Fair Protocols for Sequential Allocation Problems

Honours project

Authors:
Sosha Happel
Marysia Winkels
Eva van Weel

Supervisor:
Ulle Endriss

University of Amsterdam

Contents

1 Introduction

During the first semester of this year we have been working on an Honours project focussing on fair protocols for sequential allocation problems in multi-agent systems, supervised by Ulle Endriss.

The project started off by understanding the article "*A General Elicitation-Free Protocol for Allocating Indivisible Goods*" by Bouveret and Lang [1]. This article is about a sequential allocation process, where a well-meaning, non-biased central authority has to divide a set of objects (indivisible goods) among a set of agents, whose preferences it does not know. The idea is to use a decentralised approach to divide the goods, which means the central authority designates an agent to pick an object among those that remain until all objects are allocated. The problem consists in choosing the "best" sequence of agents, according to some optimality criterion.

Section 2 contains most of the concepts used by Bouveret and Lang [1] on which this project is based, as well as our own thoughts and additions to this. We have written simulation software - as described in Section 3 - to find the "best" sequence of agents for various inputs. These results could then be used to make observations and recognise general patterns in these "best" sequences, which have been described in Section 4. The conclusion of the project can be found in Section 5.

2 Theory

In this section of the report, the notions of the article by Bouveret and Lang, [1] will be explained, aswell as our own additions to their article.

2.1 Terminology

We have a set of indivisible **objects** $\mathcal{O} = \{o_1, ..., o_p\}$ and a set of **agents** $\mathcal{N} = \{a_1, ..., a_n\}$. A specific agent is refered to as a_i, where i is the index of the agent in the set $\mathcal{N}$. During the **allocation process**, an agent from the set of agents is designated to pick an object from the set of objects at that moment. By picking an object, it is removed from the current set of available objects and is assigned to that specific agent. At the end of the allocation process, when all objects have been allocated, the final distribution of objects over the set of agents is called the **allocation**.

The order in which the agent places the objects according to his preference is called a **preference order**, ranking the objects from most to least prefered. We write $o_1 \succ o_2$ to express that agent a_i strictly prefers o_1 over o_2. If, for instance, the objects to be allocated are different kinds of fruits, one agent might prefer the banana over the pear and the pear over the apple. The preference order of this particular agent a_i would be represented as follows:

$$banana \succ pear \succ apple$$

Objects are usually, rather than refered to by name, denoted with their objectnumbers. The set of objects $\mathcal{O} = \{banana, apple, pear\}$, is represented as $\{o_1, o_2, o_3\}$, meaning the correct representation of the preference order mentioned above is:

$$o_1 \succ o_3 \succ o_2$$

A preference order of an agent lists all the available objects, and therefore has a length which is equal to the amount of objects available to be allocated.

At each **stage** of the allocation process, *one* agent is allowed to pick *one* object from the set of remaining objects, meaning that the number of stages is equal to the amount of objects. To determine which agent is allowed to choose an object at what stage in the allocation process, a picking order is defined. This picking order is called either a **protocol** or a **policy** interchangeably, and is represented by π. Each number in the protocol represents the index of a specific agent in the set of agents. An example of a protocol would be:

$$\pi = 122$$

In this protocol, the first number is a 1, meaning the first agent that is allowed to choose an object from the set of objects is agent a_1[1]. Both the second and third numbers in this protocol are 2, which means agent a_2 is allowed to pick the following two times.

We assume that each agent designated to pick an object, will always choose its most prefered object among those that are available. Additionally, none of the agents are aware of the other agent's preferences, thereby excluding the possibility of a strategical choice of object.

Example #1

Using this knowledge on agents, objects, preference orders and protocols, a simple example can be constructed to illustrate these concepts. Suppose we have three agents, $\mathcal{N} = \{a_1, a_2, a_3\}$, and five objects, $\mathcal{O} = \{o_1, o_2, o_3, o_4, o_5\}$. The protocol for this example is given to be $\pi = 12332$ and the preference orders of the respective agents are as follows:

$$\succ_{a_1} : o_1 \succ o_2 \succ o_3 \succ o_4 \succ o_5$$
$$\succ_{a_2} : o_4 \succ o_2 \succ o_5 \succ o_1 \succ o_3$$
$$\succ_{a_3} : o_1 \succ o_3 \succ o_5 \succ o_4 \succ o_2$$

With these given preference orders and protocol, and the assumption that each agent will always pick the best available object according to its own preference order, the allocation process will be as shown in Table 1. Each column in the table denotes an agent and by cross-referencing the agent's column and the stage of the allocation process, one can see what objects that specific agent posesses at that specific stage.

Stage	agent 1	agent 2	agent 3
1	o_1	-	-
2	o_1	o_4	-
3	o_1	o_4	o_3
4	o_1	o_4	$o_3 o_5$
5	o_1	$o_4 o_2$	$o_3 o_5$

Table 1: The allocation process for the given protocol and preference orders

As can be seen from the final row - indicating the last stage in the allocation process after which nothing changes - in Table 1, agent a_1 ends up with

[1]In this report, agent a_i is most often simply refered to as agent i

object $\{o_1\}$, agent a_2 with both objects o_2 and o_4 and agent a_3 with objects $\{o_3, o_5\}$.

2.2 Scoring function

Every agent has a preference order for the objects, but one might wonder how this preference order has been constructed. Each agent has a **utility function** to create its personal preference order. The utility function $u_i(o)$ is a function that determines the value that agent i gives to object o. Agent i prefers $o_1 \succ o_2$ if and only if $u_i(o_1) \geq u_i(o_2)$. This utility function is unique for every agent and while the utility function determines the preferences, it cannot simply be retrieved from the preference order of an agent. Instead, a **scoring function** is used to map from any given preference order to a corresponding utility function. Important to note at this point is that the agents have additive utilities, meaning the value of a subset of objects is equal to the sum of the values of its elements.

To map from a preference order of an agent to a utility function, the position of an object in a preference order is be used. This position of object o in the preference order of agent i is called the **rank**. The rank of object o in preference relation of agent i is denoted as follows, where p represents the total number of objects:

$$rank_i(o) \in \{1, ..., p\}$$

If an agent were to have the preference order $o_2 \succ o_1 \succ o_3$, object o_1 would get a rank of 2.

The ranks of the objects in the preference order of agent i provide us with parameter k which can be used in the scoring function, in addition to the total number of objects p, to calculate the utility for agent i. Rather than refering to the rank of an object in relation to a preference order as $rank_i(o)$, from this point onwards k will be used to represent the rank of object o for agent i.

Three types of scoring functions have been described in the article by Bouveret and Lang, [1]; **Borda**, **Lexicographic** and **Quasi-Indifferent**. In addition to this, we have also added our own scoring function **Fibonnaci**.

2.2.1 Borda

The Borda scoring function, named for mathematician and political scientist Jean-Paul de Borda's voting system, grants the first object in an agent's pref-

erence order (rank $k = 1$) a score equal to the amount of objects, the second object in an agent's preference ($k = 2$) one less, and so on until it reaches the least prefered object in an agent's preference order, which will be assigned a score of one. The end result is that the sum of some less desireable objects is actually quite likely to turn out to be higher than the score of solely the most prefered object.

The Borda score is calculated with the following formula:

$$g_B(k) = p - k + 1$$

In practice, for preference order $o_2 \succ o_1 \succ o_3$, the utility over the objects for agent i would be:

$$\begin{aligned} o_1 : g_B(2) &= 3 - 2 + 1 = 2 \\ o_2 : g_B(1) &= 3 - 1 + 1 = 3 \\ o_3 : g_B(3) &= 3 - 3 + 1 = 1 \end{aligned}$$

2.2.2 Lexicographic

The lexicographic scoring function ensures that an object at any rank will *always* provide a higher utility than the sum of the utilities of all the other objects below that ranking. This scoring function will be a good measure if the preferences of the agents are outspoken, meaning the agents prefer certain objects over other objects relatively much. The lexicographic score is calculated using the following formula:

$$g_L(k) = 2^{p-k}$$

In practice, for preference order $o_2 \succ o_1 \succ o_3$, the utility over the objects for agent i would be:

$$\begin{aligned} o_1 : g_B(2) &= 2^{3-2} = 2^1 = 2 \\ o_2 : g_B(1) &= 2^{3-1} = 2^2 = 4 \\ o_3 : g_B(3) &= 2^{3-3} = 2^0 = 1 \end{aligned}$$

For this particular agent, the posession of o_2 leads to a higher utility, than having both o_1 and o_3.

2.2.3 Quasi-indifferent

The QI or quasi-indifferent scoring function is a reasonable choice when the number of objects is of primary importance. Using this scoring function, it is always better for agents to acquire a lot of objects than a few higher ranked objects, meaning the objective of each agent is to collect as many objects as possible. The formula to calculate the QI score is:

$$g_I(k) = 1 + \epsilon \times (p - k), \text{where } \epsilon \ll 1$$

In practice, for preference order $o_2 \succ o_1 \succ o_3$, the utility over the objects for agent i would be:

$$\begin{aligned} o_1 : g_B(2) &= 1 + \epsilon \times (3 - 2) = 1 + \epsilon \times 1 = 1 + \epsilon \\ o_2 : g_B(1) &= 1 + \epsilon \times (3 - 1) = 1 + \epsilon \times 2 = 1 + 2\epsilon \\ o_3 : g_B(3) &= 1 + \epsilon \times (3 - 3) = 1 + \epsilon \times 0 = 1 \end{aligned}$$

As per definition ϵ is a very small number, the differences between the scores are minimal if agents all have the same amount of objects. As mentioned before, this type of scoring function is particularly useful if the number of objects is high.

2.2.4 Fibonacci

In mathematics, the Fibonnaci series is a series of numbers following the recurrence relation $F_n = F_{n-1} + F_{n-2}$, with seed values $F_0 = 1$ and $F_1 = 1$. Using this as a scoring function means that the utility of every object is equal to the sum of the following two objects in that preference order. This means that for every object, the value is at least equal and in many cases more than the combination of any two objects that are less prefered. This means this function is roughly similar to the lexicographic scoring function, in the sense that the sum of two lesser prefered objects is lower than the more prefered object, but less extreme as with the Fibonacci scoring function, the sum of all the remaining objects lower ranked objects (assuming there are more than two left) can in fact overtake the score of the better liked object. The score is calculated with the following formula:

$$\begin{aligned} fib(0) &= 1 \\ fib(1) &= 1 \\ fib(x) &= fib(x - 1) + fib(x - 2) \\ g_F(k) &= fib(p - (k - 1)) \end{aligned}$$

In practice, for preference order $o_4 \succeq o_2 \succeq o_1 \succeq o_3$, the utility over the objects for agent i would be:

$$
\begin{aligned}
o_1 : g_F(3) &= fib(4-(3-1)) = fib(2) = fib(0) + fib(1) = 1+1 &&= 2 \\
o_2 : g_F(2) &= fib(4-(2-1)) = fib(3) = fib(2) + fib(1) = 2+1 &&= 3 \\
o_3 : g_F(4) &= fib(4-(4-1)) = fib(1) &&= 1 \\
o_4 : g_F(1) &= fib(4-(1-1)) = fib(3) + fib(2) = 3+2 &&= 5
\end{aligned}
$$

2.3 Social welfare

We are now able to assign utility values to agents after having completed the allocation process based on the provided protocol, but as of yet, we still have no way of knowing whether this is a "good" policy, as we have not yet defined what we consider to be a good policy. Is a protocol good when all the agents are more or less equally happy with the allocation, or is it good if the highest possible combined utility is reached, even if it is at the expense of one or several agents?

To provide us with a measure of optimality for a protocol, **social welfare** is used. A protocol π is optimal if it maximizes the social welfare for a given allocation. The type of social welfare that is used depends on the objective. One objective could be to use to level of contentness for the *least* happy agent as standard. This is what **egalitarian** social welfare does. Another possibility is to look at the sum of the utilities of all the agents, which is called **utilitarian** social welfare.

The social welfare is calculated with **aggregation function** F, where u_i represents the utility u of agent i.

2.3.1 Egalitarian social welfare

Egalitarian social welfare measures the optimality of the division, by determining the utility level of the *least* happy agent.

Egalitarian: $F(u_1, ..., u_n) = min_{i=1,...,n} u_i$

2.3.2 Utilitarian social welfare

Utilitarian social welfare takes the sum of all the agent's utilities, meaning that a policy that is considered good from a utilitarian point of view sometimes sacrifices the utility level of individual agents, in order to maximize the total sum. Utilitarian social welfare might be an unbalanced measure, as it is easily manipulated by a few high utilities.

Utilitarian: $F(u_1, ..., u_n) = \sum_{i=1,...,n} u_i$

Example #2

Taking the data from Table 1 as a basis, the notion of scoring function and social welfare can easily be illustrated. In *Example #1*, the following subsets of objects were found for each agent:

$$1 : \{o_1\}$$
$$2 : \{o_2, o_4\}$$
$$3 : \{o_3, o_5\}$$

One can map from the given preference orders to the utilities for each agent. This has been done for the above sets of objects per agent and the results are shown in Table 2, where the rank of the objects is deduced from the preference orders of the agents.

Agent	Objects	Object rank	Borda score	Lexi score	QI score
1	$\{o_1\}$	$\{1\}$	5	16	$1 + 4\epsilon$
2	$\{o_2, o_4\}$	$\{1, 2\}$	$5 + 4 = 9$	$16 + 8 = 24$	$2 + 7\epsilon$
3	$\{o_3, o_5\}$	$\{2, 3\}$	$4 + 3 = 7$	$8 + 4 = 12$	$2 + 5\epsilon$

Table 2: Agent utilities depending on scoring function

With these utilities for objects in possession of the agents, the different types of social welfare can be computed. Table 3 shows the results for the two types of social welfare; egalitarian and utilitarian.

Social welfare	Borda score	Lexi score	QI score
Egalitarian	5	12	$1 + 4\epsilon$
Utilitarian	21	52	$5 + 16\epsilon$

Table 3: Social welfare

2.4 Uncertainty over profiles

As every agent has its own preference order, we need a way to define the set of the preference orders of all the agent in the set $\mathcal{N}$. This set of preference orders is called a **profile** R and is denoted as $R = \langle \succ_1, ..., \succ_n \rangle$, where $\succ_i$ represents the preference order of agent i.

In previous example, *example #2*, the preference orders of the agents were known in advance and because of this, the allocation process was fixed. As we do not always have the luxury of knowing the preference orders of the agents, there is an uncertainty over profiles. What can be given, rather than the preference order of each individual agent, is a type of correlation, which is a measure of dependency among the possible preference orders. The article by Bouveret and Lang [1] covers the two extremities of the spectrum; preference orders are either fully independent from one another (**full independence**) or the preference orders for all agents are identical (**full correlation**).

Additionally, we have thought about a type of correlation which is different from full correlation and full independence.

2.4.1 Full correlation (FC)

Under full correlation, all agents have the exact same preference order and will therefore want to obtain the same objects. This will inevitably lead to bad social welfare, since all the agents prefer the same objects. To calculate the probability of a certain profile R occuring for full correlation, the following formula is used:

$$Pr(R) = \frac{1}{(p!)}$$

With this probabibility Pr for profile R one knows the chance that profile R occurs for this amount of objects.

2.4.2 Full independence (FI)

Full independence describes the situation in which all profiles are equally probable and the rankings of the agents are independent. For full independence the uncertainty over profiles can be calculated with the following formula:

$$Pr(R) = \frac{1}{(p!)^n}$$

This means every profile has a $\frac{1}{(p!)^n}$ chance of being the true case for these agents and amount of objects.

2.4.3 A new type of correlation

Full correlation and full independence are two extrema of the spectrum. In many cases, a more "realistic" type of correlation would be one that would take into account that some objects are generally liked more than others, though not exactly as much. A practical example would be when the available objects are a laptop, a flatscreen tv, a piece of paper and a peanut. Generally, when you ask a certain amount of people at random how they would order these objects, the laptop and flatscreen will probably in most cases end either in the first or second place, while the peanut and the piece of paper will most likely be least prefered by a lot of people. However, to someone who is very hungry, the peanut might be more prefered than the piece of paper and to someone who is keen on drawing, the piece of paper might be more liked. A profile that is compliant with this would, realistically, have a higher chance of being true than others.

The Idea
A possible new form of correlation would be one that is based on a score, called the *similarity score*, which evaluates the average "distance" (difference in rank) between objects in the preference orders of two agents. Calculating the similarity score between all pairs of agents in a profile and averaging this out would possibly be some form of measure of similarity between the preference orders of agents within a profile. This measure of similarity can then be used to value some profiles, for example one with a higher or lower similarity score, as more likely, thereby increasing the chance of that profile being true, while deceasing the chances of profiles that differ much from the desired similarity score.

Possible Implementation
To implement this into our simulation software, one would add a possible correlation form to the input, which requires an extra parameter, in the range of zero to one, which is similarity score. Profiles with a similarity score nearer to the input value have a higher probability than those further away from it. The similarity score between the different preference orders is calculated by looking at the average distance between ranked objects divided by the total amount of objects.
The nearer the similarity score is to zero, the more similar two preference orders are. As a higher number would suggest a higher similarity, this is

counter-intuative and an alternative might be to make the similarity score one minus the previously calculated number.

$$1 : o_1 \succ o_2 \succ o_3 \succ o_4 \succ o_5$$
$$2 : o_4 \succ o_2 \succ o_5 \succ o_1 \succ o_3$$

Distances

$$o_1 = 3$$
$$o_2 = 0$$
$$o_3 = 2$$
$$o_4 = 3$$
$$o_5 = 2$$

$$\text{Average distance} = \frac{3+0+2+3+2}{5} = \frac{10}{5} = 2$$
$$\text{Similarity score} = \frac{2}{5} = 0.4$$
$$\text{One-minus alternative} = 1 - 0.4 = 0.6$$

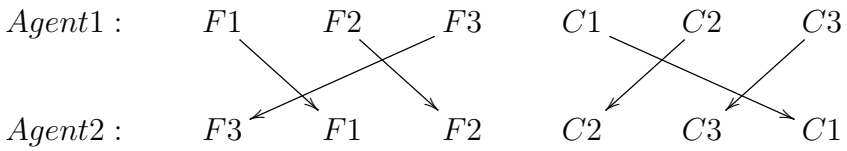
This is the similarity score between two agents, but we want the similarity score for the entire profile. Calculating that can done by taking the similarity score between each pair of agents in the profile, adding them up, and dividing them by the total amount of 'pairs of agents' to find an average similarity score.

With respect to Full Correlation and Full Independence
Full correlation would have a similarity score of zero, as the average distance between preference orders within the profile is per definition zero. However, this new type of correlation only ensures that profiles with a similar similarity score to the input variable have a higher chance of occuring, while full correlation fully excludes every profile in which the similarity score is not zero. This means this new type of correlation cannot represent full correlation.
It cannot represent full independence either, as the basic idea of this new type is that some profiles are more likely than others, while for full independence, every profile is equally probable.

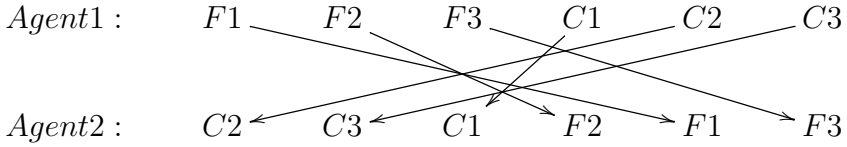
Possible Use
The similarity score gives weights to specific profiles, making some more likely to occur based on on how similar the average distance between objects in the preference orders of the agents is. It can also function to make a less-similar profile more likely than a similar one, which can be desirable in some instances.

An example of when this type of correlation would be useful is when the objects can be divided into different categories, and we are aware of the similarities between agents when it comes to category preferences. For instance, when the set of objects contain six books, three of which belong to the fantasy genre and three of which are cooking books.

CASE 1



CASE 2



The similarity score between the two agents in case one would be $\frac{1+1+2+2+1+1}{6^2} = \frac{8}{36} = 0.222$ or $1 - 0.222 = 0.778$, while the similarity score for case two is equal to $\frac{4+2+3+1+4+4}{6^2} = \frac{18}{36} = 0.5$ or $1 - 0.5 = 0.5$. A difference in category preference will lead to a higher value for the similarity score (or a lower number in the one-minus alternative). How exactly a profile with a similarity score nearer to the input value entered when calling for the function is set to have a higher chance of being true than a profile with a similarity score that differs more has not yet been decided. Ultimately, this type of correlation between agents only makes sense using the Borda or quasi-indifferent scoring function and - in most cases - only when having more than two categories of objects, for example two intermediate book categories such as science-fiction and historical novels, as the difference between the least prefered object in the liked category and the most prefered object in the least liked category should be in line with the chosen scoring function. In the case of lexicographical scoring, which returns a high difference in value between two consecutive objects and in which the sum of lower ranked objects can never overtake a

higher ranked object, the preference order of an agent matters much more than is the case with the Borda or quasi-indifferent scoring functions.

Up until now, all calculations have been done between the preference orders of only two agents, while in reality, a whole profile should be evaluated on similarity if this is to be used. We have not been able to implement this into our similation software yet, and therefore do not know whether this is, in reality, a useful addition.

The Downside and the Alternative
The dowinside of this method is that the similarity score principle only looks at high or low similarities between preference orders within a profile, and evaluates this based on averages. In some cases, certain objecs in a set are simply more likely to be highly evaluated by agents than others, as was briefly described in the opening example of this paragraph. For example, when a group of people is asked what character traits they would prefer their potential partner to have, humor and kindness are likely to end in the top regions of every interviewee, while egoistic and aggressive are likely to be in the bottom regions of each person's list. In this case, where one does not expect to find huge differences in the ranking of the objects, the similarity score alone would not be a good measure as this would also provide a high score (indicating a high similarity and therefore giving a higher probability to that profile occuring) for a profile consisting of preference orders with a large amount of objects, where most part of the preference orders among agents are more or less identical, with some exceptions. For example, a profile consisting of four preference orders for agents, ranking fifty objects, where only the first and the last objects are ranked differently, would be considered more probable to occur when using the type of correlation consistent with the similarity score, because on *average*, the distance between objects is pretty low. A realistic type of correlation to represent this would value profiles in which all objects are at the same or a neighbouring rank. Unfortunately, we have not been able to implement this.

2.5 Expected utility

With the probability for a certain profile, denoted as $Pr(R)$, one can calculate an **expected utility** for an agent i and protocol π. This utility is not the actual utility, but rather an expected utility, due to the uncertainty over the profile. The real utility of the agent i, might in fact be different. The expected utility can be calculated as follows, where $R(\mathcal{N}, \mathcal{O})$ is one profile out of all possible profiles for the set of objects and agents.

$$\overline{u(i,\pi)} = \sum_{R \in Prof(\mathcal{N},\mathcal{O})} Pr(R) \times u_i(\pi, R)$$

Example #3

A simplified version of the previous examples, with less agents and less objects, can be used to illustrate the effect of the different forms of correlation. To do this, we shall take a look at both full correlation and full independence, not taking into account our own possible form of correlation. As there is an uncertainty over profiles, the agent's preference orders are not known in advance. In this example, the Borda scoring function is used and the following set of agents, objects and protocol have been given:

$$\mathcal{N} = \{1, 2\}$$
$$\mathcal{O} = \{o_1, o_2, o_3\}$$
$$\pi = 122$$

Full independence
As there are three objects to be allocated, the number of possible preference orders per agent is 3!. This are all possible preference orders for the set of objects $\mathcal{O}$:

$$o_1 \succ o_2 \succ o_3$$
$$o_1 \succ o_3 \succ o_2$$
$$o_2 \succ o_1 \succ o_3$$
$$o_2 \succ o_3 \succ o_1$$
$$o_3 \succ o_1 \succ o_2$$
$$o_3 \succ o_2 \succ o_1$$

For two agents, any combination of two of these preference orders are a possible profile. The probability for any profile occuring is therefore equal to:

$$Pr(R) = \frac{1}{(3!)^2} = \frac{1}{36}$$

It is known that during the allocation process, the agent designated to pick will always pick his most preferred object. Therefore agent 1 will always get his most prefered object, leaving the second agent two possibilities; either his most prefered object has been taken by agent 1 ($\frac{1}{3}$ chance) or his most prefered object is still in the remaining set of objects ($\frac{2}{3}$ chance). In Table 4, the Borda score is calculated, taking these probabilites into account.

Agent	Probability for object × object rank	Borda score
1	$1 \times 3 = 3$	3
2	$\frac{1}{3} \times (3+2) = \frac{5}{3}$ $\frac{1}{3} \times (3+1) = \frac{4}{3}$ $\frac{1}{3} \times (1+2) = 1$	$\frac{5}{3} + \frac{4}{3} + 1 = 4$

Table 4: Borda score for full independence

The value for agent two has been calculated as follows: as agent 1 was allowed to pick first, there is a $\frac{1}{3}$ chance that agent 2's favourite object has been taken, leaving the (lesser liked) objects with rank 2 and 3 for agent 2. There is a $\frac{2}{3}$ chance that agent 1 chose a different object; $\frac{1}{3}$ chance that it was agent 2's second favourite object, leaving agent 2 with his most prefered (k=1) and least prefered (k=3) object and a $\frac{1}{3}$ chance that it was agent 2's least favourite object, leaving his most prefered (k=1) and second prefered object (k=2).

From these Borda scores, the expected social welfare can be calculated. It is called expected social welfare, because the Borda scores are calculated with a certain probability, leaving the social welfare with a certain probability aswell. The results are shown in Table 5.

Social welfare	Expected SW with Borda scoring function
Egalitarian	3
Utilitarian	7

Table 5: Social welfare for full independence

Full correlation
In full correlation, the preference orders of the agents are the same. This leaves the probability for a profile to:

$$Pr(R) = \frac{1}{(3!)} = \frac{1}{6}$$

Table 6 shows the Borda scores for full correlation, which differ slightly from the ones of full independence.

Agent	Probability for object × object rank	Borda score
1	$1 \times 3 = 3$	3
2	$1 \times (2+1) = 3$	3

Table 6: Borda score for full correlation

This ofcourse also leads to different expected social welfare, which can be seen in Table 7.

Social welfare type	Expected SW with Borda scoring function
Egalitarian	3
Utilitarian	6

Table 7: Social welfare for full correlation

3 Software implementation

The software was written with as many independently working functions as possible, so they could be tested seperately from each other. The main function performs two main tasks; determining the expected utilities and different possible policies with the given input, and returning the "best" policy along with the policies within five percent of the best score. The first is done by either calling for the appropriate subfunctions to calculate the expected utility and policies and writing them to the result folder if the function has not previously been called for with similar input parameters, or immediately reading the expected utility and policies from a `.dat` file in the result folder. When calling for the appropriate subfunctions, an estimation is given how much time the calculation will take, based on the amount of loop iterations necessary.

Secondly, the best policies according to the different forms of social welfare are selected and presented. When it concerns Full Correlation, only egalitarian social welfare is relevant to present as the utilitarian social welfare value will be equal for each policy. Then another function displays two graphs which presents the relationship between the average expected utilities or social welfares with respect to the maximal difference in occurance of an agent within a policy.

3.1 Representation

The representation of the program was set up in a way that deals primarily with the resulting social welfare score for every policy given an number of objects, number of agents and a scoring function. Since we are not concerned with specific agents receiving specific objets, the representation is very abstract.

Policies Policies are represented as an array containing integers indicating the agent picking that turn. The first agent that picks is number one, the second agent is number two and so on. The position in the array indicates the turn, starting with the first turn as the first element of the array. Example:

$$\pi = 12332 \rightarrow [1, 2, 3, 3, 2]$$

Preference Orders Preference orders are represented as an array containing intigers indicating an object, the position in the array determines the preference, with the first element as the highest rank. Example:

$$o_4 \succ o_2 \succ o_5 \succ o_1 \succ o_3 \rightarrow [4, 2, 5, 1, 3]$$

Profiles Profiles are represented as an array containing preference orders (array) for each agent, with the position indicating the agent. Example:

$$\begin{array}{ll} a_1 : o_1 \succ o_2 \succ o_3 \succ o_4 \succ o_5 & \\ a_2 : o_4 \succ o_2 \succ o_5 \succ o_1 \succ o_3 \rightarrow & [[1, 2, 3, 4, 5], [4, 2, 5, 1, 3], [1, 3, 5, 4, 2]] \\ a_3 : o_1 \succ o_3 \succ o_5 \succ o_4 \succ o_2 & \end{array}$$

Object Allocation When we run through the program we store the object allocation for every policy and profile in an array containing integers indicating the agent that got the object corresponding to the index. Example:

$$\begin{array}{ll} a_1 : \{o_1\} & \\ a_2 : \{o_2, o_4\} \rightarrow & [1, 2, 3, 2, 3] \\ a_3 : \{o_3, o_5\} & \end{array}$$

3.2 Implementation

The implementation can be divided into three sections:

1. Pre-calculation
    - Find all protocols
    - Find all Profiles
2. Main loop
    - Allocate objects

- Score agents
- Calculate social welfare

3. Analysis

The first section allocates all objects to agents for every protocol and profile, the second section calculates the social welfare of the object allocation and the third section gives an analysis of the result to provide extra information on how different protocols relate to each other.

Data: n = number of agents, p = number of objects, g = scoring function, c = correlation
Result: Social welfare score for each protocol
Protocols ← `FindAllProtocols(`*n, p*`)`;
Profiles ← `FindAllProfiles(`*n, p, c*`)`;
foreach *protocol pt in Protocols* **do**
  foreach *profile pr in Profiles* **do**
    $A_{pt,pr}$ ← `AllocateObjects(`*pt, pr*`)`;
    $S_{pt,pr}$ ← `AgentScores(`$A_{pt,pr}$*, pr, g*`)`;
  end
  SW_{pt} ← `SocialWelfareScores(`$S_{pt,-}$*, sw*`)`;
end

Algorithm 1: Main

3.2.1 Pre-calculation

FindAllProtocols In order to allocate objects for all protocols, the algorithm has to do a seperate run for each protocol of which there are n^k in total. Since we are not interested in knowing the score of a specific agent for every protocol, only the order of agents matter. In other words, it does not matter whether, for example, *Bob* or *Robot1* picks first, it only matters which agent picks in relation to the previous picks. By naming the agents after the order in which they pick we can look at the problem in a more abstract manner. As a result, every case where $n \geq k$ is treated as $n = k$ and every protocol where the order is equivelant, is treated as one case. This reduction has no effect on the resulting social welfare score because every agent has the same value in social welfare, that is to say, every individual score counts equally in the final result.

Example of protocols with an equivelant order:

$$1 : \{1, 2, 3, 3, 2\}$$
$$2 : \{3, 2, 1, 1, 2\}$$
$$3 : \{2, 3, 1, 1, 3\}$$

Example of protocols with a unique order:

$$1 : \{1, 2, 3, 3, 2\}$$
$$2 : \{1, 2, 3, 1, 2\}$$
$$3 : \{1, 2, 1, 2, 3\}$$

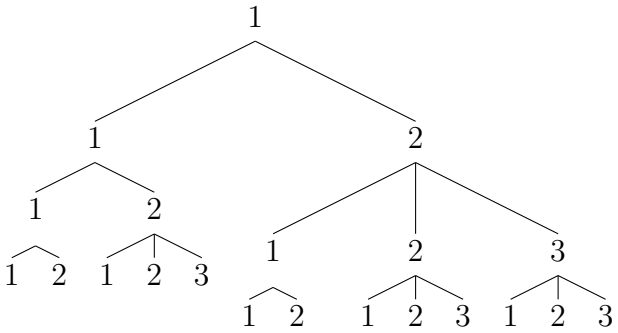
Finding these unique order protocols can be compared to constructing a tree of which the every node has a value starting with the root node containing value 1. The depth of the tree is equal to the number of objects. Every parent node has the following set of children:

$$\{\forall x \| v \geq x \vee p \geq x\}$$

Where v is the value of the parent node and p is the number of agents.

Every path from the root node to one of the leaf nodes represents a unique order protocol. An example of a tree can be found in figure 1 and the algorithm to find the protocols can be seen in algorithm 2.

Figure 1: Protocol tree with number of objects = 4 and number of agents = 3



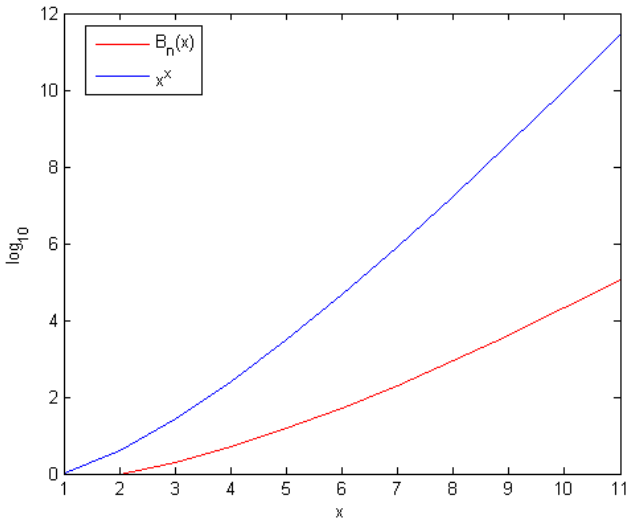
```
Data: n = number of agents, p = number of objects
Result: AllProtocols
P ← {{1}};
for i ← 2 to p do
    P_new ← {};
    foreach protocol pr in P do
        m ← max(pr);
        if m < n then
            A ← 1 : m + 1;
        else
            A ← 1 : m;
        end
        foreach agent a in A do
            pr_new ← Append(a, pr);
            P_new ← Add(pr_new, P_new);
        end
    end
    P ← P_new;
end
```

Algorithm 2: Constructing all protocols

The number of protocols resulting from this method is highest when $n \geq p$. In this case the amount of protocols correspond to $B_n p$, where B_n stands for Bell-numbers[2]. In this worst-case, the resulting number of protocols without the reduction would be equal to p^p. Figure 2 shows the amount of protocols per method for $p = 1 : 11$.

[2] $B_n x$ correspond to the number of all possible partitions of the set $\{1 : x\}$.

Figure 2: worst-case number of protocols with and without reduction



FindAllProfiles When looking for all relevant profiles we can do the same as with protocols, since the value of a certain preference is equal for all agents. This means that we can say that agent 1 always has the same preference order without changing the outcome of the social welfare. With full correlation all agents have the same preference order and because agent 1 can always have the same preference order, they can all have one preference order and the result will be the same. So in this case the number of unique profiles is 1. With full independence this reduction has less of an impact. Normally, the amount of profiles would be $(p!)^n$, with the reduction this changes to $(p!)^{n-1}$. This is still a relevant reduction since the algorithm can only run with very small input. In order to find all these profiles we have to find all combinations of $1:n$ of all permutations of $1:p$

3.2.2 Main loop

The main loop runs through every combination of protocols and profiles. For full correlation we showed that running through one profile is enough to get the social welfare score. So in this case the main loop runs for $B_n p$ in the worst case where $n = p$. For full independence we are still left with $B_n p * p!^{n-1}$.

AllocateObjects The assignement of objects to agents is a matter of going through the protocol and for every pick determine the highest ranked object available for the agent picking. This object is then assigned to the agent and

removed from the list of available objects.

AgentScores After all the objects are allocated we can determine the expected utility for each agent using the scoring function from the input.

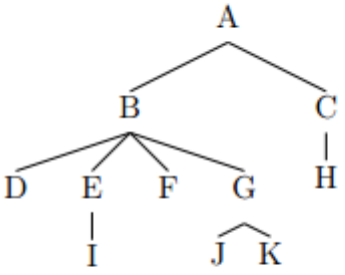
SocialWelfareScores When all the profiles are run for a certain protocol, the social welfare score for that protocol can be determined using all the expected utilities per agent and profile and the social welfare function.

3.2.3 Analysis

To help find patterns in the best protocols and we ran some analysis on the resulting social welfare of protcols to show the effect of picking frequency per agent and finding all protocols that were within a margin of 0.95 of the best protocol.

3.2.4 Program structure

Figure 3 gives an overview of the functions used in the simulation software and shows which function calls what function.



$A : PolicyAnalysis$
$B : expectedUtilityPerPolicy$
$C : agentFrequencyAnalysis$
$D : allPreferenceOrders$
$E : allProfiles$
$F : allPolicies$
$G : allocateObjects$
$H : agentFrequencyPerPolicy$
$I : allProfilesFI$
$J : bestAvailableObject$
$K : scoreObject$

Figure 3: Program structure

3.3 Program call

To run the software one needs to define the input parameters. For the main function *PolicyAnalysis*, four parameters are needed: the number of agents

$\mathcal{N}$, the number of objects $\mathcal{O}$, the scoring function g and the correlation c. A call to *PolicyAnalysis* would look like this:

```
>> PolicyAnalysis(AgentNum, ObjectNum, ScoringFunction, correlation)
```

As output the software returns the best policies for both egalitarian and utilitarian social welfare in the case of full independence. When the user has chosen full correlation as input parameter, only the best policy for egalitarian social welfare will be returned, since the utilitarian social welfare is always the same for full correlation.

$$\begin{aligned} \pi &= 122 \\ 1 &= 3 \\ 2 &= 2 + 1 \\ utilitarian &= 3 + 2 + 1 = 6 \end{aligned}$$

$$\begin{aligned} \pi &= 121 \\ 1 &= 3 + 1 \\ 2 &= 2 \\ utilitarian &= 3 + 1 + 2 = 6 \end{aligned}$$

Because of this phenomenon, it is not interesting to show the optimal policies for utilitarian social welfare under full correlation. All policies are equally good.

With the software, it is possible to verify whether policy $\pi = 122$ used in example #3 is an optimal policy for $\mathcal{N} = \{1, 2\}$, $\mathcal{O} = \{o_1, o_2, o_3\}$, $g = Borda$ and $c = FI$. Running these parameters with the simulation software yields the following results:

```
>> PolicyAnalysis(2, 3, 'borda', 'FI')

==UTILITARIAN==
best policy:
     1     2     1


==EGALITARIAN==
best policy:
     1     2     2
```

As can be seen policy $\pi = 122$ is an optimal policy for egalitarian social welfare, but not for utilitarian social welfare. The program also shows all the policies, including the best one, that are in a five percent range of this best policy. Remember that the best policy is the one that maximizes the chosen social welfare. For the above example, the policies that are in a five percent range of the best policies are:

```
==UTILITARIAN==
All policies within 5 percent of best score:
     1     1     2
     1     2     1
     1     2     2

==EGALITARIAN==
All policies within 5 percent of best score:
     1     2     2
```

For utilitarian social welfare, there are a couple of policies that are close to the optimal one in maximizing the social welfare. As can be seen, these other policies are similar to each other; $\pi = 112$ and $\pi = 122$. Table 8 shows the optimal policies for various values of p and n under full independence, Borda scoring function and egalitarian social welfare.

p	n = 2	n = 3
3	122	123
4	1221	1233

Table 8: Optimal policies for various value of p and n under full independence, Borda and egalitarian social welfare

The optimal policies for utilitarian social welfare are shown in Table 9.

p	n = 2	n = 3
3	121	123
4	1212	1231

Table 9: Optimal policies for various value of p and n under full independence, Borda and utilitarian social welfare

4 Observations

After succesfully implementing the theory into simulation software, which could handle up to about 4 objects and 3 agents for full independence, and 10 objects and 5 agents for full correlation, we used this software to make some observations.

4.1 Full Correlation

The results of full correlation gave us a good understanding of the format of optimal protocols and for every scoring and scoial welfare function we can now determine the optimal protocol without running the allocation.

4.1.1 Utilitarian

For utilitrian social welfare under full correlation it is not hard to show that every protocol will have an equivelant social welfare score. Because every agent has the same preference order and we know that evaluating one preference order gives us the result of all preference orders. Every object that is picked will get a set amount of score regardless of the agent picking the object. In utilitarian social welfare the expected utility of an individual agent is irrelavant to the social welfare since it consists of the sum of the expected utility of all agents. Therefore each protocol is equal under full correlation and utilitarian social welfare.

4.1.2 Egalitarian

With egalitarian social welfare under full correlation we only consider the case where $n < p$. If this is not the case there will allways be at least one agent that receives no objects and therefore the social welfare score equals zero. For every scoring function the results showed us a pattern for the optimal protocol and we were able to formalize them.

4.1.3 Lexicographic

Using the lexicographic scoring function under full correlation always resulted in the same pattern. First, every agent but the last one would be allowed to pick one object, and all the remaining objects would go to the last agent. This makes sense when realising that the lexicographic scoring function means that a higher ranked object will *always* get a higher rating that the sum of all the lower ranked objects. The first agent will pick his most

desired object, and therefore get the highest score possible with one object. Even if the first agent would be allowed to pick all the remaining objects as well, he would not even be able to double his score. The agent following the first agent will, as the correlation is full correlation, not be able to pick the most desired object, but choose the second best. When the last agent is allowed the pick, all previous agents have chosen objects with a higher ranking than the ones still available, which means their score is higher than the sum of the remaining objects. The only option left to make the division still as fair as possible, is to allocate all the remaining objects to the last agent, as his score will never surpass any of the previous agent's scores.

For instance, when running `PolicyAnalysis(4,6,'lexi','FC')`, the best policy returned under egalitarian social welfare is $\pi = 123444$, of which the (expected) utilities for each agent can be seen in Table 10.

Agent	Score		
1	$g_L(1) = 2^{6-1}$	$=$	32
2	$g_L(2) = 2^{6-2}$	$=$	16
3	$g_L(3) = 2^{6-3}$	$=$	8
4	$g_L(4) = 2^{6-4}$	$=$	4
	$g_L(5) = 2^{6-5}$	$=$	2
	$g_L(6) = 2^{6-6}$	$=$	1 +
		$=$	7

Table 10: Table showing the expected utilities for each agent for four agents, six objects, and using a lexicographic scoring function under full correlation.

In short, using the lexicographic scoring function under full correlation ensures that the last agent to pick will always have the lowest utility. In order to maximize the expected egalitarian social welfare, the expected utility of the agent who is certain to have the lowest utility should be maximized. This can be done by designating the last agent to pick all the remaining objects, ensuring a score as high as possible.

4.1.4 Fibonnaci

The result of fibonacci scoring under full correlation and egalitarian social welfare was similar to that of lexicographic. First, all agents except two pick one object: $s = \{1, ..., n-2\}$. Then the remainder of the picks, $m = p - (n-2)$, are divided among the last two agents, a_{n-1}, a_n, in sets of three

consecutive picks, $q = m \bmod 3$, and a possible remainder of one or two picks. Every set, one of the two agents gets the first pick and the other agent the remaining two.

$$\sigma = \{n-1, n, n\}$$

After all the sets, depending on p, there could be one or two picks left, these are divided among a_{n-1}, a_n as equally as possible.

$$r = \{\} \vee \{n-1\} \vee \{n-1, n\}$$

Resulting in the following formalized pattern for the optimal protocol:

$$\pi = \{1, ..., n-2\}\sigma_1, ..., \sigma_q, r$$

Example:

$$\pi = 1234434434, (s = 12, \sigma_1 - \sigma_2 = 344, r = 34)$$

Here, the first $n-2$ agents get an expected utility equal to $g_F(x), x \in \{1, ..., n-2\}$ of which the lowest will be $g_F(n-2)$. The last two agents will receive approximatly half of the the remaining score because they both get the same score for every three objects picked since each rank is equal to the sum of the two following ranks in score. The remaining one or two objects will result in a difference of 1 maximum. This means that from an egalitarian point of view under full correlation their respective expected utility is optimized.

$$u_n, u_{n-1} = \frac{\sum_{i=n-1}^{p} g_F(i)}{2}$$

It can be shown that $g_F(n-2)$ is always more then u_n and u_{n-1}. In order to proof this we can look at F_x, the mathmatical function for fibonacci, and use the following facts that are proven for F_x, where $x > 1$:

$$\sum_{i=1}^{x} F_i = F_{x+2}$$

$$F_x > \frac{1}{2} F_{x+1}$$

We write the expected utility in terms of F_x:

$$\begin{aligned} u_{n-1}, u_n &= \frac{1}{2} \sum_{i=n-1}^{p} g_F(i) \\ &= \frac{1}{2} \sum_{i=1}^{m} F_i \\ &= \frac{1}{2} F_{m+2} \\ u_{n-2} &= g_F(n-2) \\ &= F_{m+1} \end{aligned}$$

Now we see that it is proven that $u_{n-2} > u_n, u_{n-1}$. Since agent a_{n-2} only receives one object it can not loose any more of its 'share' without getting less then the last two agents and since the last two agents get a near equal expected utility these protocols are optimal.

4.1.5 Quasi Indifferent

This is an observation we did not make ourselves, but came from the original article [1]. The best policy for quasi indifferent scoring function under full correlation can always be described by combining two subpolicies. The first subpolicy is of length $(n-1)*m$ where $m = \lfloor \frac{p}{n} \rfloor$. This subpolicy is the same as the optimal policy for Borda under full correlation, with $(n-1)*m$ agents and $((n-1)*m)*m$ objects. Every agent in this subpolicy gets allocated m objects.

The second subpolicy assigns $m+1$ objects to the remaining agents, in any order.

4.1.6 Borda or Quasi-Indifferent with 2n = p

In the specific case that there are twice as many objects as there are agents, both the Borda and quasi indifferent scoring functions behave in the same way when under full correlation. All the agents get to pick exactly twice, first in normal order, than in reverse order. This means the first agent gets to pick first and last, the second agent second to first and second to last, and so on. As Borda is a linear function (and quasi indifferent scoring is partly similar to Borda when under full correlation), these will result in exactly the same score.

For example, when running `PolicyAnalysis(3,6,'borda','FC')`, the best policy returned will be $\pi = 123321$, as can be seen in Table 11. The relevance of this observation is that, even though our simulation software is not capable of calculating the best policy for *n=50,p=100*, we can nevertheless return it.

Number of agents and objects	Best policy
n=1, p=2	11
n=2, p=4	1221
n=3, p=6	123321
n=4, p=8	12344321
n=5, p=10	1234554321
n=9, p=18	123456789987654321

Table 11: Optimal policies p objects and n agent under both the borda and quasi-indifferent scoring function

4.2 Full Independence

The patterns in the returned best policies under full independence rarely seemed to hold up as we tried more variations. For instance, at first it seemed as though the best policy based on expected egalitarian social welfare using the Borda scoring function under full independence was always the policy of an object less, with one additional pick for an agent. For example, when *n=3* and *p=4*, the optimal policy was $\pi = 1233$. For *n=3* and *p=5*, it was $\pi = 12332$ and with *p=6* it was $\pi = 123321$. Continuing this pattern, one would expect to see the optimal policy for *n=3* and *p=7* start with 123321, but the actual best policy for these input values is $\pi = 1122332$.

Another trend we saw was that the average expected social welfare went downhill as the maximum difference between how many times agents were allowed to pick was raised. However, we soon saw that it was premature to assume that policies with a high maximum difference in how often the agents occured in the policy could be disregarded, as some of the policies that completely excluded certain agents sometimes did appear in the top five percent policies (looking at utilitarian social welfare).

4.2.1 Borda

Observation suggests that when using the Borda scoring function under Full Independence, the best policy when only considering the expected utilitarian

social welfare is always a strict alternating pattern, as can be seen in Table 12

Number of objects	n = 2	n = 3
2	12	12
3	121	123
4	1212	1231

Table 12: Optimal policies for p objects and n agents

5 Conclusion

During this project, we have succesfully managed to come to understand the article by Bouveret and Lang about a fair protocol for the division of goods. In addition to that, we have created a piece of software that can provide the user with the 'fairest' protocol given the necessary input. Lastly, we have managed to come up with our own ideas which were different from the ones in the article. There is still room for improvement; when it comes to the software, it would be nice to be able to run it with larger input values, and when it comes to the theoretical side, we feel that only considering full correlation and full independence is not realisticenough for the practical applications calculating the best protocol might have.

Acknowledgements

We want to thank our supervisor Ulle Endriss for his patience and support during this Honours project. We also want to thank Raquel Fernández for her coordination of the Honours Program and all the improvements that she has made to it.

References

[1] Sylvain Bouveret and Jérôme Lang. A general elicitation-free protocol for allocating indivisible goods. *IJCAI*, pages 73–78, 2011.