of the median suspensory ligament on the udder support [22]

Let's have a cow and a bull. We associate a penalty when the combination of the scores of the median suspension is not good. Here, c represents the value of the median suspension of the cow, b the value of the bull. The penalties are positive or nil integers. In the table 4.6, when the $\perp$ sign appears, it means that the bull must be discarded from the domain of the variable representing the cow when the constraint is set to "hard" mode. When the "soft" mode is set, only the penalties are used. The objective is to minimize the sum of the penalties for each couple of the solution.

	$b \leq -5$	$b \in]-5;0]$	$b = 0$	$b \in]0;5[$	$b \geq 5$
$c = -2$	4 / $\perp$	3 / $\perp$	2	1	0
$c = -1$	3 / $\perp$	2	1	0	0
$c = 0$	2	1	0	0	0
$c = 1$	1	0	0	0	0
$c = 2$	0	0	0	0	0

Table 4.6: Costs of the median suspension constraint



Drawing 1: Teat size is very small and symmetrical. Teat size score = 9.
Drawing 2: Teat size is small and symmetrical. Teat size score = 7.
Drawing 3: Teat size is intermediate in length; still have symmetry. Teat size score = 5.
Drawing 4: Teat size is large, variable in length and symmetry. Teat size score = 3.
Drawing 5: Teat size is very large, variable in length and symmetry. Teats appear to be thick. Teat size score = 1.

Figure 4.6: Teat length

4.5.4 Position of the teats, front and back

Let's have a cow and a bull. We associate a penalty when the combination of the scores of the position of the front teats (resp. back teats) is not good. Here, c represents the value of the front teats (resp. back teats) position of the cow, b the value of the bull. The penalties are positive or nil integers. In the table 4.7, when the $\perp$ sign appears, it means that the bull must be discarded from the domain of the variable representing the cow when the constraint is set to "hard" mode. When the "soft" mode is set, only the penalties are used. The objective is to minimize the sum of the penalties for each couple of the solution. Note that there are two different constraints described here: one for the front teats and one for the back teats. Both behave the exact same way so we chose to use only one table to represent them.

	$b \leq -5$	$b \in]-5;0]$	$b = 0$	$b \in]0;5[$	$b \geq 5$
$c = -2$	4 / $\perp$	3 / $\perp$	2	1	0
$c = -1$	3 / $\perp$	2	1	0	0
$c = 0$	1	0	0	0	1
$c = 1$	0	0	1	2	3 / $\perp$
$c = 2$	0	1	2	3 / $\perp$	4 / $\perp$

Table 4.7: Costs of the front/back teat constraints

4.5.5 Teat length

Let's have a cow and a bull. We associate a penalty when the combination of the scores of the teat length is not good. Here, c represents the value of the teat length of the cow, b the value of the bull. The penalties are positive or nil integers. In the table 4.8, when the $\perp$ sign appears, it means that the bull must be discarded from the domain of the variable representing the cow when the constraint is set to "hard" mode. When the "soft" mode is set, only the penalties are used. The objective is to minimize the sum of the penalties for each couple of the solution.

Figure 4.6 shows representations of some kind of observable teat lengths. Note that the scores attributed in the figure do not correspond to the scores given by our farmer. The figure is taken from documentation about meat cows [22], thus the scoring depends on other criterion.

4.5.6 Limbs

Let's have a cow and a bull. We associate a penalty when the combination of the scores of the limbs is not good. Here, c represents the value of the limbs of the cow, b the value of the bull.

	$b \leq -5$	$b \in]-5;0]$	$b = 0$	$b \in]0;5[$	$b \geq 5$
$c = -2$	4 / $\perp$	3 / $\perp$	2	1	0
$c = -1$	3 / $\perp$	2	1	0	0
$c = 0$	1	0	0	0	1
$c = 1$	0	0	1	2	3 / $\perp$
$c = 2$	0	1	2	3 / $\perp$	4 / $\perp$

Table 4.8: Costs of the front/back teat constraints

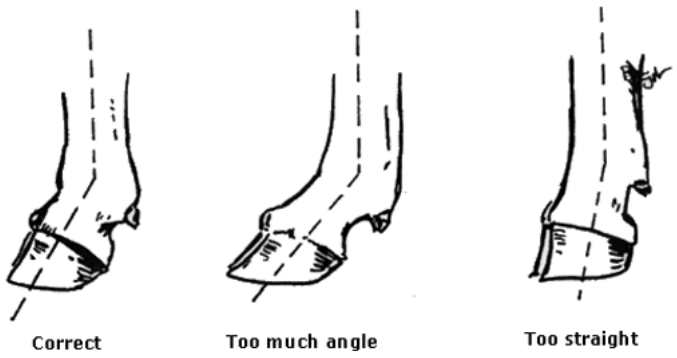


Figure 4.7: Foot angle [21]

The penalties are positive or nil integers. In the table 4.9, when the $\perp$ sign appears, it means that the bull must be discarded from the domain of the variable representing the cow when the constraint is set to "hard" mode. When the "soft" mode is set, only the penalties are used. The objective is to minimize the sum of the penalties for each couple of the solution.

	$b \leq -5$	$b \in]-5;0]$	$b = 0$	$b \in]0;5[$	$b \geq 5$
$c = -2$	4 / $\perp$	3 / $\perp$	2	1	0
$c = -1$	3 / $\perp$	2	1	0	0
$c = 0$	2	1	0	0	0
$c = 1$	1	0	0	0	0
$c = 2$	0	0	0	0	0

Table 4.9: Costs of the limbs constraint

4.5.7 Foot angle

Let's have a cow and a bull. We associate a penalty when the combination of the scores of the foot angle is not good. Here, c represents the value of the foot angle of the cow, b the value of the bull. The penalties are positive or nil integers. In the table 4.10, when the $\perp$ sign appears, it means that the bull must be discarded from the domain of the variable representing the cow when the constraint is set to "hard" mode. When the "soft" mode is set, only the penalties are used. The objective is to minimize the sum of the penalties for each couple of the solution.

4.5.8 Side view of the legs

Let's have a cow and a bull. We associate a penalty when the combination of the scores of the side view of the legs is not good. Here, c represents the value of the side view of the legs of the cow, b the value of the bull. The penalties are positive or nil integers. In the table 4.11, when the $\perp$ sign appears, it means that the bull must be discarded from the domain of the variable representing the cow when the constraint is set to "hard" mode. When the "soft" mode is set,

	$b \leq -5$	$b \in]-5;0]$	$b = 0$	$b \in]0;5[$	$b \geq 5$
$c = -2$	4 / $\perp$	3 / $\perp$	2	1	0
$c = -1$	3 / $\perp$	2	1	0	0
$c = 0$	1	0	0	0	1
$c = 1$	0	0	1	2	3 / $\perp$
$c = 2$	0	1	2	3 / $\perp$	4 / $\perp$

Table 4.10: Costs of foot angle constrain

only the penalties are used. The objective is to minimize the sum of the penalties for each couple of the solution. Figure 4.4 shows what the side view of the legs might represents in real life.

	$b \leq -5$	$b \in]-5;0]$	$b = 0$	$b \in]0;5[$	$b \geq 5$
$c = -2$	0	0	0	0	0
$c = -1$	0	0	0	0	1
$c = 0$	0	0	0	1	2
$c = 1$	0	0	1	2	3 / $\perp$
$c = 2$	0	1	2	3 / $\perp$	4 / $\perp$

Table 4.11: Costs of the side view limbs constrain

4.5.9 Back view of the legs

Let's have a cow and a bull. We associate a penalty when the combination of the scores of the back view of the legs is not good. Here, c represents the value of the back view of the legs of the cow, b the value of the bull. The penalties are positive or nil integers. In the table 4.12, when the $\perp$ sign appears, it means that the bull must be discarded from the domain of the variable representing the cow when the constraint is set to "hard" mode. When the "soft" mode is set, only the penalties are used. The objective is to minimize the sum of the penalties for each couple of the solution. Figure 4.4 shows what the back view of the legs might represents in real life.

	$b \leq -5$	$b \in]-5;0]$	$b = 0$	$b \in]0;5[$	$b \geq 5$
$c = -2$	4 / $\perp$	3 / $\perp$	2	1	0
$c = -1$	3 / $\perp$	2	1	0	0
$c = 0$	2	1	0	0	0
$c = 1$	1	0	0	0	0
$c = 2$	0	0	0	0	0

Table 4.12: Costs of the back view limbs constrain

4.5.10 Modelization

For each conformation constraint, we modelize the problem like this:

- $|C|$ intermediate variables i_v that represent the cost associated with each cow depending on its value, the bull that has been chosen for it. The costs are described in the corresponding table and may vary if the constraint is hard.
- A variable $TotalCost$ representing the sum of the costs attributed to each cow.

The constraints are:

1. $|C|$ constraint element [12] [13], linking the penalties found in corresponding table with the intermediate variable i_v, where c is the cow represented by the variable v and b the bull represented by the value of the variable v.
2. $\sum_{v \in V} i_v = TotalCost$. A constraint sum [14] [15] states that the variable $TotalCost$ must be equal to the sum of the intermediate variables.

We have a $TotalCost$ variable for each of the conformation constraints. Each of those are to be minimized in order to optimize the conformation of the animals in the herd.

4.6 Global Improvement constraints - Milk quantity

Global improvement constraints focus on the global improvement that can be achieved with the bulls that are selected. Obviously, increasing the milk quantity is an coveted goal. When the cows are milked, the liquid is put in a large jar capable of containing a many liters of milk, merging the product of every cow together. To earn more money, the farmer needs to fill the most jars as possible. We are interested in the global quantity obtained in the end; we consider all our cows at once here. Fortunately, the Canadian Dairy Network provide us information about the milking quantities produced by the daughters of the bulls. More precisely, it is the indexed value of the daughters: their values are compared to the mean value of a sample herd to see if they perform better or not.

We have:

- C the set of cows.
- b_c the bull selected to impregnate cow $c \in C$.
- $indexedMilk_{b_c}$ the indexed milk value of the bull b_c
- t a given threshold of the minimum quantity we want to have per cow. Its default value is 1000.

We want the sum of the values of the selected bulls to be greater or equal to the threshold multiplied with the number of cows to impregnate. The computation to perform is the following:

$$|C| * t \leq \sum_{c \in C} indexedMilk_{b_c}$$

4.6.1 Modelization

We have:

- C the set of cows
- $|C|$ intermediate variables called i_c
- $IndexedSum$ an intermediate variable
- t a given threshold of the minimum quantity we want to have per cow. Its default value is 1000.

The constraints are:

- $|C|$ element constraints [12] [13] to states that the i_c variable is equal to $indexedMilk_{b_c}$

- A Sum constraint [14] [15] to states that the variable $IndexedSum$ is equal to the sum of the variables $i_c \forall c \in C$
- An Element-GreaterOrEqual constraint [23] [24] to states that the $IndexedSum$ variable must be greater or equal to the threshold t multiplied by the number of cows in the set

4.7 Global Improvement constraints - Protein and fat rates

Global improvement constraints focus on the global improvement that can be achieved with the bulls that are selected. This is a quantity constraint: we want to make sure that the general rates of fat and proteins will at least stay as good as before, if this set of bull is chosen. For that, we need the indexed value of milk of each bull (the quantity of milk produced by its daughters in comparison to a sample herd) and the indexed percentage of difference in fat and protein. This value is a positive or negative percentage that indicates if the proportion of matter that the daughter have in their milk compared to a sample herd. Those values are provided by CDN in the "production tab".

We have:

- C the set of cows
- b_c the chosen bull for cow $c \in C$
- $indexedMilk_{b_c}$ the indexed value of the milk of the bull b_c
- fat_{b_c} the indexed percentage of fat for the bull b_c
- $protein_{b_c}$ the indexed percentage of protein of the bull b_c
- A the global improvement constant, expressed in percentage

By default A is set to 0, meaning we want to keep the rates as good as before at the very least. The computation is the following:

$$\sum_{c \in C} indexedMilk_{b_c} * A \leq \sum_{c \in C} (indexedMilk_{b_c} * fat_{b_c})$$

$$\sum_{c \in C} indexedMilk_{b_c} * A \leq \sum_{c \in C} (indexedMilk_{b_c} * protein_{b_c})$$

4.7.1 Modelization

We have:

- C the set of cows
- $|C|$ intermediate variables called $previousValue_c$
- $|C|$ intermediate variables called $nextProteinValue_c$
- $|C|$ intermediate variables called $nextFatValue_c$
- $indexedMilk_{b_c}$ The indexed milk value of the bull b_c chosen for the cow c
- fat_{b_c} the indexed percentage of difference in protein for the bull b_c chosen for the cow c
- $protein_{b_c}$ the indexed percentage of difference in fat for the bull b_c chosen for cow c

- $PreviousSum$ an intermediate variable
- $NextProteinSum$ an intermediate variable
- $NextFatSum$ an intermediate variable
- A the global improvement constant, expressed in percentage

The constraints are:

- $|C|$ element constraints [12] [13] stating the equality between the intermediate variable $previousValue_c$ and the constant $indexedMilk_{b_c}$
- $|C|$ element constraints [12] [13] stating the equality between the intermediate variable $nextFatValue_c$ and the constant $indexedMilk_{b_c}$ multiplied by the constant fat_{b_c}
- $|C|$ element constraints [12] [13] stating the equality between the intermediate variable $nextProteinValue_c$ and the constant $indexedMilk_{b_c}$ multiplied by the constant $protein_{b_c}$
- A sum constraint [14] [15] to state that the $PreviousSum$ variable is equal to the sum of the $previousValue_c$ for each $c \in C$
- A sum constraint [14] [15] to states that the $NextProteinSum$ variable is equal to the sum of the $nextProteinValue_c$ for each $c \in C$
- A sum constraint [14] [15] to states that the $NextFatSum$ variable is equal to the sum of the $nextFatValue_c$ for each $c \in C$
- An Element-GreaterOrEqual constraint [23] [24] to states that the $NextFatSum$ must be greater or equal to the $PreviousSum$ variable multiplied by A.
- An Element-GreaterOrEqual constraint [23] [24] to states that the $NextProteinSum$ must be greater or equal to the $PreviousSum$ variable multiplied by A.

4.8 The objective function

We have a $TotalCost$ variable for the consanguinity constraint, for the quality constraints, for the security constraint and every conformity constraint. Those must be minimized. We have weight parameters associated with each constraints to give importance to each depending on the needs of the farmer. We need to consider all the constraints at once in order to weight them; having a minimizing function for each of the $TotalCost$ variables would not allow that. We have a $FinalTotalCost$ variable that represents the weighted sum of all the $TotalCost$ variables, stated with the WeightedSum constraint [25]. We have a single objective function minimizing the $FinalTotalCost$ variable.

Chapter 5

Parameterization of the model

The final objective of this project is to be used by final users. To achieve that goal, the application must fits the needs of the farmers willing to use it. It is thus interesting to allow the users to parameterize the model as they want.

5.1 Parameterization of the constraints

We implemented as many constraints as possible in order to make them available for the user to use. Using all the constraints at once could lead to a dead end, because the model would be too constrained and no possibilities are left, which means that no solution is found and we still don't know which bull to choose for which cow. The next sections present solutions to parameterize the constraints.

5.1.1 Weighted constraints

One solution to that problem is to weight each constraint according to the importance we want to give it. We have:

- N the number of constraints
- w_i the weight associated with the constraint $i \in N$
- $\forall i \in N, w_i \in [0; 20] \cap w_i \in \mathbb{N}$

In other words, each of the weight must be an integer between 0 and 20. When the weight equals zero, the associated constraints is discarded. The higher the weight, we more importance the constraint have in the model.

Implementation

The weights have as an objective to accord importance to or on the contrary discard some constraints. To effectively reach the minimization objective while taking the weights into account, we need to consider every variable to minimize at once. We have a weighted sum of all the "TotalCost" variables to be minimized in a single objective function. See section 4.8 for more information.

5.1.2 Hard and soft constraints

Another solution is to use hard and soft constraints. As explained in the state of the art, section 2.1, a hard constraint is a constraint that cannot be violated in any way, while a soft constraint relaxes the model by allowing "violations" of the constraint.

In our case, hard constraints will discard from the domain of a particular cow every bull that do not perfectly respect the constraint. For example, in the case of the fat and protein constraints, table 4.3 shows us that the bull is to be rejected if its value is negative and the value of the cow is below the average value of the herd. This might be a too harsh treatment, because the bull could be very interesting considering other constraints.

Using soft constraints is a solution. Instead of discarding the bulls, we give them a penalty. The more the constraint is violated, the higher is the penalty. The objective of the soft constraint is to minimize the penalties. That way, a bull having a really good score for all other constraints but being slightly under the goal value for one particular constraint can still be chosen for some cows (if it is the best match of course) instead of being discarded for the violation of one constraint.

Implementation

The implementation is quite simple. When a constrain is hard and we find a pair whose cost is too bad (the costs represented by a $\perp$ sign in the tables in chapter 4), we simply remove the bull from the domain of the variable of the cow. The minimization still holds, because we can still attribute penalties to some bulls whose value is not good but not bad enough to be discarded. On the other hand when the constraint is soft, we don't discard any bull and simply to the minimization with all the costs.

5.1.3 The Consanguinity Strategy

As stated in section 4.1.3, there exists different strategies to compute the consanguinity rate. The strategies have a different impact on the consanguinity constraint, impacting the decisions during the search.

The three strategies are "Average", "Worst" and "Threshold". The latter requires an extra parameter, the threshold determining the worst consanguinity rate acceptable.

5.2 Selection of the cows to inseminate

Usually, the farmer doesn't inseminate all his cows at the same time. Pregnancy and birth giving require some supervision and it wouldn't be manageable to have to handle all the birth over a single month. So the users must be able to choose the set of cows to use in the application according to their needs.

To do so, they must provide the application with a file containing a list of cows. This file could contain only the cows to be selected, or the whole set of cows that compose the herd at that time. In the case of the latter, more criterion should be provided in order to select from the herd a subset of cows. The subsets can be selected based on the IDs when two border IDs are provided, selecting every cow whose ID is strictly between the two borders. Following the same semantic, the subset selection can be done based on the birth dates. To do so, we had to use a data structure keeping track of the birth dates, as we cannot query our data based on their attribute solely like a DBMS with SQL (see section 3.3). The last way to select a subset from the

herd is to ask for the heifers only or on the contrary the cows that have already calved before only.

The last kind of selection that would be useful for the farmer is a selection by hand, one by one, from the cows of the herd. This require a graphical user interface (GUI) to display the loaded cows along with checkboxes to do the selection. A GUI is not complicated to set up but takes some time. We chose to concentrate our work on the modelization of the problem with the idea to do the GUI if we had time, time that we spent on other things that we considered more important.

5.3 Final number of bulls

We have a certain number of cows to inseminate and a certain set of reproducing bulls from which to pick in order to choose the best fit for each cow. We want the set of bulls to be large enough so that the range of selection is diversified, which is an important criterion in optimization. If we don't put any restriction, we will find all the perfect fits. But we might have a solution where no bull is selected twice. In practice, this can be a problem due to the number of cows to inseminate.

Having the same bull to impregnate more than one cow is more interesting: it is easier to find the cows corresponding to the insemination dose, so the risk of errors is reduced and the overall operation takes less time. It can even cost less because some breeders give discounts when we buy a sufficient amount of doses. To force that into the model, we have a parameter that indicates the maximum number of different bulls we want to have in the solution. That parameter can take its value between 1 and the total number of reproducing bulls that are registered in the application. The solution may contain less bulls than expected by this parameter, bot not more. See section 4.1.4 for more information about the utilisation of this parameter.

5.4 In practice

In practice, the parametrization is done via a json file. The following code shows the structure of the file. Of course it is only a portion of the file, many constraints have been discarded as they follow the same format as those shown.

```
1  {
2    "birthsPath" : "data/Naissances_v2.txt",
3    "lostNamesPath" : "data/lostNames.txt",
4    "bullBreedingPath": "data/TaureauxElevage.txt",
5    "samplePath" : "data/sample.txt",
6    "conformationPath" : "data/Conformation.txt",
7    "chosenCowsPath" : "data/cows.txt",
8    "chosenBullsPath" : "data/bulls.txt",
9    "nbMaxBulls" : 4,
10   "constraints" : [
11     {
12       "name" : "consanguinityworst",
13       "weight" : 1,
14       "isHard" : false,
15       "threshold" : -1
16     },
17     {
18       "name" : "consanguinitythreshold",
```

```
19          "weight" : 1,
20          "isHard" : false,
21          "threshold" : -1
22        },
23        {
24          "name" : "consanguinityavg",
25          "weight" : 1,
26          "isHard" : false,
27          "threshold" : -1
28        },
29        {
30          "name" : "milkimprovement",
31          "weight" : 1,
32          "isHard" : false,
33          "threshold" : 1000
34        }],
35       "selections" : [
36        {
37          "name" : "id",
38          "activated" : false,
39          "from" : "345 345",
40          "to" : "450 450"
41        },
42        {
43          "name" : "birth",
44          "activated" : false,
45          "from" : "25/09/94",
46          "to" : "25/09/02"
47        },
48        {
49          "name" : "calving",
50          "activated" : false,
51          "from" : "0",
52          "to" : "<"
53        }]
54    }
55
```

The five first parameters are the path towards the different files we need to start the program. See chapter 3 for more information. The next two file paths are used for the selection of the bulls and the cows to be matched together. The chosen bull file is used on its own whilst the use of the cow file depends on the parametrization on the "selection" tab.

The "selection" tab contains data about the three type of selection we can us to determine the cows that we want to inseminate among the cows given in the file. each possible selection type has a name, a Boolean value indicating is we want to activate it or not and two values "from" and "to". In the case of the "ID" selection and the "birthdate" selection, we provide the border values to select cows whose value is strictly between them. The default values display the format we expect. In the case of the "calving" selection, the "from" attribute require a number and the "to" an inequality sign. Here, the inequality is not strict. The default value selects the cows having 0 or less calving, also known as the heifers. If no selection is activated, we use all the cows in the file. If more than one selection is activated, we use only the first one.

NbMaxBulls is the maximum number of different bulls we want in the solution. Then, we have the list of constraints. They are all in the same format: the name to identify it, a weight used to give it importance or not (a weight of zero means that the constraint is deactivated), a Boolean value to determine if the constraint should be hard ("true") or soft ("false") when applicable and a threshold, needed for some of the constraints. The consanguinity constraint exists in three "formats", depending on the strategy we want to apply. Only one of them can be activate at a time, so the other two must have a weight of zero. The threshold strategy uses the "threshold" attribute.

We use a library called Json4s to parse the file and extract the parameters [26]. The parameters are parsed in Object specific classes to make them easy to use in the code.

Chapter 6

Validation and tests

During the implementation of the model, we made sure to be on the right path by checking that the constraints worked. In this chapter, we detail the tests and checks we made and how we are satisfied with what we observed.

6.1 Test the constraints individually

We started by checking that each constraint worked as expected on its own. To do so, we created fictive cows and bulls with made-up values in order to know in advance the outcome that the constraint should provide. With those tests, we were certain that the constraints worked as expected with fake value, which we thought was mandatory before any test with real data.

6.1.1 Consanguinity constraint

For the consanguinity constraint, we created manually the genealogy of a few animals. We created animals not related at all and others with consanguinity on purpose. With those data, we checked that the consanguinity cost is indeed the sum of the observed consanguinities if it is expected by the strategy, that the different strategies worked as expected and that the solver chooses indeed the bull with the best rate.

6.1.2 Security constraint, quality constraints and conformation constraints

For the security, quality and conformation constraints, we gave to some of our fictive animals extreme data and to others very average data in order to observe the behaviour of the constraints. We checked that the soft constraints chooses indeed the best bull for each cow and that the hard constraints , in addition, do discard the bulls when the combinations require it. The way those constraints work is pretty similar so we did not encounter any problem from one to another.

6.2 Validation of the consanguinity constraint - Scalability tests

The consanguinity constraint is the main constraint and the most important one. the scalability tests aim at proving that the model is scalabled to the data, that the model is logical and coherent in its results and the time it takes according to the data it is provided. By variating the size of the set of cows to inseminate, the set of bulls to be matched to the cows and the maximal number of bulls allowed in the solution, we were able to obtain graphs that show the results and hopefully match our expectations.

We have been given six files containing lists of bulls to use for the selection. Those files are named from "T-C1" to "T-C6". The first five contains a total of 42 bulls and the last one contains 9 bulls that were breeded in the herd. Depending on the files that are used, the results should be different.

6.2.1 First validation - consanguinity constraint with relatively easy problems

The first validation is done with the 42 bulls of the first five files of bulls. We have C the set of cows, B the set of bulls and M the maximal number of bulls in the solution. We made 27 combinations with $|C| \in \{20, 50, 100\}$, $|B| \in \{15, 20, 40\}$ and $M = \{2, 5, 10\}$. The set of animals are composed by a random selection among the available data.

Obviously, the selection of the animals have an impact on the consanguinity cost. In order to balance that, we computed our tests three times with each configuration, resampling the sets at each test. We obtained this 81 measures of the time (in milliseconds) needed by the program to set up the model and find a solution, and of the consanguinity cost of the herd normalized by the number of cows in the test. The costs are percentages multipiled by 1000 in order not to deal with float values. We used the "Average" strategy, the most common one.

We expect the problem to be more difficult when the number of cows increases and when the maximal number of bulls in the solution decreases. We put the results in graphs in order to visualize the scalability. We have two graphs by maximal number of bulls, the parameter M, one showing the normalized costs and the other the time.

Figure 6.1 shows a clear tendency in the time needed by the program to find the optimal solutions. The more cows, the more variables and thus the more time it needs. The more bulls in the selection set, the more time it takes to try and find the best among the possibilities.

By comparing the figures 6.1, 6.3 and 6.5, we can see that the more bulls allowed in the final solution, the less time it takes.

Concerning the consanguinity costs, it is hard to observe a tendency, as no pattern seems to stand out. The sampling of the data may have an important impact on the results. The second validation is there to confirm that.

6.2.2 Second validation - consanguinity constraint with hard problem

For the second validation, the set from which we chose the bulls is the set given in file "T-C6". With this file, the consanguinity costs are expected to be much higher because the file contains mostly bulls that were breeded in the farm and thus that have many relatives with the cows in the herd. We have $|C| \in \{20, 50, 100\}$, $B = 9$ and $M = \{1, 2, 3\}$. The lesser the maximum number of bulls in the solution, the harder the problem. Again, we normalize the costs by the number of cows. The selection of the animals are random before each test, so in order to decrease the influence of the selection we performed each test five times.

The graph in figure 6.7 shows the obtained results. The results prove exactly what we have been expecting. The costs are much higher than those observed in the first validation. The worst costs were about 300 in figure 6.4 where it reaches 1500 here! Moreover, we can see that with only one bull allowed in the solution, the costs are usually higher than in the other situations.

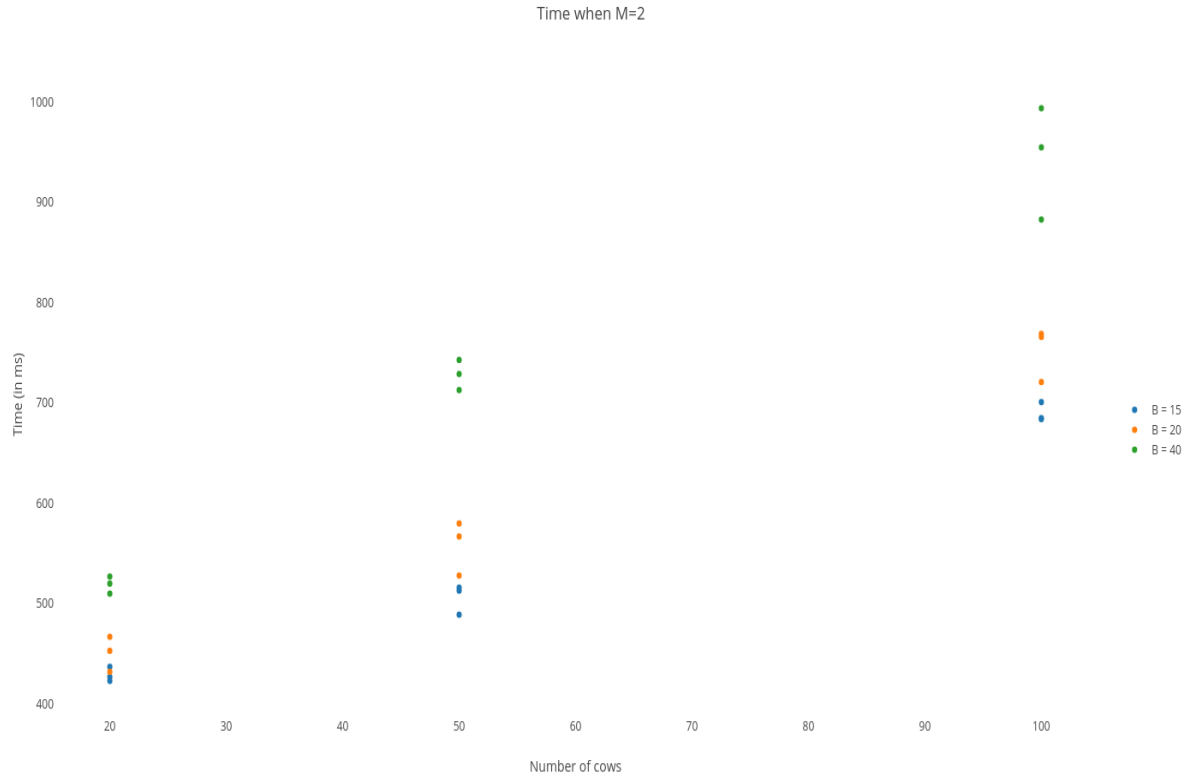


Figure 6.1: First validation - Time needed by the consanguinity constraint with M=2

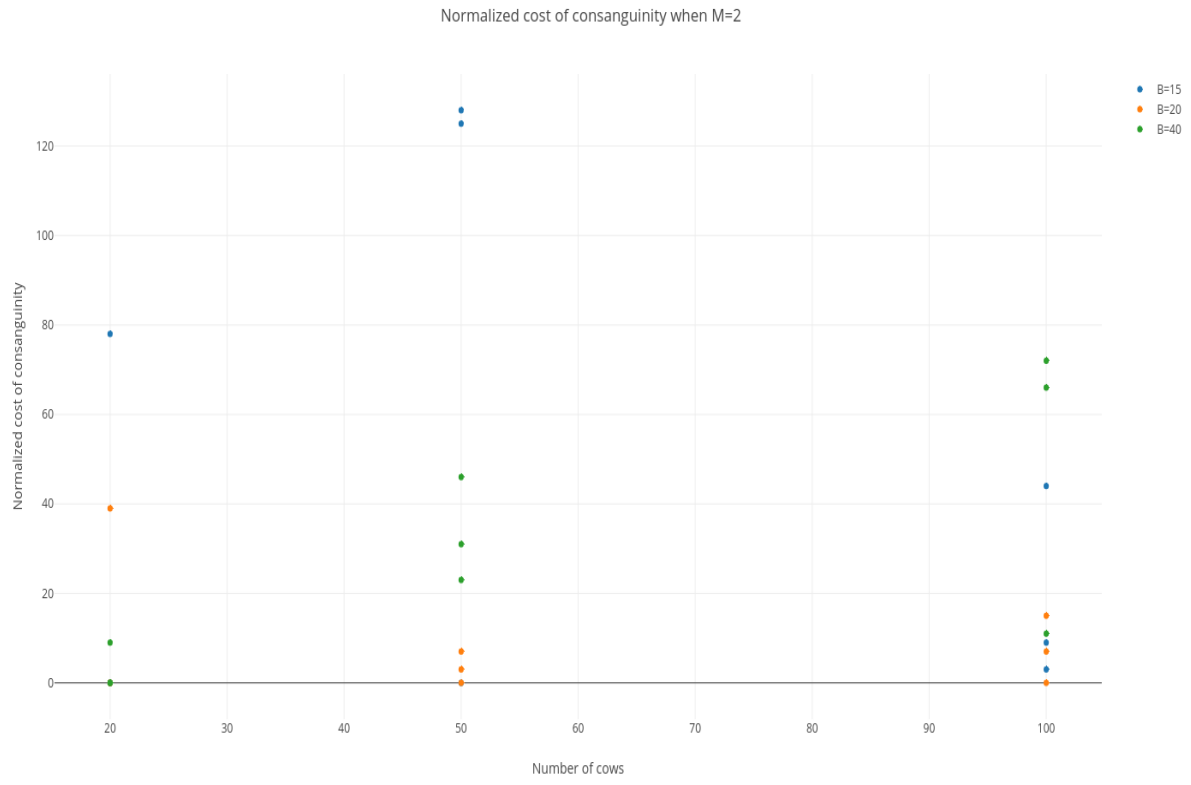


Figure 6.2: First validation - Normalized consanguinity cost when M=2

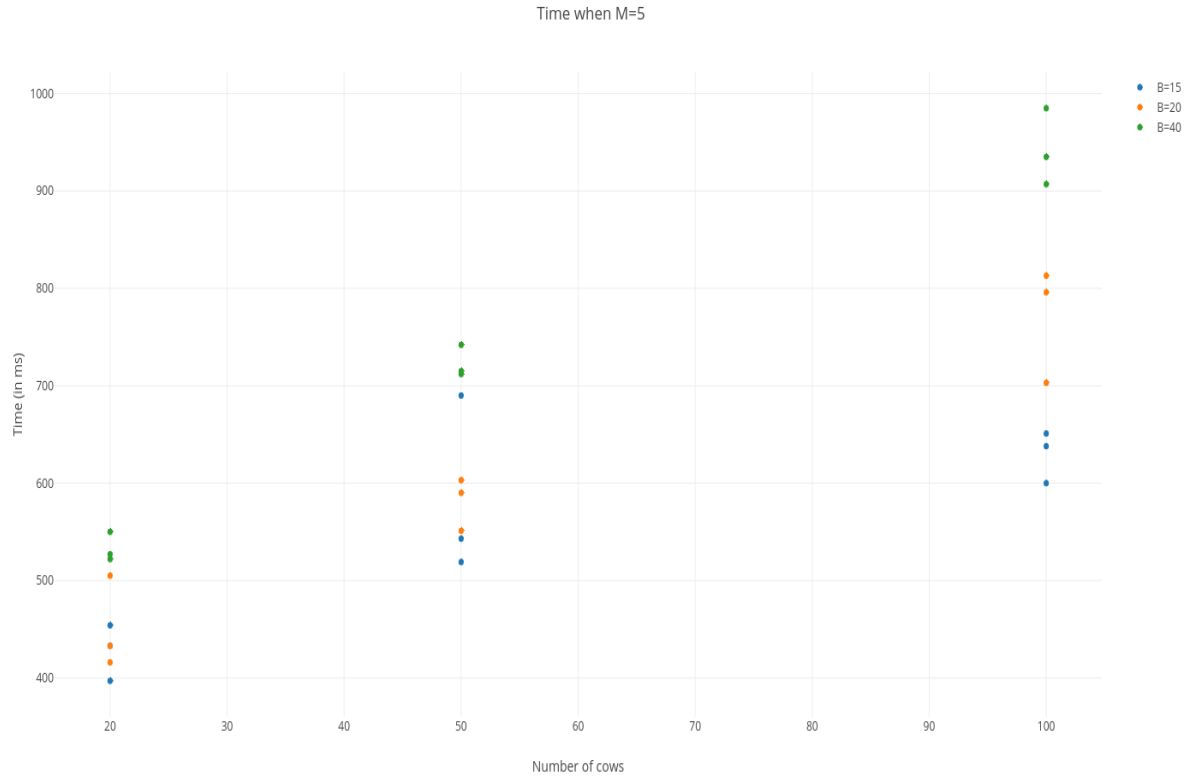


Figure 6.3: First validation - Time needed by the consanguinity constraint with M=5

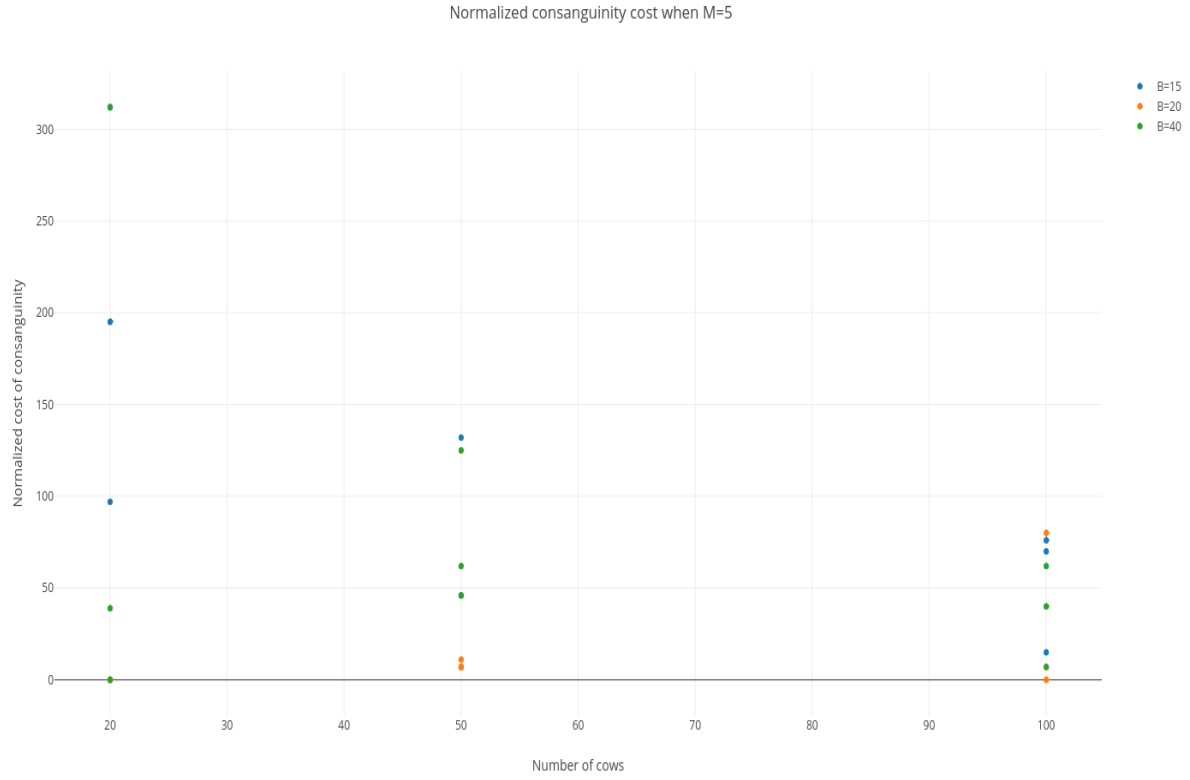


Figure 6.4: First validation - Normalized consanguinity cost when M=5

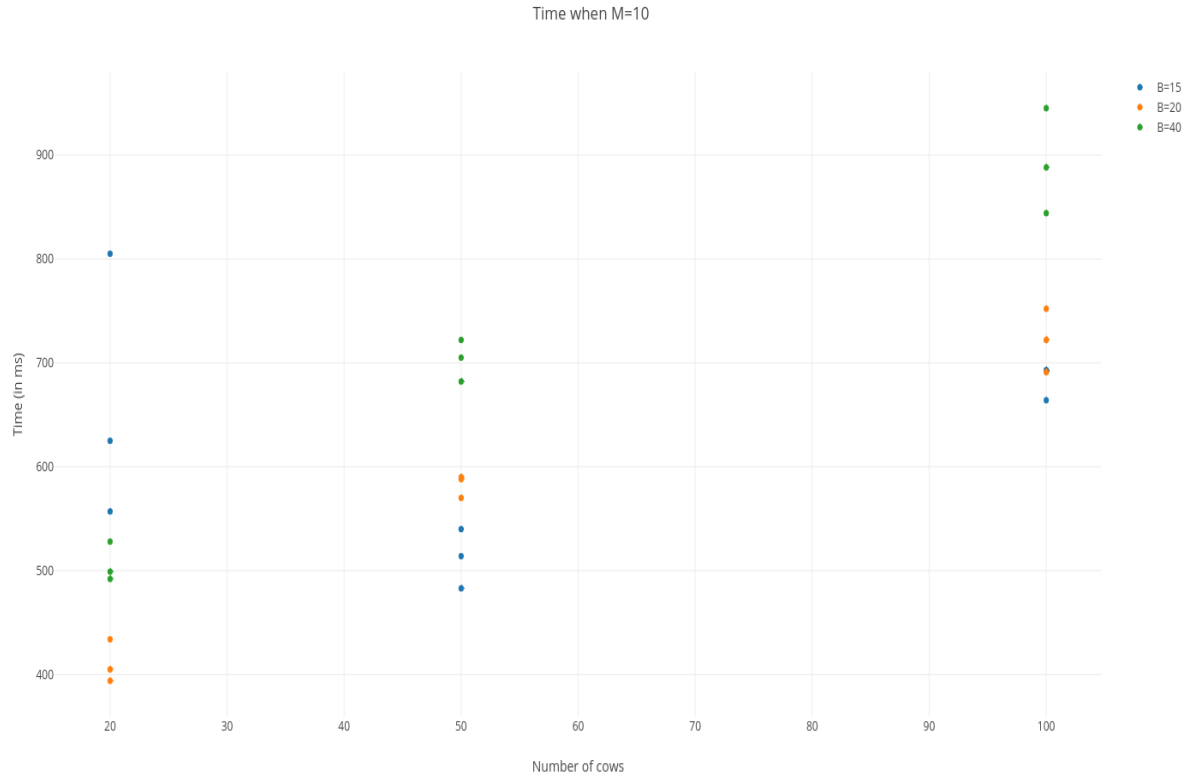


Figure 6.5: First validation - Time needed by the consanguinity constraint with M=10

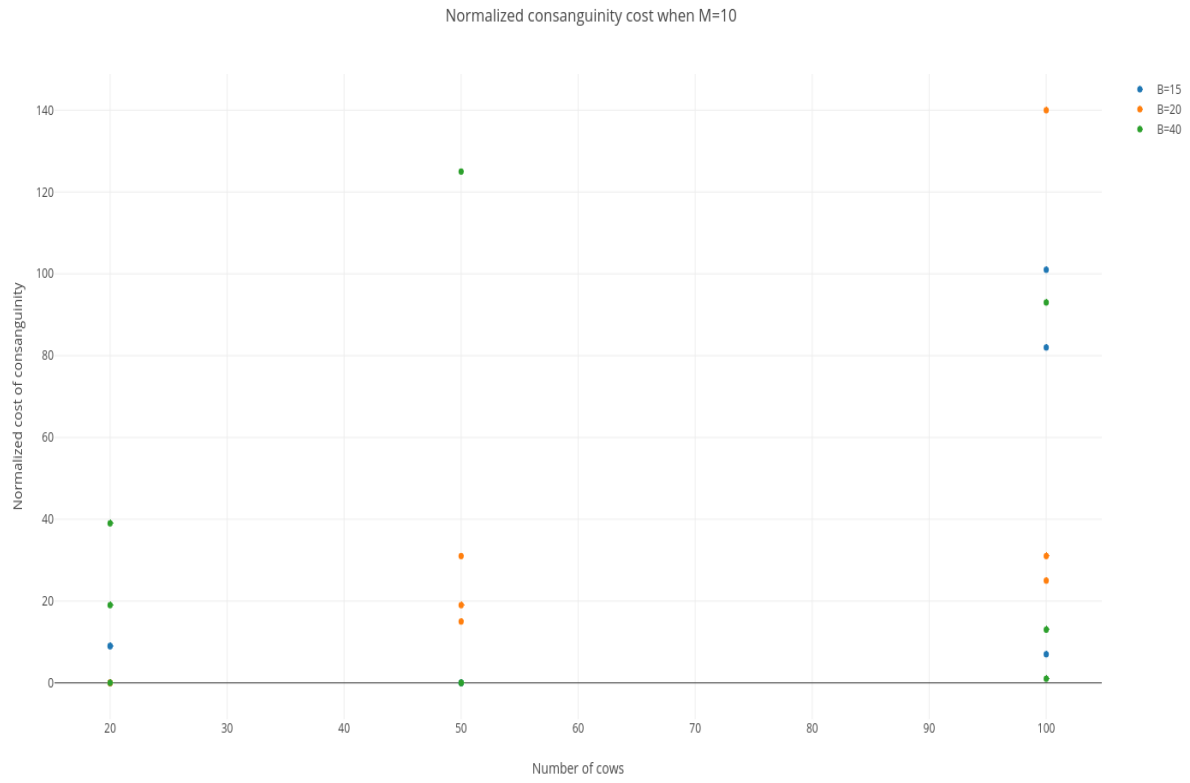


Figure 6.6: First validation - Normalized consanguinity cost when M=10

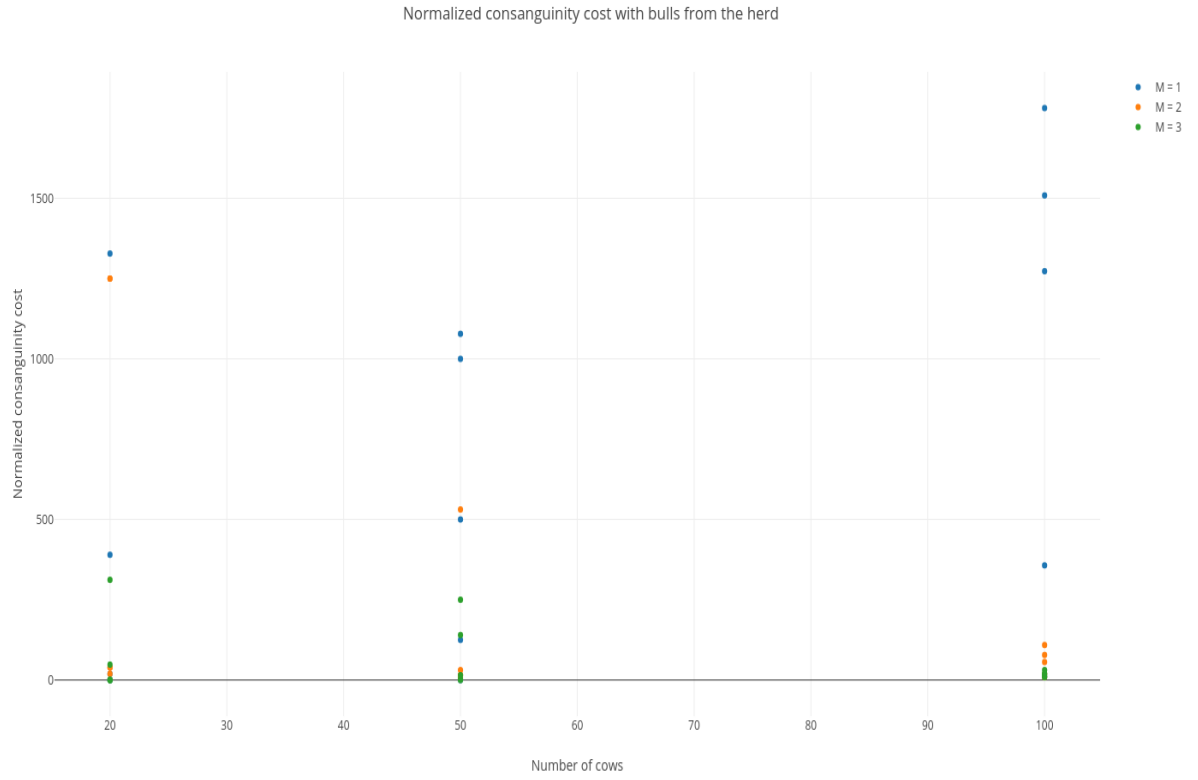


Figure 6.7: Second validation - Normalized costs of the consanguinity constraint

With three bulls in the solution, the consanguinity costs can stay under 350 in the worst case when we have only a few cows, which is really good considering the data.

6.3 General validation

General validations take many constraints at the time and even weight them to give more importance to some and less to others.

6.3.1 First general validation - Consanguinity constraint; Security, conformation and quality constraints in soft mode

The first general validation is similar to the first validation of the consanguinity constraint in the sense that we use the same parameters for the selection process and we have three occurrences of each test in order to balance the effects of the selection process. We have $|C| \in \{20, 50, 100\}$,, $|B| \in \{15, 20, 40\}$ and $M \in \{2, 5, 10\}$. In addition, we use more constraints: the consanguinity constraint weighted by one, the conformation constraints weighted by five, the quality constraints weighted by five and the security constraint weighted by five. We put all of those constraints in soft mode, so we should always find a solution, due to the absence of rejection of any value in the domain of the variables. However, we encountered time outs, when we judged the application to take too much time to be used friendly. Those appear in the graph under the name of "errors" coloured in red.

We forced the program to select only one solution so that we could compare the results in graphs. Indeed, the different configurations and animal selections can lead to a different number of solutions. The runs that have more solutions inevitably take more time. To avoid that, we

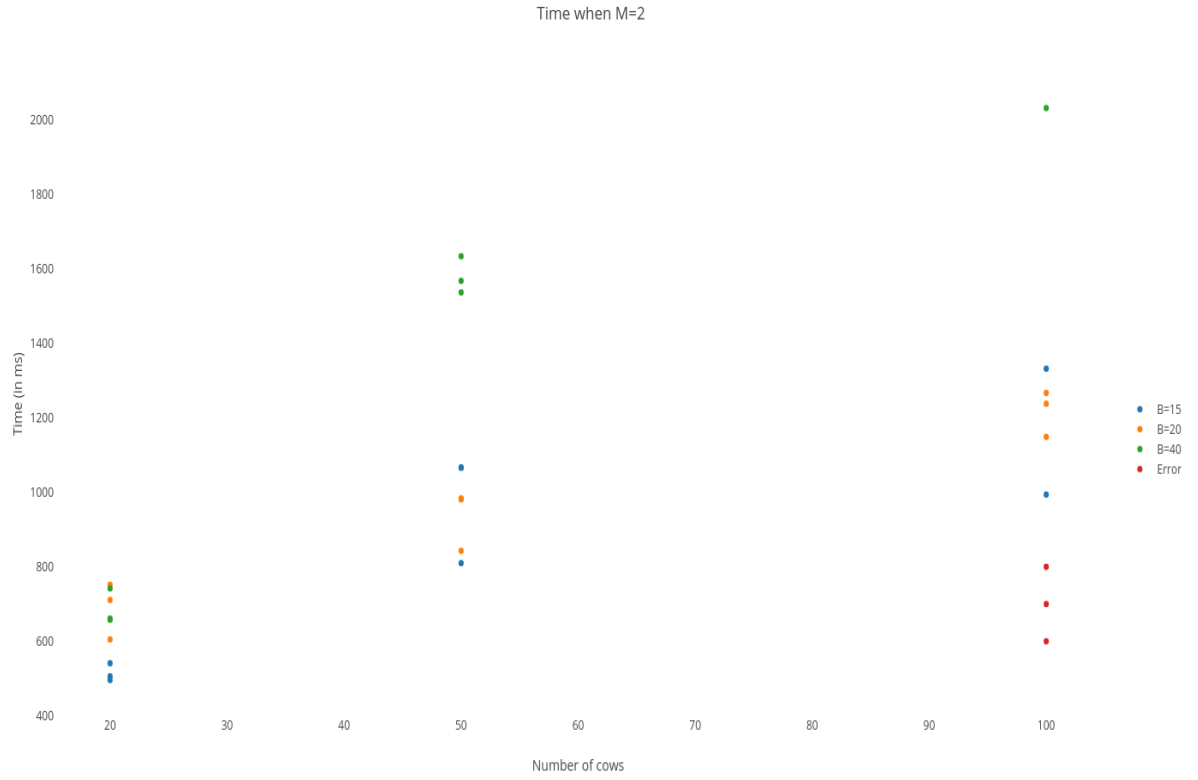


Figure 6.8: First general validation - Time needed to find a solution when M=2

limited each run to one solution.

Again, we can observe a pattern in the time required by the process, the more variable, the more time needed. Even, in the case of the number of bulls allowed in the solution limited to two, which is pretty harsh, we had to force a time out. So the maximum value for the time in figure 6.8 is not even to its true maximum.

When we look at the costs and time graphs that go together, having the same value for M, it stands out that having less variables is always more interesting in term of time as well as in term of costs. Globally, the costs are better when the model have less variables to assign. This makes sense, we know that the less variables there are, the easier the problem and the more bulls allowed in the solution, also the easier the problem.

6.3.2 Second general validation - Consanguinity constraint; Security, conformation and quality constraints in hard mode

The second general validation is the same as the first general validation, except that all the constraints are set to hard mode. It means that we may encounter situations where no solution is found. Those appear in red in the following graphs, under the name "errors".

We expect the search phase to be faster with hard constraints, as the domain of the variables decrease in size with every violated constraint. It is indeed observable in the graphs. Figures 6.14 and 6.18 reach a maximum of 1600 ms. Graphs of the first general validation in figures 6.8 and 6.12 exceed the 2000 ms and the latter should even reach more if we did not provoke time outs. Surprisingly, when M equals 5 like in figures 6.10 and 6.16 the maximum time is 1400 ms,

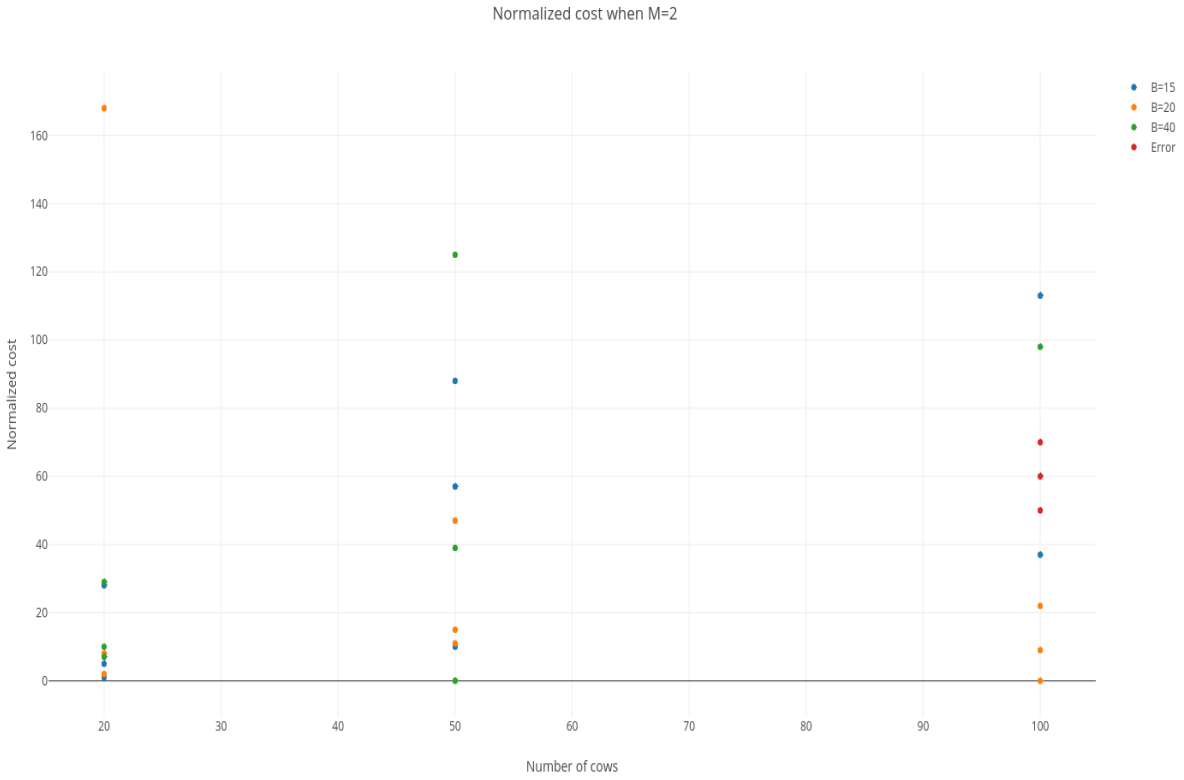


Figure 6.9: First general validation - General cost to find a solution when M=2

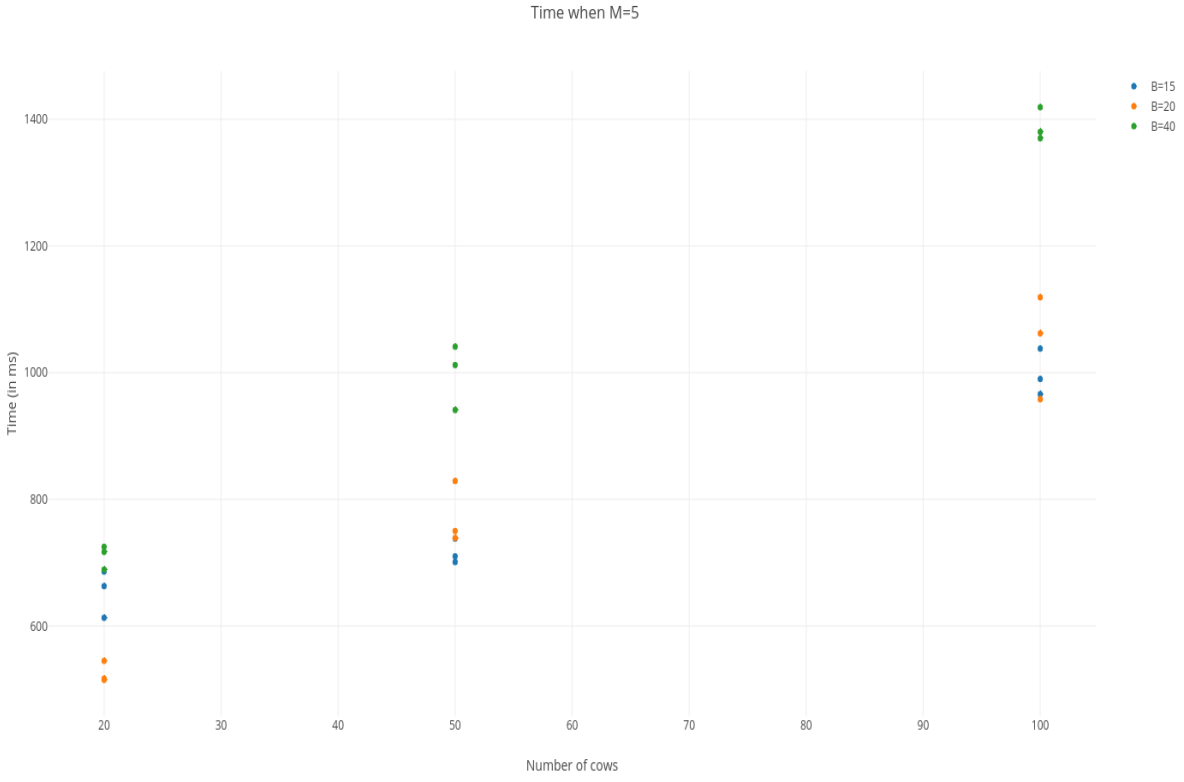


Figure 6.10: First general validation - Time needed to find a solution when M=5

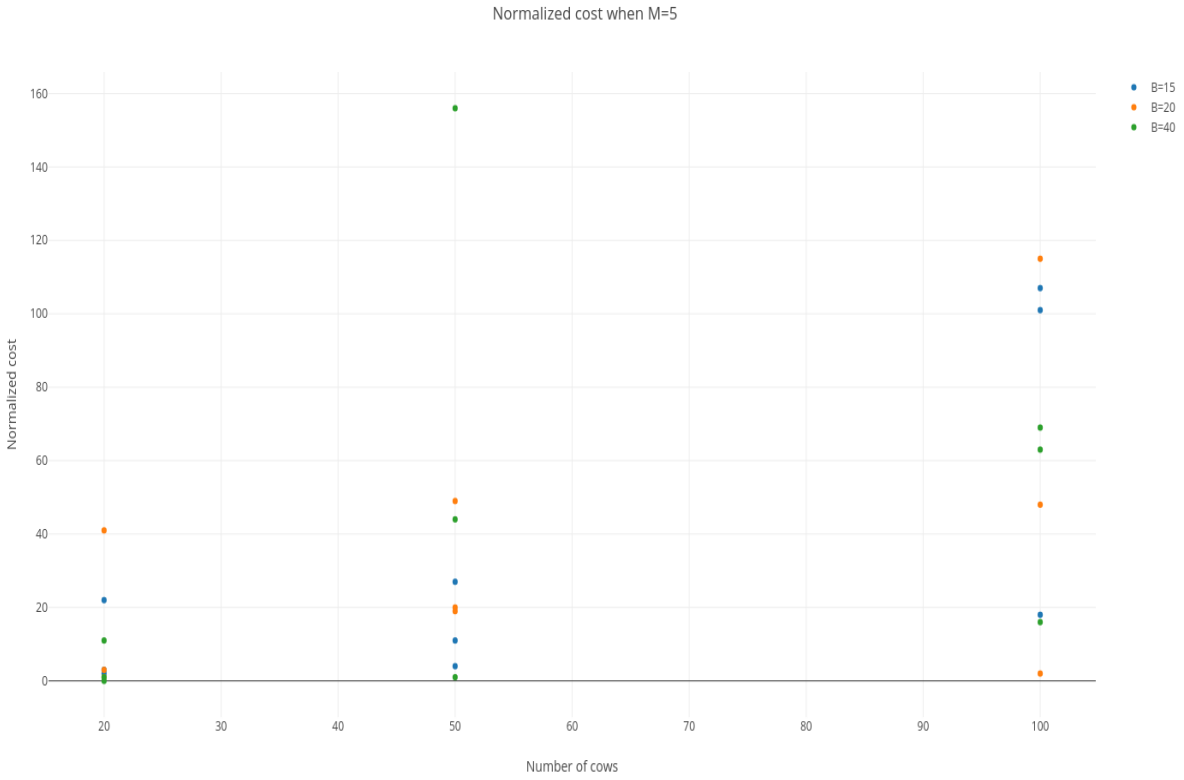


Figure 6.11: First general validation - General cost to find a solution when M=5

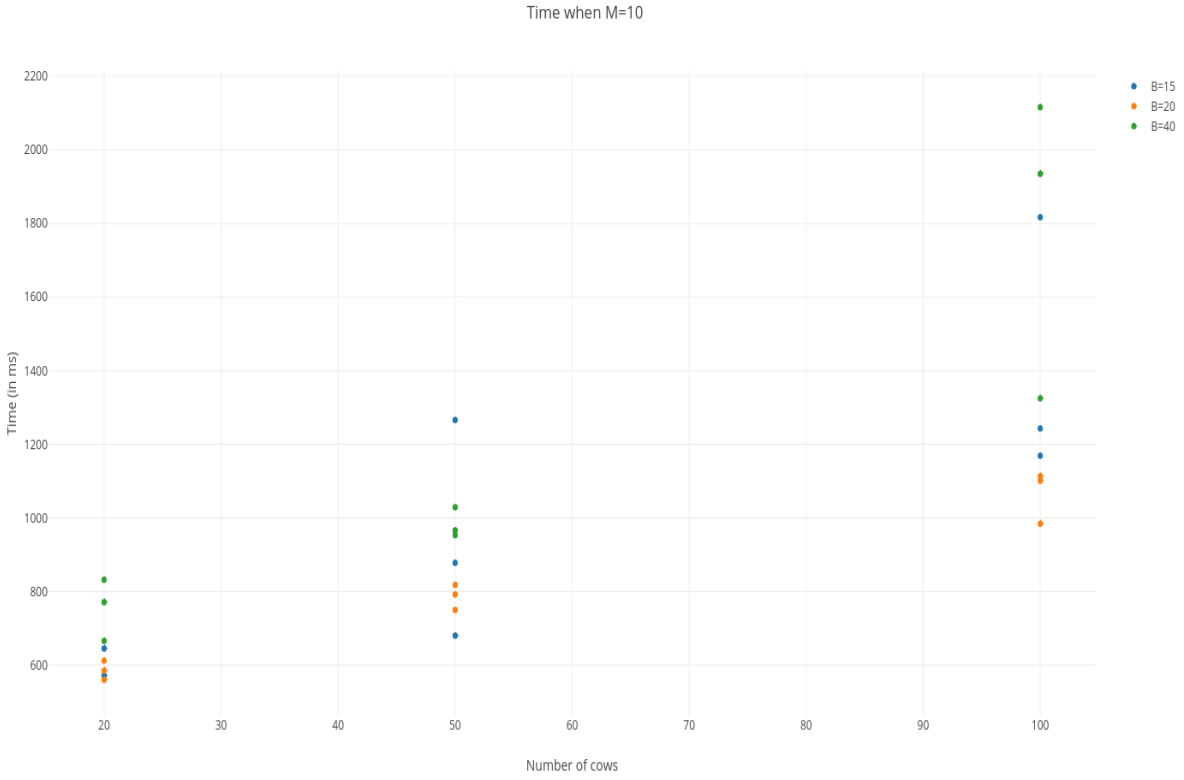


Figure 6.12: First general validation - Time needed to find a solution when M=10

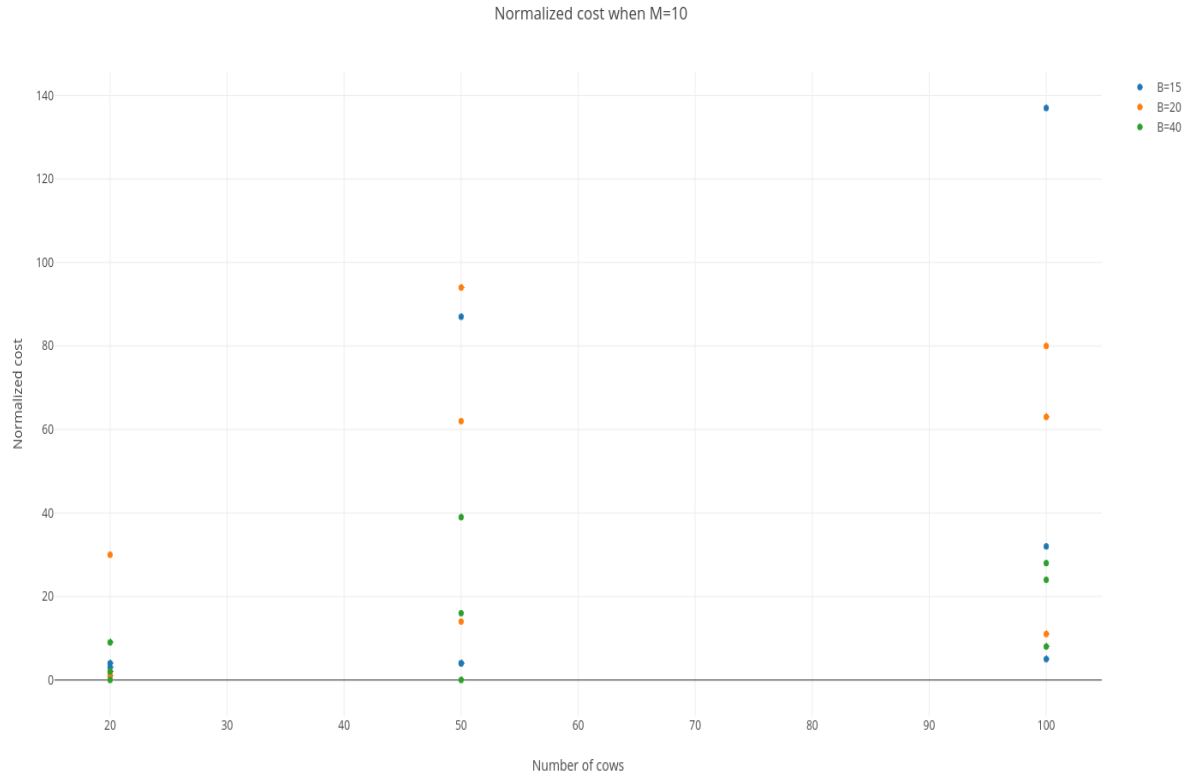


Figure 6.13: First general validation - General cost to find a solution when M=10

either with hard or soft constraints.

6.3.3 Third general validation - Consanguinity constraint; Security, conformation and quality constraints in hard mode; Global improvement constraints

The third and last general validation takes the global improvement constraints into account. Those do not have a hard or soft mode, they are soft by nature. They should add computational time, due tot the number of intermediate variables that are needed to model them. Moreover those constraints are global, they impact all selected bulls at once, it require thus to find a combination of values, not just a single value, to satisfy the constraint.

6.4 Conclusion

The different tests and graphs confirm that the model behaves as expected. The problem is indeed harder when more constraints are involved in the model. It is also harder when the number of cows to inseminate, thus the number of variables increase, as well as when the maximum number of bulls allowed in the solution is smaller. Hard constraints can lead to a too constrained problem having no solution while soft constraints might lead a high constrained problem to taking too much time to provide a solution. The solver proposes multiple solutions when there exists more than one.

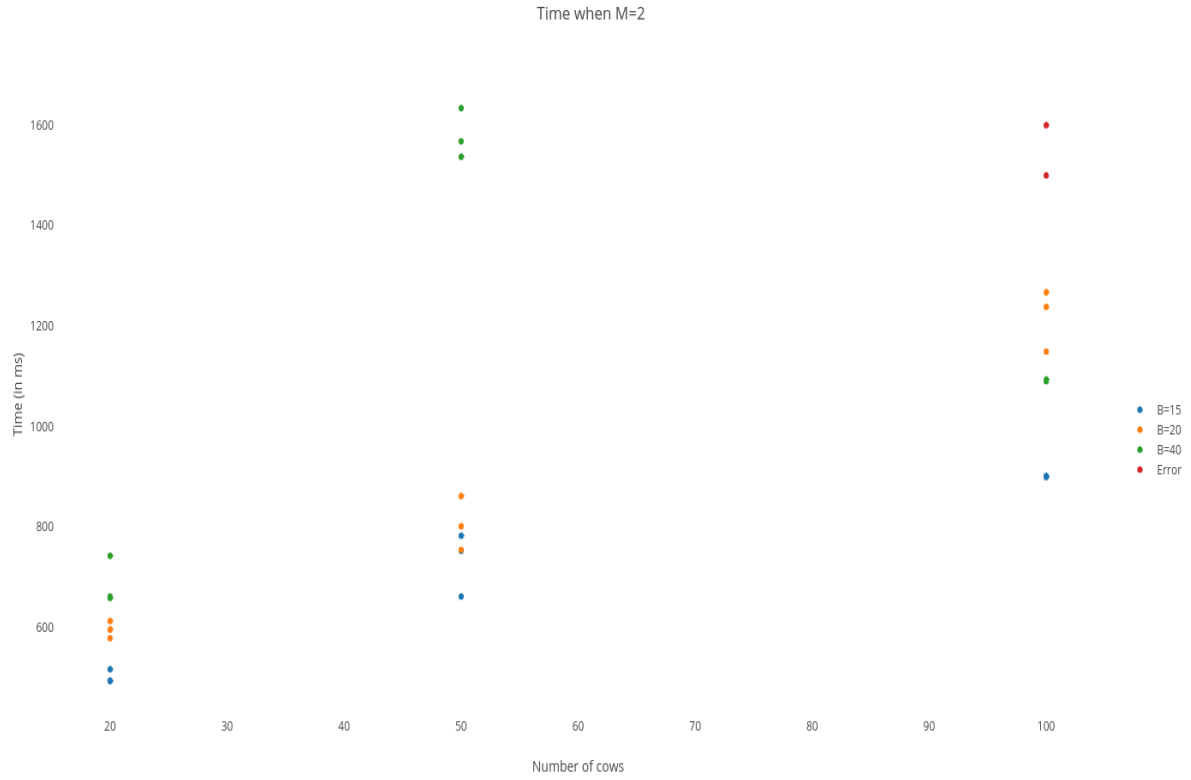


Figure 6.14: Second general validation - Time needed to find a solution when M=2

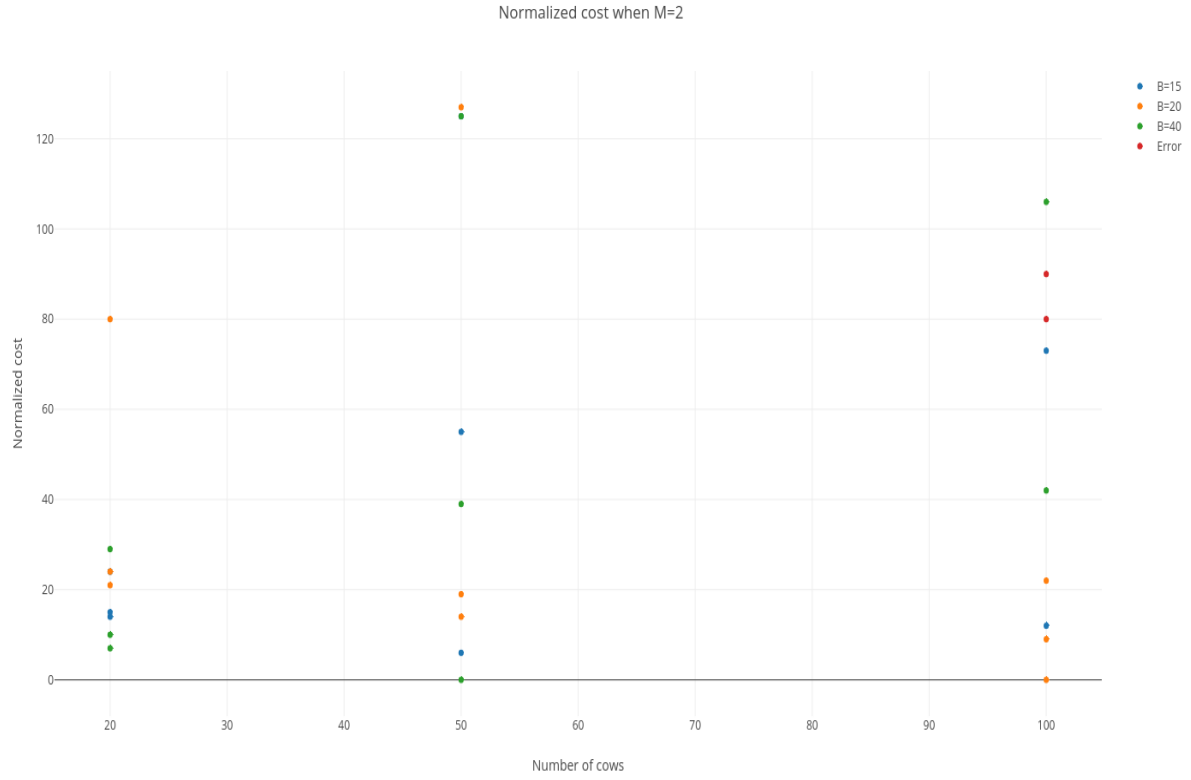


Figure 6.15: Second general validation - General cost to find a solution when M=2

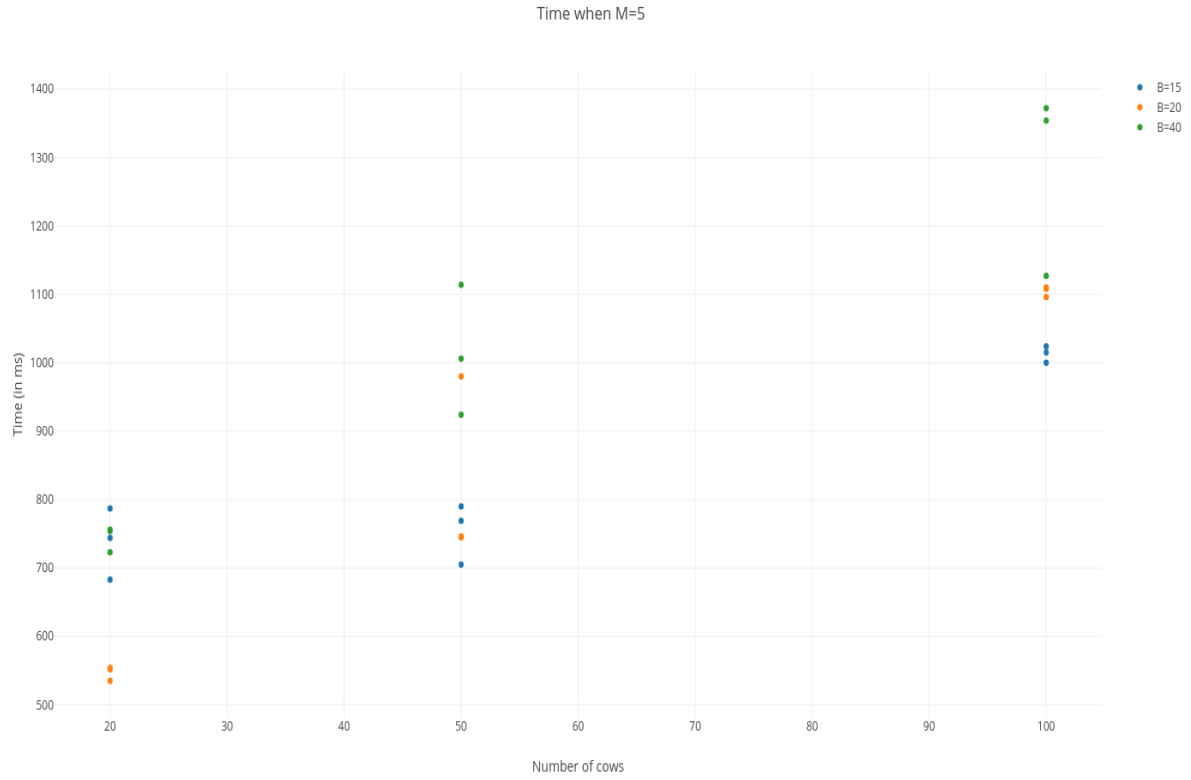


Figure 6.16: Second general validation - Time needed to find a solution when M=5

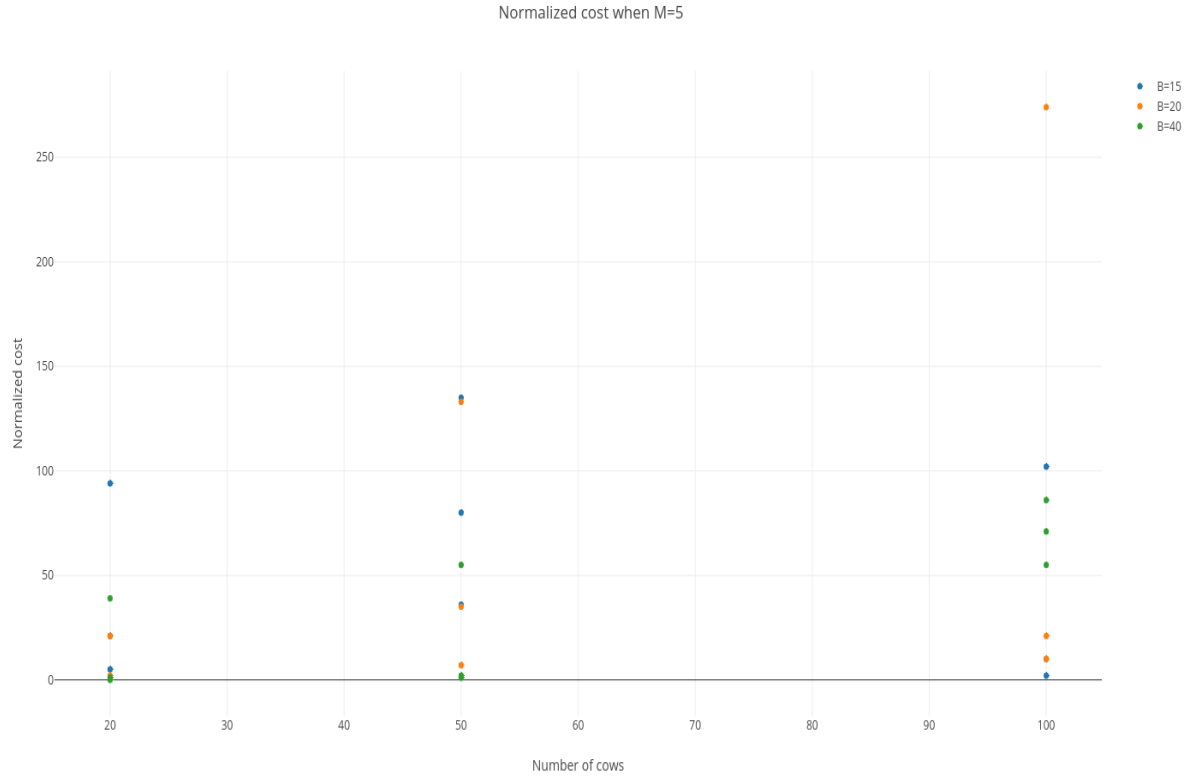


Figure 6.17: Second general validation - General cost to find a solution when M=5

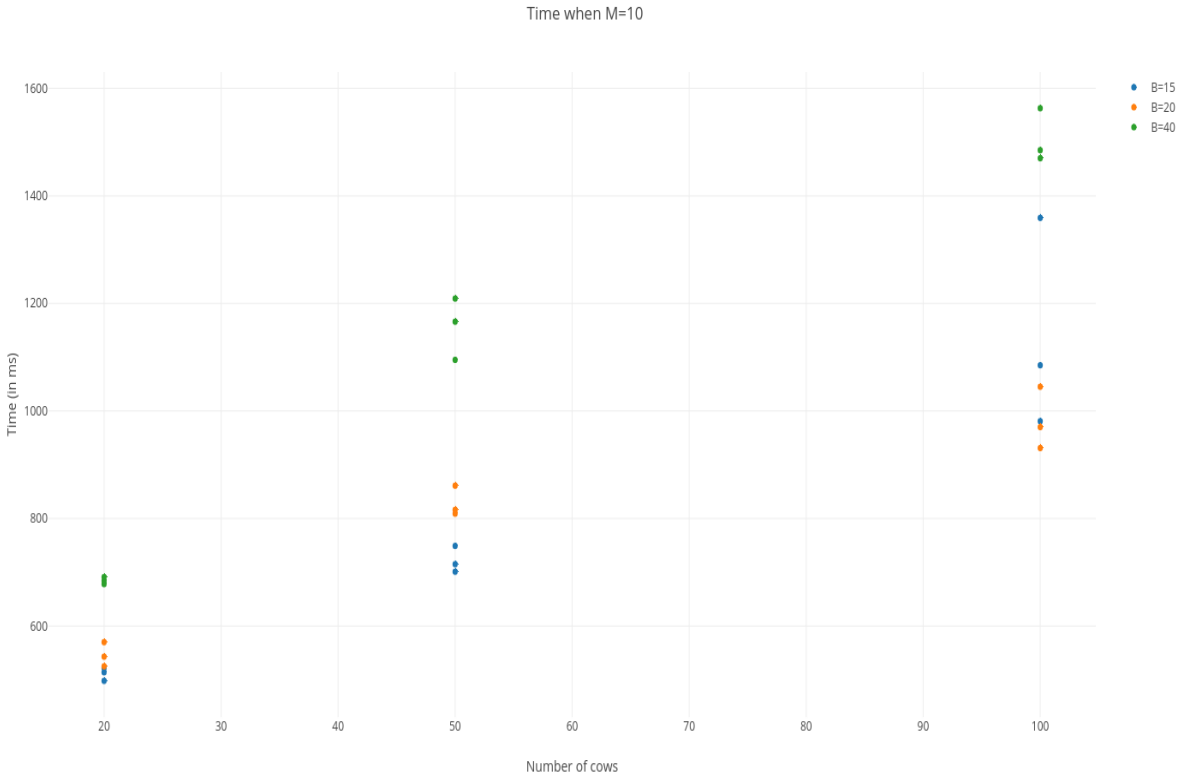


Figure 6.18: Second general validation - Time needed to find a solution when M=10

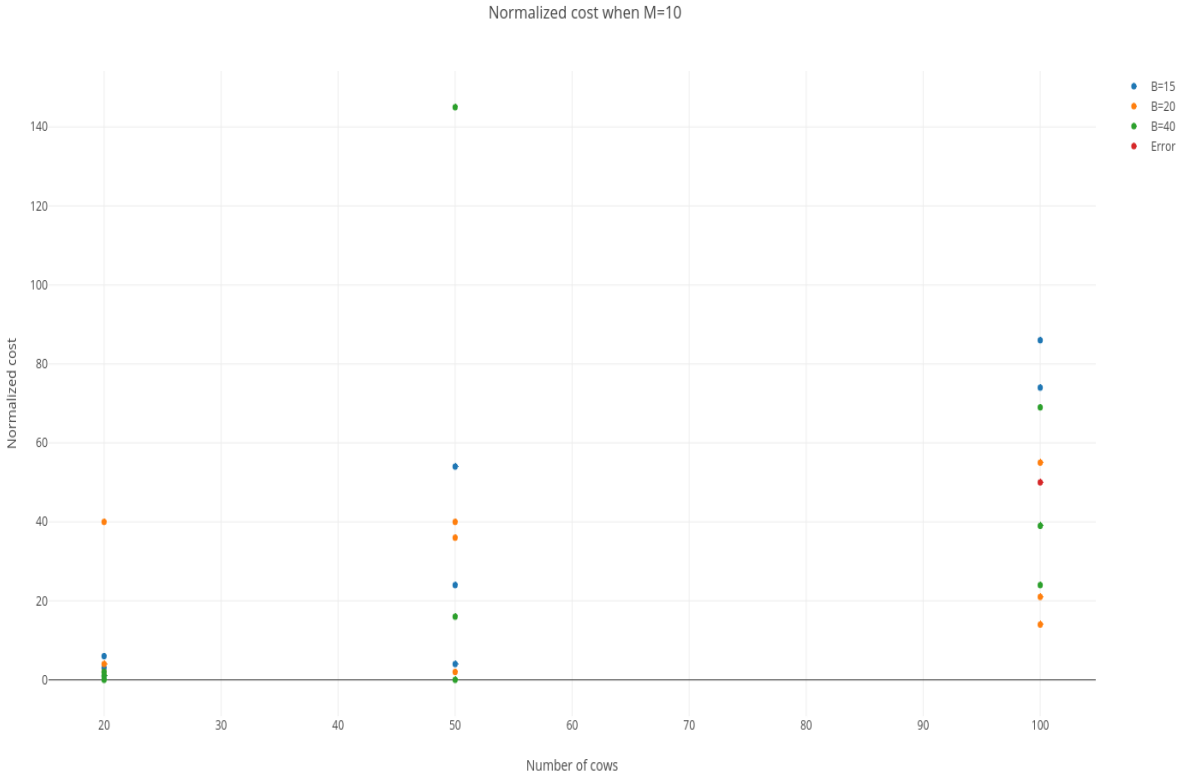


Figure 6.19: Second general validation - General cost to find a solution when M=10

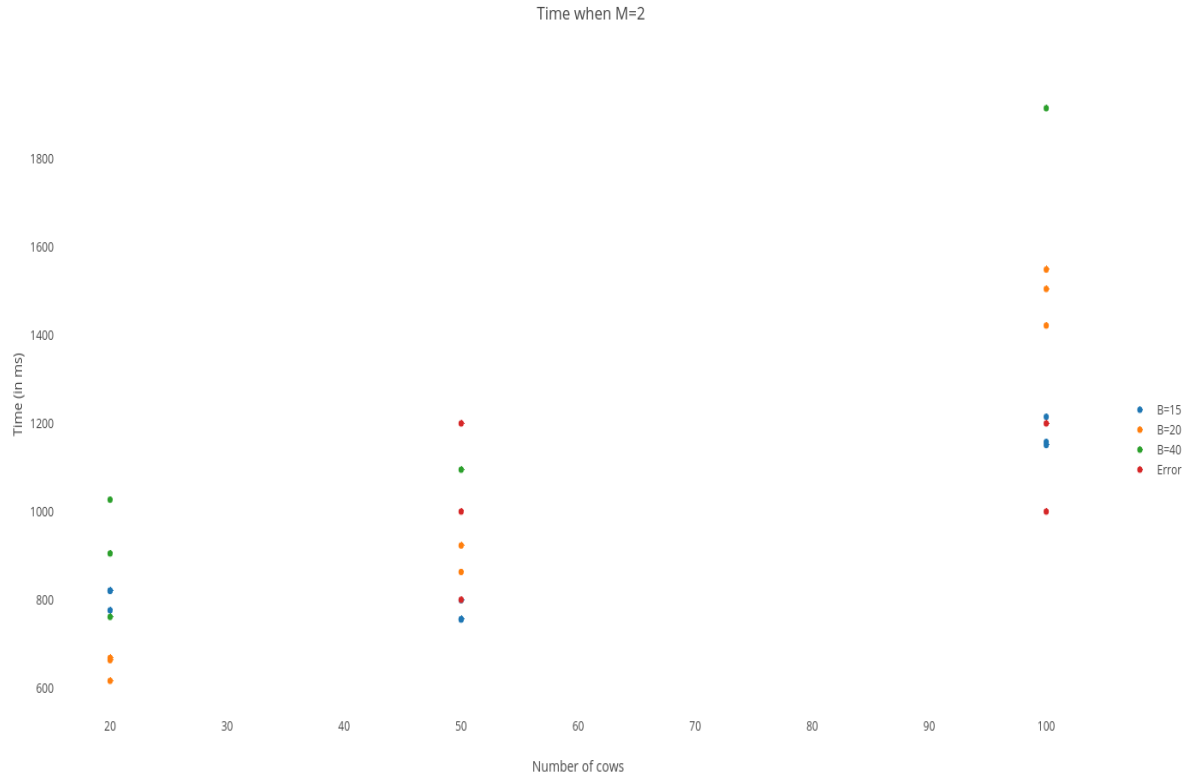


Figure 6.20: Third general validation - Time needed to find a solution when M=2

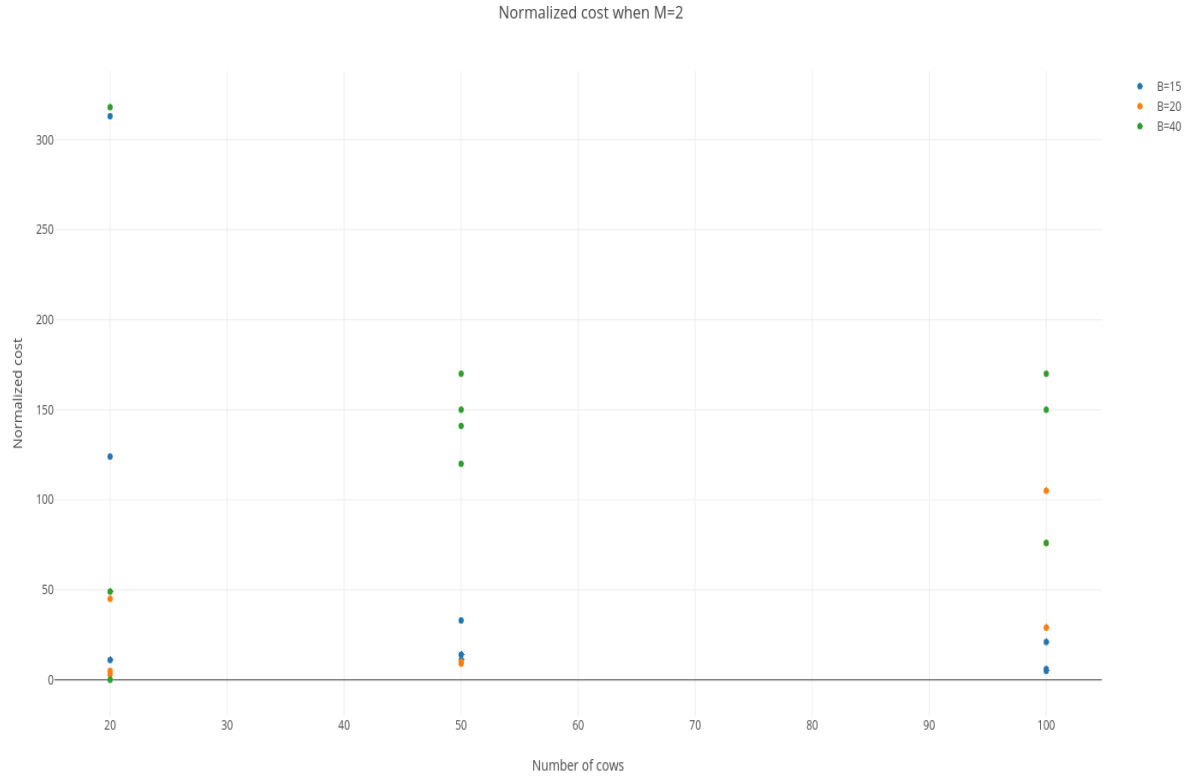


Figure 6.21: Third general validation - General cost to find a solution when M=2

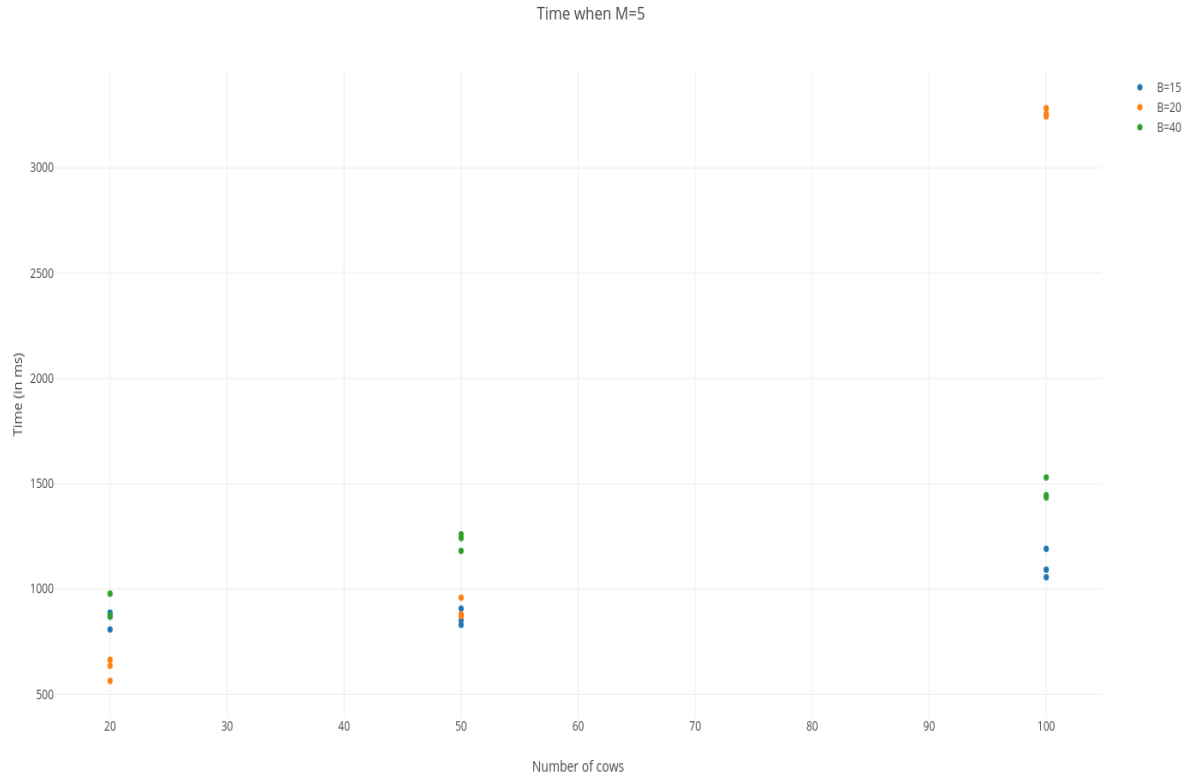


Figure 6.22: Third general validation - Time needed to find a solution when M=5

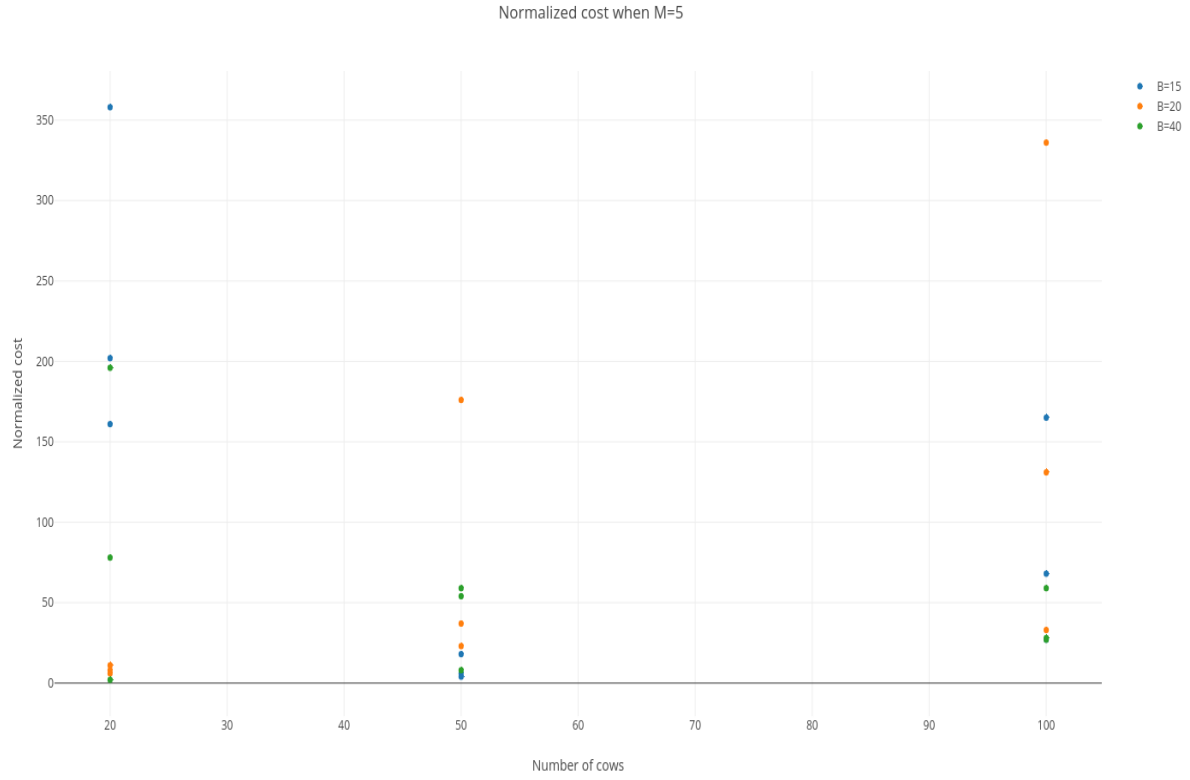


Figure 6.23: Third general validation - General cost to find a solution when M=5

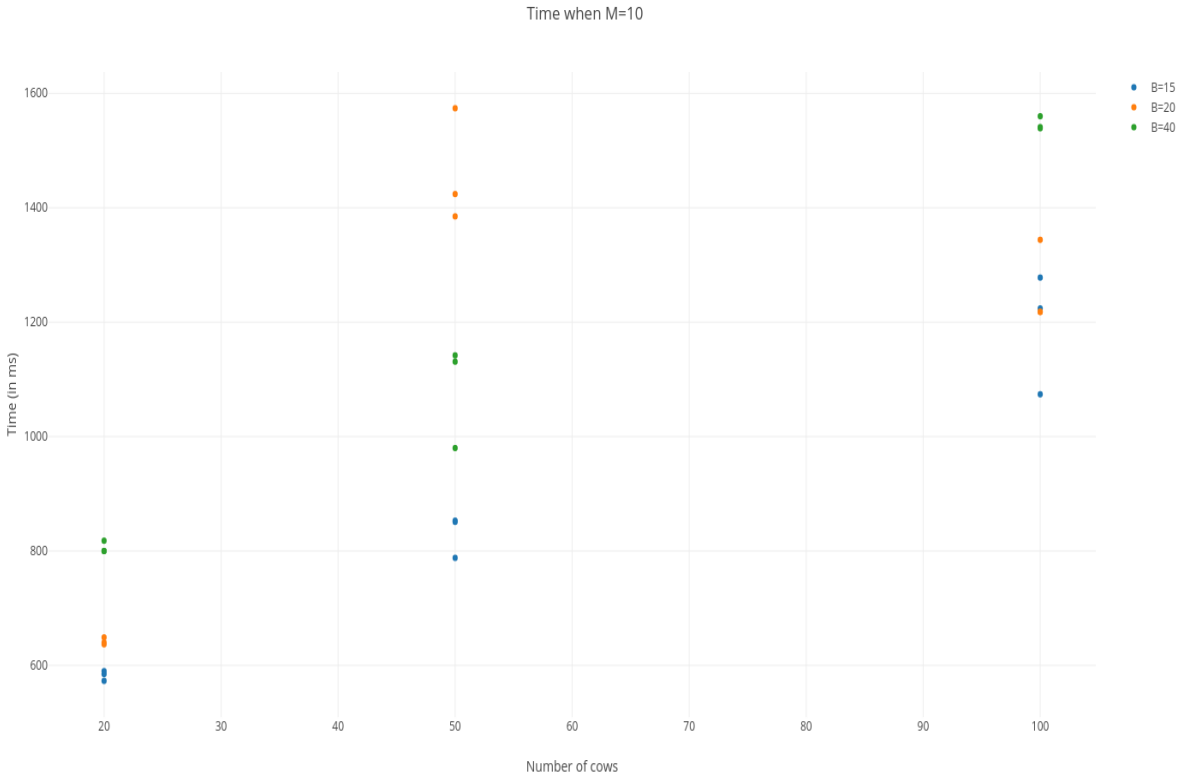


Figure 6.24: Third general validation - Time needed to find a solution when M=10

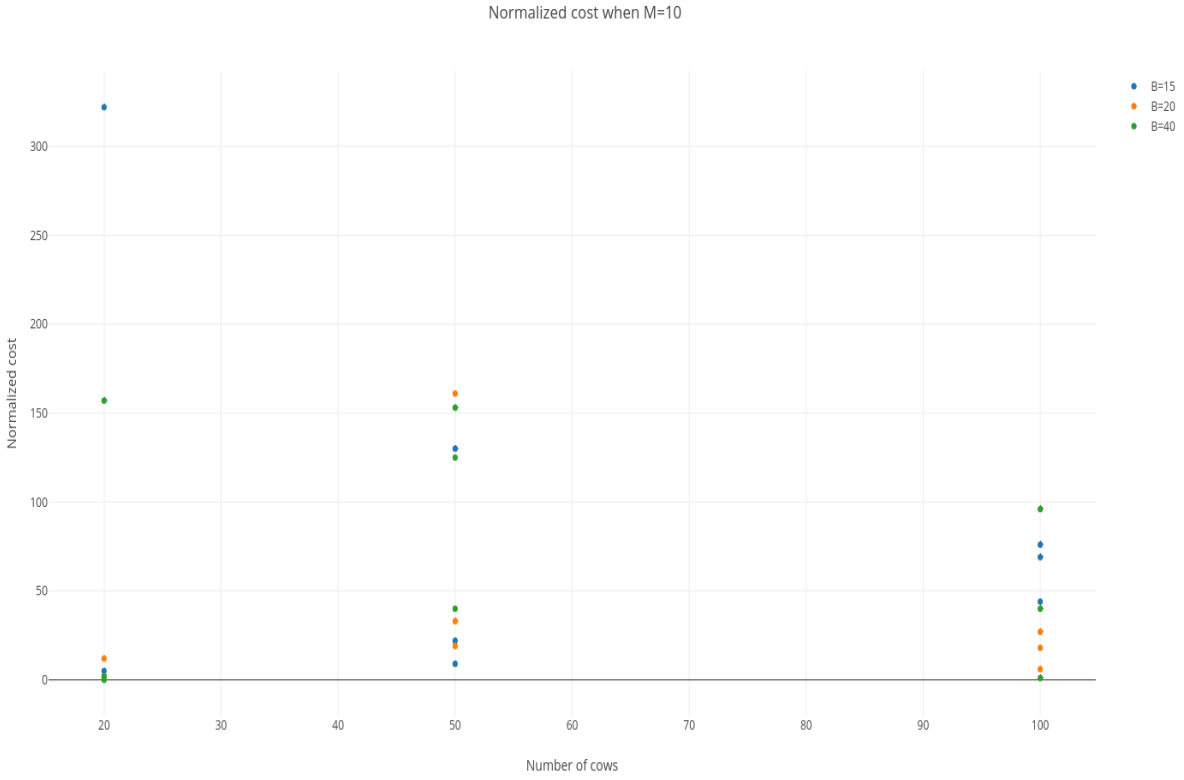


Figure 6.25: Third general validation - General cost to find a solution when M=10

Chapter 7

Conclusion

7.1 Objectives

First, we start this conclusion by looking at the objectives sets at the beginning of this work and how we reached them. The objective was to modelize, if possible, a real life problem concerning the reproduction of dairy cows. The task comported many different kind of selection criterion to be modelized and implemented, as well as the development of some tools to reach this objective. The main tool that we developed is a web parser that is adapted to the online resource CDN to fetch data that are not under the responsibility of the user, and that evolve with time.

We modelized the different constraints that compose the problem, then we implemented them with a modern framework called OscaR. We made sure that our work met the expectations by validating it through observations of the behaviour of our program under certain circumstances. Those observations lead to the assertion that our work is satisfactory and usable. Moreover, we made it adaptable to the needs of the user through parameterization.

7.2 Workload

During the year, the work that required the more time was clearly the consanguinity constraint and, consequently, the web parser. The creation of the genealogy graph, the retrieval and adaptation of the bull's data on CDN and the search through the genealogy graph to compute the consanguinity cost occupied us for a major part of the time we had allowed to this project. On the contrary, adding more constraints revealed itself to be pretty quick and easy, as we managed to use globally the same technique for each of them.

7.3 Difficulties and forces

One of the difficulties of this thesis was that the subject is deeply linked to a domain which is (or was) mostly unknown for us. We didn't know a lot of things about the dairy cow industry and learned a lot in order to understand what was asked of us. We had to learn fast and demonstrate good adaptability skills.

The title of the thesis being quite fun and unexpected in the field of computer science, we had the opportunity to test our synthesising and popularizing skills every once in a while to satisfy the curiosity of people around us.

7.4 To go further

This project could go further with the integration of more constraints, as some of the aspects of the reality have not been taken into account yet. Some of those aspects are the price of the insemination doses, their storage, their packaging, the discounts that might exist under conditions, other genetic criterion and probably many other things that we do not think of because we are not experts in the field.

The farm we worked with is used to work with digital tools. They even develop some tools themselves, their daughter being no other than Hélène Verhaeghe, one of the readers of this master thesis. This project could and will probably be integrated to existing custom tools in order to improve the system already existing.

Last but not least, this project could be adapted to fit the needs of more than one farmer. With the sufficient amount of data and with the help of experts in the field, this project could lead to a commercialised product. Though it lacks a GUI to make it more user friendly and thus, more widely and willingly used in the industry.

Bibliography

[1] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*. Global edition, third edition edition, 2016.

[2] Mats Carlsson Nicolas Beldiceanu. *Global Constraint Catalogue*, 2014. `http://www.emn.fr/x-info/sdemasse/gccat/Calldifferent.html`.

[3] J.-L. Laurière. *Un langage et un programme pour énoncer et résoudre des problèmes combinatoire*. PhD thesis, Paris 6 University, May 1976.

[4] Dr. Rong Qu. Constraint programming, cp techniques and search tree. Technical report, University of Nottingham, 2009.

[5] *The Scala programming language*. `https://scala-lang.org/`.

[6] OscaR Team. OscaR: Scala in OR, 2012. Available from `https://bitbucket.org/oscarlib/oscar`.

[7] Htmlcleaner transforms html to well-formed xml. `http://htmlcleaner.sourceforge.net`.

[8] Wikipedia the free encyclopedia. `https://en.wikipedia.org/wiki/Serialization`.

[9] Java SE Documentation. *Interface Serializable*. `https://docs.oracle.com/javase/7/docs/api/java/io/`

[10] Brian VAN DOORMAAL. La consanguinité au fil du temps. In *Centre de Référence en Agriculture et Agroalimentaire du Québec (CRAAQ)*, 2008. `https://www.agrireseau.net/bovinslaitiers/documents/Van%20Doormaal_Brian_AR.pdf`.

[11] Wallonie Elevage. *Élevage des Princes de Ligne*, december 2009.

[12] Mats Carlsson Nicolas Beldiceanu. *Global Constraint Catalogue*, 2014. `http://www.emn.fr/x-info/sdemasse/gccat/Celement.html`.

[13] National Conference on Artificial Intelligence. *Generality vs. Specificity: an Experience with AI and OR Techniques*, 1988.

[14] Mats Carlsson Nicolas Beldiceanu. *Global Constraint Catalogue*, 2014. `http://www.emn.fr/x-info/sdemasse/gccat/Csum.html`.

[15] Tallys H. Yunes. On the sum constraint: Relaxation and applications. In *Principles and Practice of Constraint Programming*, pages 80–92. Springer -Verlag, 2002.

[16] Mats Carlsson Nicolas Beldiceanu. *Global Constraint Catalogue*, 2014. `http://www.emn.fr/x-info/sdemasse/gccat/Catmost`$_n$*value.html*.

[17] International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. *Filtering Algorithms for the nvalue Constraint*, 2005.

[18] Bendigo Geoff Kroker and Lisa Clarke. Control of calving difficulty in beef heifers. Technical report, Agriculture Victoria, 2000.

[19] Réseau canadien de recherche sur la mammite bovine et la qualité du lait, editor. *Les capsules mammite*, chapter Que sont les cellules somatiques? Réseau canadien de recherche sur la mammite bovine et la qualité du lait, 2010. http://www.medvet.umontreal.ca/rcrmb/fr/page.php?p=104tm=i.

[20] Walter L. Hurley. *Milk Composition and Quality.* University of Illinois, 2009. http://ansci.illinois.edu/static/ansc438/Mastitis/milkquality.html.

[21] New South Wales Government - Department of Primary Industries. *Bull soundness - structural.* https://www.dpi.nsw.gov.au/animals-and-livestock/beef-cattle/breeding/bull-selection/structural-soundness.

[22] Rick Rasby. A guide to udder and teat scoring beef cows. *AngusJournal*, August 2011. http://www.angusjournal.com/ArticlePDF/Udders%2008$_1$1%20*AJ.pdf*.

[23] Mats Carlsson Nicolas Beldiceanu. *Global Constraint Catalogue*, 2014. http://www.emn.fr/x-info/sdemasse/gccat/Celement$_g$*reatereq.html*.

[24] J. N. Hooker G. Ottosson, E. S. Thorsteinsson. Mixed global constraints and inference in hybrid ip-clp solvers. In *Post-Conference Workshop on Large-Scale Combinatorial Optimization and Constraints*, pages 57–78, 1999.

[25] Pierre Schaus OscaR Team. Weighted sum. Online repository. https://bitbucket.org/oscarlib/oscar/src/tip/oscar-cp/src/main/scala/oscar/cp/constraint

[26] *Json4s documentation.* http://json4s.org/.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**