CARLETON UNIVERSITY

Animal Care System
CARLETON UNIVERSITY

# System Design Document

**Team Animal Farm**

Azim Baghadiya - 101044100
Cameron Hawtin - 101047338
Nicolas Lalonde - 101031228
Nick Simone - 101034935

Submitted to:
Dr. Christine Laurendeau
COMP 3004 Object-Oriented Software Engineering
School of Computer Science
Carleton University

April-9-2019

# Table of Contents

# 1. Introduction

## 1.1 Project Overview

Animal shelters provide a crucial service for homeless animals awaiting adoption into a loving home, as well as for the people who are seeking the comfort and companionship that a beloved pet can offer. Sometimes however, adoptions can go wrong, due to a personality mismatch between the animal and the client.

The Carleton University Animal Care System (cuACS) proposes to address this issue by providing a tool that automatically pairs together, based on compatibility, shelter animals and the clients who wish to adopt them. The goal of the cuACS system is to generate an optimal set of pairings (referred to as a matching), where a pairing consists of an animal available for adoption and a human client who is well suited to adopt it.

The cuACS system supports two categories of users: clients and shelter staff, and facilitates a few key features. These include the ability of staff to manage the shelter's animals and their detailed profile information, as well as to assess animal-client compatibility and compute an optimal set of animal-client pairings. The system also allows clients to manage their profile information, including personal data and matching preferences.

The Animal-Client matching(ACM) algorithm uses the profiles of all clients and animals in order to compute the best possible matching between animals and clients. The ACM algorithm uses animal and client profiles to compute the best matching, and it outputs the results to the shelter staff. These results must consist of both a summary of pairings, and the details of a selected pairing. The summary information indicates the names of the matched pairs of animals and clients. The detailed information for a given pairing specifies the exact rules that were used to compute that pairing, as well as the data supporting how and why that specific pairing was computed.

The cuACS user interface (UI) will be graphical in nature. The user features should be easily navigable, either as menu items and/or pop-up menus. The cuACS system will run on a single host, with all its data stored on that same host. Data will be stored in persistent storage (flat files) and will be loaded into the UI when cuACS launches. Modifications to data in the UI will be saved to persistent storage after every change.

Animal attributes include physical attributes, such as type of animal, breed, gender, age, and size. They also include non-physical attributes that will relate to temperament, personality traits, habits, etc. Specific values for these attributes will be included in every animal profile. A client profile will contain specific values for the attributes that are relevant to humans in the adoption process, as part of the client's personal data.

Each client profile must also contain the client's matching preferences, which are the values for the attributes that the client is looking for in an adopted animal.

All the attributes defined in the animal and client profiles will be used by the algorithm's rules to compute the best matching. This includes a client's own attributes, as well as their matching preferences. The algorithm must use its own unique set of rules for matching together animal and client profiles, based on the attributes and matching preferences contained in those profiles.

## 1.2 Document Overview

This document details the algorithm used for generating the optimal matching given a list of clients and animals. To find the optimal matching, it is imperative to be able to compare different matchings. This can be done by devising a way to assign a score to any arbitrary matching, this way, we can confidently compare two matchings to determine the better one. This allows us to pick the optimal matching with the certainty that no other matching would be better, which we can check by comparing their scores. As matchings are simply a set of individual pairings, a matching's score is bound to be determined by the composition of its pairs, with more favorable pairs resulting in a higher matching score. To determine just how favorable a pair is, a score is assigned to each one, which allows a comparison between different pairings. These scores are generated from an analysis of how well various attributes of the animal mesh with those of a client in order to produce a successful adoption. These attributes were carefully chosen in order to maximize the confidence of a pairing resulting in a successful adoption, as well as to minimize the risk of missing important details which could result in an unhappy adoption. To ensure that our algorithm is satisfactory with no missing aspects, a very clear problem statement is given and referred to throughout the text.

The document first highlights the attributes that were selected for the animals and clients. Details are given about the reasons leading to the choice of each animal attribute. Corresponding human attributes are also given, and more complex attributes explained. We then discuss what information users enter into the system, and how this information is transformed to result in a value for each attribute.

Following this, a discussion of the methods used in scoring individual pairings is presented. In this section, we explain how we normalize the attribute values into manipulatable numbers. We then explain how these normalized values for the client and animals are compared to generate scores for each attribute-attribute pair. Finally, we explain how each of these pairs are weighted and the reasoning between each individual weight

After establishing a method for scoring pairings, we then discuss our method for scoring matchings. We explain how the pairings that make up a matching help determine its score. We then explain the methods we use to bias the score in favour of more good pairings rather than a few excellent pairings mixed with a few terrible ones. We define a good pairing and use this definition to transform the scores of matchings to reflect whether they are composed of mostly good pairings.

Next, we discuss how to determine the best matching from all possible matchings. This boils down to an optimization problem where we want to optimize the score of the matching. In this section, we discuss multiple optimization algorithms as well as their respective advantages and disadvantages. The appropriate algorithm is chosen and we explain the reasons behind this choice.

We conclude the document with an overview of our chosen algorithm. We explain each step of the process. First, how the information provided by the user is turned into the values for the attributes given to the animal and client. Second, how these values are used to generate a score for each animal-client pairing. Third, how these scores are pre-processed to bias the algorithm to contain a higher number of good matches. Finally, how the resulting scores are used to determine the best possible matching. We give implementation notes throughout.

At the end of the document, a glossary of terms is provided. It may serve the reader to glance at the glossary before continuing their reading, as it will help clarify terms which have a special definition in this text.

## 1.3 Problem Statement

Given a set of clients and a set of animals, find an appropriate set of pairings, meaning that all the animals will be paired with clients that will give them a happy home, based on traits of both the client and animal. The happier the home, the better. Solutions with a high number of good pairings should be favoured over one with a few excellent pairings and a few poor ones. In essence, the goal is to give every animal a good home.

# 2. Animal and Client Attributes

## 2.1 Animal Attributes

While aiming to generate an optimal set of matches for the clients and animals, it's important to consider a variety of criteria that collaboratively contribute to make a good animal-client pairing. The following attributes are proposed for this purpose.

Let's first look at the attributes for an animal. It is essential to not only consider the general physical attributes but also the more important unique and non-physical attributes.

### Animal Type

This is possibly the most important attribute of all. We will later use it as a heuristic in our matching algorithm. Animal type plays a very crucial role in the matching decision; for instance, if a client prefers a dog, matching that client with a cat or a parrot is probably not a very good decision.

### Aggression

Some clients may prefer aggressive animals if they are to serve as guards or hunting partners, while some clients may prefer less aggressive and more caring animals to play with the kids at home.

### Attachment

Some clients may spend too little time at home and wouldn't want their pets to be extremely attached to them while some clients are looking for a pet who can be a companion and always by their side.

### Obedience

Some clients may want to be able to tell their animal exactly what to do and have it listen to them and some clients may prefer a more independent pet likes to express his own personality.

### Energy Level

Some clients may like their pets to be extremely energetic and in the mood to play physical games while some may like animals that are more peaceful and like to relax at home with their owner.

### Crate-Trained

This attribute can be crucial for many clients who travel a lot and need their pet to be crate-trained so it can accompany them where they go. However, some clients might not travel too much and wouldn't mind if the pet is relaxing inside the house most of the time.

### Hypoallergenic

This is again an extremely essential attribute as many clients may be allergic to a certain kind of animal type (for instance, a client may be allergic to cat fur), and living with a pet that causes an allergic reaction in the client could lead to resentment towards the animal.

### Neutered

This attribute plays a crucial role in the adoption decision as some clients may encourage their pets to have children while some prefer not to risk any unintentional and possibly bothersome little pets.

### Good With Children

While some clients are childless and are looking for a companion or a guard, a lot of families are also looking for pets which can play with their children.

### Loudness

While some clients may not mind the noise their pet may make, some clients prefer going to bed early and avoid getting disturbed late night after a certain time. In case, they would prefer an animal that is not very loud.

### Cost

Each client will have their own budget and the cost of the animal might affect their willingness to adopt it.

### Average Cost Per Year

Each animal has a different pattern of eating habits, need different vaccinations at different time, etc. So these bring an additional cost (apart from the sale price) every year which may vary and the client's budget may affect his ability to support the needs of the animal without having to cut back on other things.

### Intelligence

Most clients may not care how intelligent their pet is beyond a certain point, and only want one for the company and love it can provide, whereas some clients with more niche interests, such as training a pet for a contest or show, or simply getting their pets to perform tricks, might prefer an animal which has been shown to be a quick learner.

### Cleanliness

Clients have different sensitivities to the amount of mess they can handle. Clients with more free time may be able to handle cleaning up after a messy animal, while busy clients may have less time for this. Clients may also differ in their tolerance to a certain level of dirtiness.

### Playfulness

Some clients may have the desire to spend time playing with their pets. Clients with more free time will be able to spend more time doing this - so the animal will not become bored or lonely.

### Loyalty

The loyalty of an animal can be an important factor for a client who is seeking either a companion, guard or hunting pet. For other purposes such as breeding, it is not particularly important for the animal to be significantly loyal.

### Life Expectancy

While life expectancy doesn't tend to be the most important factor in animal-client matching, a low weighted matching between life expectancy and client age may tip the scales if in need of a tiebreaker. Ideally, we want to match a client with an animal that can live the entire rest of their lives with that particular client.

| **Non-Physical Attributes** | **Physical Attributes** |
|---|---|
| Aggression<br>Attachment<br>Obedience<br>Energy Level<br>Crate-Trained<br>Good With Children<br>Loudness<br>Intelligence | Animal Type<br>Hypoallergenic<br>Neutered<br>Cost<br>Average Cost Per Year |

| | |
|---|---|
| Cleanliness<br>Playfulness<br>Loyalty<br>Life Expectancy | |

Table 2.1 Animal Attributes

## 2.2 Client Attributes

With the animal attributes out of the way, let us throw some light upon the client attributes and preferences. A client has certain preferences that exactly correspond to attributes of animals; these include preferences for animal type, purpose, intelligence, attachment, good with children, crate trained, hypoallergenic, budget, and many more. Let's discuss the additional attributes and preferences for the client that are important and must be considered while generating an ideal set of matches.

### Patience

How patient a client is determines how well they can deal with disobedient or more messy animals. For instance, if the client's patience level is relatively low, an ideal match would not be with animal that is poor in obedience. This could cause tensions between the two and the client may give back the animal for adoption.

### Home Type

While the type of the home(house, apartment, etc.) doesn't directly map/correlate to any of the animal attributes, it definitely affects (if not by a lot) the characteristics of the ideal animal. For instance, you would not want to match a loud animal to a client that has an apartment. In this case, there is a chance that the noise level of the pet may disturb the neighbors, leading to an unsuccessful match with that client.

### Free Time

Whether the client is busy or not is an important aspect to be considered when deciding on a match. This is because less free time means less time to spend with the animal. If the animal becomes very attached to a client with little time to spend with his pet, this would not lead to a successful match as the pet may lack the attention it needs.

## 2.3 User-Input and Attribute Derivation

All animal attributes and client preferences are easily added and edited by a user using the cuACS GUI, which are stored within the application database. Every attribute and preferences is matched in one way or another with varying importance. Every single client and animal attribute pairing is evaluated out of 100 (with a few exceptions) to assess the compatibility between the specific elements of the profiles; this “scoring” process is discussed in elaborate detail in section 3.

## 3. Scoring Pairings

Now that the attributes of animals and clients have been defined, we can use them to calculate how likely it is that a specific animal-client pairing will be successful, or the estimated happiness of the animal with this client.

The simplest way to get such a pairing score would be to obtain the dot product of the animal attribute vector and the matching human preference vector, such as

$$\begin{bmatrix} type \\ hypoallergenic \\ goodwithchildren \\ neutered \\ aggression \\ attachment \\ cratetrained \\ intelligence \\ loyalty \\ attachment \\ obedience \\ loudness \\ cost \\ energy \\ playfulness \\ cleanliness \\ cratetrained \\ neutered \\ maintenance \\ cleanliness \\ aggression \\ cratetrained \\ cleanliness \\ lifeexpectancy \\ goodwithchildren \end{bmatrix} \cdot \begin{bmatrix} typepreference \\ allergies \\ numberofchildren \\ requestedneutered \\ purpose \\ attachment \\ travel \\ purpose \\ purpose \\ freetime \\ patience \\ hometype \\ budget \\ hometype \\ freetime \\ freetime \\ patience \\ budget \\ salary \\ patience \\ patience \\ budget \\ salary \\ age \\ patience \end{bmatrix}$$

This is a very rudimentary result, and as such, its value is a quite poor indicator of the value of the pairing. Consider for example, that the budget for an animal may be in terms of hundreds of dollars, but that the number of children would be around 2-3. While the importance of the pet getting along with children might be as important as that the pet fits within budget, the budget will be grossly overrepresented in comparison in the resulting score. The solution to this problem is to normalize each value in the attribute vectors to a value between 1-10, a lower value indicating poorer matching likelihood and higher values indicating the opposite. This way, every attribute is given equal consideration when calculating the score.

However, we may want different attributes to be more or less important in the calculation of the final score. For example, we might consider whether or not the client is allergic to the animal much more important to a successful adoption than whether the animal is within his budget. To enable this, we can add a weight vector with which each attribute pair will be multiplied with the corresponding value before all the values are summed. This allows us to give more importance to certain attribute pairs by giving them a higher weight.

Finally, we have noticed that, while there is a direct correlation between the corresponding values in the attribute vectors above, some other attribute pairs may also have a slight impact on the likelihood of a successful adoption. As an example, perhaps good with kids and number of kids are naturally corresponding attributes, but friendliness and number of kids could also be slightly related and have an impact on the score. To handle this possibility, we multiply both the attribute vectors to get an attribute pair matrix, and pair it with an attribute pair weight matrix. Most of the attribute pairs will be unrelated, for example, aggression of the pet and budget of the client, and thus will have weights of zero, but this will allow us to give a score for each of the attribute pairs in case they have an effect on the scoring. Thus we have

$$\begin{bmatrix} type \\ hypoallergenic \\ goodwithchildren \\ neutered \\ aggression \\ attachment \\ cratetrained \\ intelligence \\ loyalty \\ obedience \\ loudness \\ cost \\ energy \\ playfulness \\ cleanliness \\ maintenance \\ lifeexpectancy \end{bmatrix} \times \begin{bmatrix} typepreference \\ allergies \\ numberofchildren \\ requestedneutered \\ purpose \\ attachment \\ travel \\ freetime \\ patience \\ hometype \\ budget \\ salary \\ age \end{bmatrix}$$

For which we multiply each cell of the resulting matrix by the corresponding weight in the following weight matrix.

$$\left[\begin{array}{cccccccccccccc} & typepref & allergies & numhildren & reqneutered & purpose & attachment & travel & freetime & patience & hometype & budget & salary & age \\ type & 225 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ hypoallergenic & 0 & 150 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ goodwithchildren & 0 & 0 & 140 & 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 & 0 \\ neutered & 0 & 0 & 0 & 130 & 0 & 0 & 0 & 0 & 0 & 0 & 48 & 0 & 0 \\ aggression & 0 & 0 & 0 & 0 & 125 & 0 & 0 & 0 & 40 & 0 & 0 & 0 & 0 \\ attachment & 0 & 0 & 0 & 0 & 0 & 120 & 0 & 70 & 0 & 0 & 0 & 0 & 0 \\ cratetrained & 0 & 0 & 0 & 0 & 0 & 0 & 118 & 0 & 50 & 0 & 0 & 0 & 0 \\ intelligence & 0 & 0 & 0 & 0 & 110 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ loyalty & 0 & 0 & 0 & 0 & 80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ obedience & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 63 & 0 & 0 & 0 & 0 \\ loudness & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 58 & 0 & 0 & 0 \\ cost & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ energy & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 55 & 0 & 0 & 0 \\ playfulness & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 50 & 0 & 0 & 0 & 0 & 0 \\ cleanliness & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 53 & 43 & 0 & 0 & 30 & 0 \\ maintenance & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 45 & 0 \\ lifeexpectancy & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 \end{array}\right]$$

We justify the given weights by placing the pairs of attributes in the following order of importance, with higher importance resulting in a higher weight.

**High**

Animal type and client type preference -- 225
Hypoallergenic and client allergies -- 150
Good with children and number of children -- 140
Neutered and client requested neutered -- 130

Aggression and purpose -- 125
Attachment and attachment -- 120
Crate trained and travel -- 118
Intelligence and purpose -- 110

**Medium**
Loyalty and purpose -- 80
Attachment and free time -- 70
Obedience and patience -- 63
Loudness and home type -- 58
Cost and budget -- 57
Energy and home type -- 55
Playfulness and free time -- 50
**Low**
Cleanliness and free time -- 53
Crate trained and patience -- 50
Neutered and budget -- 48
Maintenance cost and salary -- 45
Cleanliness and patience -- 43
Aggression and patience -- 40
Crate trained and budget -- 38
Cleanliness and salary -- 30
Life Expectancy and age -- 25
Good with children and patience -- 20

The rest of the pairs of attributes, for instance, whether an animal is neutered or not and a client's preference for the animal's noise level, bear no importance to scoring the matchings, and therefore have a weight of zero.

# 4. Scoring Matchings

In order to determine if a matching is appropriate, a way of objectively scoring matchings needs to be defined. This is so that matchings can be programmatically compared on the basis of their scores. Comparing matchings allows us to say whether one matching is better than another, thus allowing us to choose between two potential matchings.

As a matching is composed of pairings, the simplest way to score one would be to sum up the scores of all of its pairings. This is a good initial solution as a matching with a high score will likely have many good pairings. However, this solution can result in matchings with a few astronomically good pairings and a few bad ones being favoured over matchings with all good scores. This does not satisfy our problem statement adequately. Thus, certain steps must be taken to favour a preponderance of good matchings.

## 4.1 Good Pairings

We define a pairing as “good” so as long as the score of the pairing is above the standard deviation of the mean pairing score. This means that 15.8% the total pairings will be good, and the other 84.2% will not be. This 84.2% of the pairing will suffer a very large penalty that will essentially remove them as a matching candidate.  Using this pruning method, we can ensure that our final list of animal-client matches will be very strong.

It is important to note that this pruning process promotes moderate matching. Since all of the bad pairings are removed, there will no longer be a chance of having an extremely compatible matching and an extremely compatible matching. All matches will at least be at least a top 15.8% scored pair; every single match will be a strong one.

In the code, this pruning is represented by the preProcess function, which will be executed on all scores after they are determined. This pruning leaves us with a much more digestible set of potential pairings; all of which will be extremely compatible. In the unlikely case that all of an animal’s pairings are pruned in this process, they will be excluded from the final set of matches. In this scenario, all of those pairings will be incompatible and are best left unmatched.

## 4.2 Logarithmic Transformation

Another modification in scoring matchings which would help promote more good matches over a split between extremes. would be to calculate the sum of the logarithms of the scores of their pairings. Applying such a logarithm transformation would penalize matchings with a couple excellent matchings and a couple poor ones while favoring those with many good pairings. This is because the benefits for a high scoring pairing become geometrically smaller as the score gets higher. The difference between a low scoring pairing and a medium scoring one becomes drastically larger than that between a medium score and a high score. This favours more balanced matchings(see Example 5.1) which, from our problem statement, is what we want. Because the logarithm is an increasing function over all of its domain, higher scores are still always favoured over lower ones, only the amount they are favoured by decreases as the scores get higher.

Scoring matchings need not be much more complex than this. This is because an optimal matching is one in which every animal will have the happiest home possible, with the caveat that we prefer having many animals with moderately happy homes than a few with extremely happy homes and a few with unhappy homes. The scores of the pairings tells us how happy each home is likely to be, whereas the logarithm transformation addresses the caveat.

Example 5.1 A demonstration of the effectiveness of a logarithm transformation

We have 2 animals $A_1$ and $A_2$, and 2 clients $C_1$ and $C_2$. To obtain all possible pairings and their score, we can perform a matrix multiplication with both vectors, defining addition as the scoring of a pairing, expressed as the function S. Thus we have

$$\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \times \begin{bmatrix} C_1 & C_2 \end{bmatrix} = \begin{bmatrix} S(A_1, C_1) & S(A_1, C_2) \\ S(A_2, C_1) & S(A_2, C_2) \end{bmatrix}$$

Which, for example, could have the the matching scores depicted as the matrix

$$\begin{bmatrix} 9 & 5 \\ 4 & 1 \end{bmatrix}$$

The optimal pairing at this point would be $\{A_1C_1, A_2C_2\}$ with a score of 10, as the other possibility $\{A_1C_2, A_2C_1\}$ has a lower score of 9. From the problem statement, we would prefer the more even matching instead. Transforming the matrix by applying the natural logarithm to all of its values will result in the matrix

$$\begin{bmatrix} 2.2 & 1.6 \\ 1.4 & 0.0 \end{bmatrix}$$

for which the second, more moderate matching would be favoured over the first with a score of 3 over 2.2.

# 5. Optimization Algorithms

Once we have a method of scoring individual pairings and matchings, the task of determining a favourable matching comes down to an optimization problem, with the goal being to optimize the total score of a matching. It is not a trivial task to determine the best algorithm for an optimization problem. For similar problems, such as the travelling salesman problem, getting the optimal solution is a problem in the NP-Hard complexity class. For this reason, a short discussion of the possible algorithms is in effect. In the discussing the time complexity of the various algorithms, we use the greater of the number of animals and the number of clients and denote this number by *n*. To avoid an excessively long discussion, we will present the time complexity of the discussed algorithms without proof, except in discussing our chosen algorithm for which we will present a short derivation of the time complexity. The other proofs are left as an exercise to the reader.

## 5.1 Simple Greedy Algorithm

A greedy algorithm is one of the first that comes to mind in an optimization problem. We would start by finding the best of all possible pairings. Once this is done, we would remove the winning client and animals from their sets and mark down that pairing. We would then repeat this process with the remaining animals and clients, again and again, until we have a list of all the pairings. This algorithm has a time complexity of O($n^3$). The weakness of this algorithm (and the reason we have chosen not to use it) is that the last few pairings generated tend to be of poor quality(See Figure 5.1). This is unaffected by our logarithm transformation, as the pairing with the maximum score is the same one before and after the transformation. Only the difference of value between it and the second highest scored pairing will change. This results in a matching composed of a few excellent matches and a few poor ones, which is something we do not want.

$$\begin{bmatrix} \enclose{circle}{100} & 88 & 45 \\ 99 & \enclose{circle}{2} & 23 \\ 64 & 16 & \enclose{circle}{24} \end{bmatrix}$$

Figure 5.1 A greedy algorithm resulting in a poor final pairing (2nd row)

## 5.2 Hill-Climbing Algorithm

Since the simple greedy algorithm does not satisfy our needs, a more complex greedy algorithm can be tried. The algorithm we are referring to is a hill-climbing algorithm. The basic idea is to take a given matching and perform incremental operations which raise the matching's score step by step. This is done by looking at all possible animal trades between clients, and retaining the trade which increases the score by the highest amount (for example in {$A_1C_2$, $A_2C_1$} $C_1$ and $C_2$ could trade animals so that the solution becomes {$A_2C_2$, $A_1C_1$}). This step is then repeated until all trades remaining decrease the matching's score.

This is an improvement over the simple greedy algorithm, especially if the given matching is the result of the simple greedy algorithm. The reason is that it can make trades between the low-scored pairings and high-scored pairings to generate a more balanced matching, which will likely have a higher score because of the logarithm transformation. Another advantage of this algorithm is that, as it makes incremental steps in improving the matching, the algorithm can be terminated at any step and still generate a reasonable matching, although the longer it is left to run, the higher the resulting matching's score will be.

The main disadvantage with this algorithm is that, while it does find local maxima for the matching's score, if the problem space is non-convex (has multiple local maxima) then the best possible outcome may not be reached(See Example 5.1). Another disadvantage is that, if the algorithm reaches a plateau state (where no immediate trade increases or decreases the matching's score) then the algorithm can have a time complexity of O($\infty$) as the algorithm walks randomly around the plateau.

Example 5.1 A minimal counterexample to the concaveness of the optimal matching optimization problem-space

Suppose we have the following pairing score table, and the state of the algorithm is represented by the circled numbers

$$\begin{bmatrix} \textcircled{9} & 0 & 6 \\ 5 & \textcircled{2} & 0 \\ 3 & 4 & \textcircled{3} \end{bmatrix}$$

then there is no immediate trade that the algorithm can make to increase the score. It has found a local maxima with a score of 14. This is not the optimal solution however, as we can see that the state

$$\begin{bmatrix} 9 & 0 & \textcircled{6} \\ \textcircled{5} & 2 & 0 \\ 3 & \textcircled{4} & 3 \end{bmatrix}$$

results in a higher matching score of 15. We can thus see that the problem-space is non-convex.

## 5.3 Naive Combinatorial Approach

Given that greedy algorithms are heuristics for approximating the optimal solution which gives poor results, the next algorithms to consider are ones which produce the optimal result. This can be acheived by using a combinatorial approach, comparing all possible matchings and picking the best one.

The simple way to accomplish this in this case would be to generate all possible matchings (which can be done iteratively via a Johnson and Trotter algorithm[1]) and pick the matching with the highest score. The downfall of this approach however, is its time complexity. As the number of possible matchings is a permutation problem, the time complexity is factorial, or O(*n*!). In fact, the number of permutations for matching 25 animals to 20 clients is near $1.3 \times 10^{23}$, which is an untenable number to check. Even if each check took a single floating point operation and the program was run by a supercomputer running at 122.3 petaFLOPS (a june 2018 record[2]), it would take almost 294 hours to run. Thus, this algorithm is unfit for our purposes.

## 5.4 Dynamic Combinatorial Approach

A demonstrably faster way to check all possible matchings is to use a dynamic approach. To do this, we must decompose the problem into various subproblems for which we can store the solution. Let us consider the set of animals **A** = $\{A_1, A_2 \dots A_n\}$ and the set of clients **C** = $\{C_1, C_2 \dots C_n\}$. Then, let us denominate the value for the highest-scored matching in these sets as highest(**A**, **C**), and the specific pairing score for $A_x$ and $C_y$ by score($A_x$, $C_y$). We recognize that

[1] Johnson, Selmer M. (1963), "Generation of permutations by adjacent transposition", *Mathematics of Computation*, **17**: 282–285
[2] Top 500, WEB:https://www.top500.org/lists/2018/06/ accessed 2019, March 2nd

highest(**A**, **C**) = max{score($A_x$, $C_y$) + highest(**A**-{$A_x$}, **C**-{$C_y$}) where $A_x \in$ **A** and $B_y \in$ **B**}.

Thus, the problem of finding the highest-scored matching can be decomposed into the smaller subproblems of finding the highest-scored matching for subsets of the animal and client sets. This could be implemented with a recursive function. In a recursive implementation however, many of the subsets tested would be tested in multiple other recursive calls. This is where the dynamic approach can be of assistance. If we generate all subsets in a bottom up approach (first generating the smaller subsets and using their score to generate the larger ones), then we can save many redundant computations.

Practically, this will be done by generating all the *highest* values for every 1-client subset in the first row of animals(for a single animal). These values will then be used to generate the *highest* values for all the 2-client subsets using the first 2 rows of animals. This process will be used for every following row until the *highest* value is obtained for the entire set of clients and animals. Using this method, the time complexity of the algorithm is given by the number of subsets we test times the number of operations per subset. Let $Sub_m k$ be the number of subsets of size m in a set of size k, and Sub(k) be the number of all possible subsets for a set of size k, then the number of subsets we must test is given by

$$Sub_1 n + Sub_2 n + \dots Sub_n n = Sub(n) - Sub(0) = 2^n - 1$$

where n is the number of clients or animals if their respective sets are of the same size (if they are of different sizes, than, n being the greater of the two sizes, the number of subsets will be smaller than $2^n - 1$). The number of operations per subset is given by the number of animals (or clients) in a subset, which is bounded by the total number of animals. Thus, the time complexity of this algorithm is $O(n2^n)$. With 25 animals and clients, this results in 33,554,431 subsets to check, which is many less than the naive approach. One of our teammate's second-hand laptop with 16,000,000,000 FLOPS could check all of them in a fraction of a second. The problem with this is that this does not hold for higher numbers of clients and animals. For double the clients/animals, the number of subsets to check becomes a staggering 1,130,000,000,000,000, which would take about 1000 hours to check on the same laptop.

## 5.5 Dynamic Combinatorial Approach Using Heuristics

To counteract this problem, a heuristic approach is used to drastically reduce the number of subsets to be checked. The heuristic that seems most appropriate is to subdivide the clients and animals according to their prefered animal type and their animal type respectively. As such, pairings will only be considered if these two types match. We can then group the clients with their prefered animal type and obtain the optimal matching for each subgroup by using the dynamic combinatorial algorithm. Combining each of the optimal matchings will give the larger optimal matching, as we can assume that any crossover between the groups (a client matched with an animal type he does not want) will severely diminish the score of any matching.

Consider our previous example with 25 animals and 25 clients. For five animal types (and assuming client preferences are evenly spread), we get 5 subgroups of five clients and animals each, and we will get only 31 subsets to compute per subgroup. With five subgroups, this gives us only 155 total subsets to compare. This is many less than the number of subsets to calculate without using heuristics (see section 5.4). Another advantage of this heuristic is that this allows us to omit the the animal type value from both beings when generating the pairing score, thus reducing the number of operations per pairing.

This scales very well up to approximately 170 clients and animals, where computations would start taking more than a second on the aforementioned laptop (and this is still assuming the improbable single floating point operation per subset, although for each doubling of the number of floating point operations needed, the number of acceptable clients/animals only goes down by one).

While the time complexity for this method is the same as the above one, $O(n2^n)$, the factor is much smaller, and this algorithm can handle any inputs within the scope of typical operations for this animal shelter, which is known to operate around 25 animals and clients. We believe the likelihood of the shelter more than sextupling its capacity is low enough to warrant the use of the algorithm. Should there be such an unforeseen event, then perhaps replacing this algorithm with a sophisticated hill-climbing algorithm would be the way to go. Until then, this algorithm should be more than satisfactory, with the advantage of giving the best possible matching rather than its approximation.

## 6. Algorithm Overview

The algorithm begins by generating a table containing every possible pairing score. These scores are obtained by taking both the animal attribute and client attribute vectors, normalizing their values to a scale of 1-10, and taking their cartesian product. A weight dependent on the importance of the attribute pair is multiplied with each element of the corresponding matrix. The sum of every element of the matrix is then obtained and a logarithmic function is applied to this sum, which produces the pairing's score. The value of a good pairing is calculated, and all non-good pairings are given a heavy penalty. The animals and clients are then separated into subgroups, grouping clients with the animal type they want to adopt. For each of these subgroups, an animal is selected. The pairing score for this animal and every client of the same subgroup is obtained, representing the highest value for the subsets of size one. Then, the highest value for each subset of size two in using the second animal and all the clients is obtained. This is done by getting the maximum value between each pairing value added to the highest value of their corresponding subset. Once this is done, the smaller subsets can be discarded. The steps can then be repeated until the highest value for the entire subgroup has been calculated. Having kept track of the pairings present in each maximum value for subsets, we now have the optimal matching for the subgroup. This matching is combined to those returned by the other subgroups to return the final optimal matching.

## 7. Glossary

**Pairing**

A client and an animal set to go home together.

**Pairing score**

A value indicating how likely the pairing is to be successful (i.e. the animal and client will be happy together). See section 3 for an in-depth  explanation.

**Pairing score table**

A table of scores for every possible pairing. Each row represents an animal and each column a client. It is a representation of the pairing scoring function being applied to the cartesian product of the animal and client vectors.

**Matching**

A set of pairings in which every client and animal appears at most once, and for the group with the lesser number of beings, all beings appear exactly once. All possible matchings compose the domain within which the possible solutions to the problem statement exist.

**Matching score**

A value used to compare matchings, based on the scores of its respective pairings. The matching score reflects how well a matching satisfies the problem statement. See section 4 for an in-depth  explanation.

**Good Pairing**

A good pairing is one which is at least one standard deviation above the mean of all possible pairings