CUACS

Carleton University Animal Care System

ACM Design Document

Team 777

By:
Foyin Ogbara

COMP3004 Winter 2019
Prof. Christine Laurendeau
School of Computer Science
Carleton University

Apr 1th 2019

Contents

Figures

Tables

1. Introduction

1.1 Purpose

The system is designed to pair up clients with their individual ideal animal from a pool of different species. To do this a reliable algorithm and proper implementation is required. The purpose of this document is to describe in detail all aspects of the Animal - Client Matching (ACM) algorithm developed. An algorithm designed to find the ideal animal - client pair based on client preferences and abilities and animal attributes.

1.2 Overview of Document

This document details the algorithm the system uses to match clients with appropriate animals. Also describes the criteria on which the the actors are assessed. Here the preferences and information of the client are outlined as well as the attributes of the animals which the clients information is compared against. Also how client and animal information is stored and processed. The fact that OKcupid is designed to find as many close matches as possible, but this system is designed to find the closest match or no match is taken into account. That means this algorithm returns a either one stable match or noe for each client animal pair.

2. Proposed System

2.1 Overview

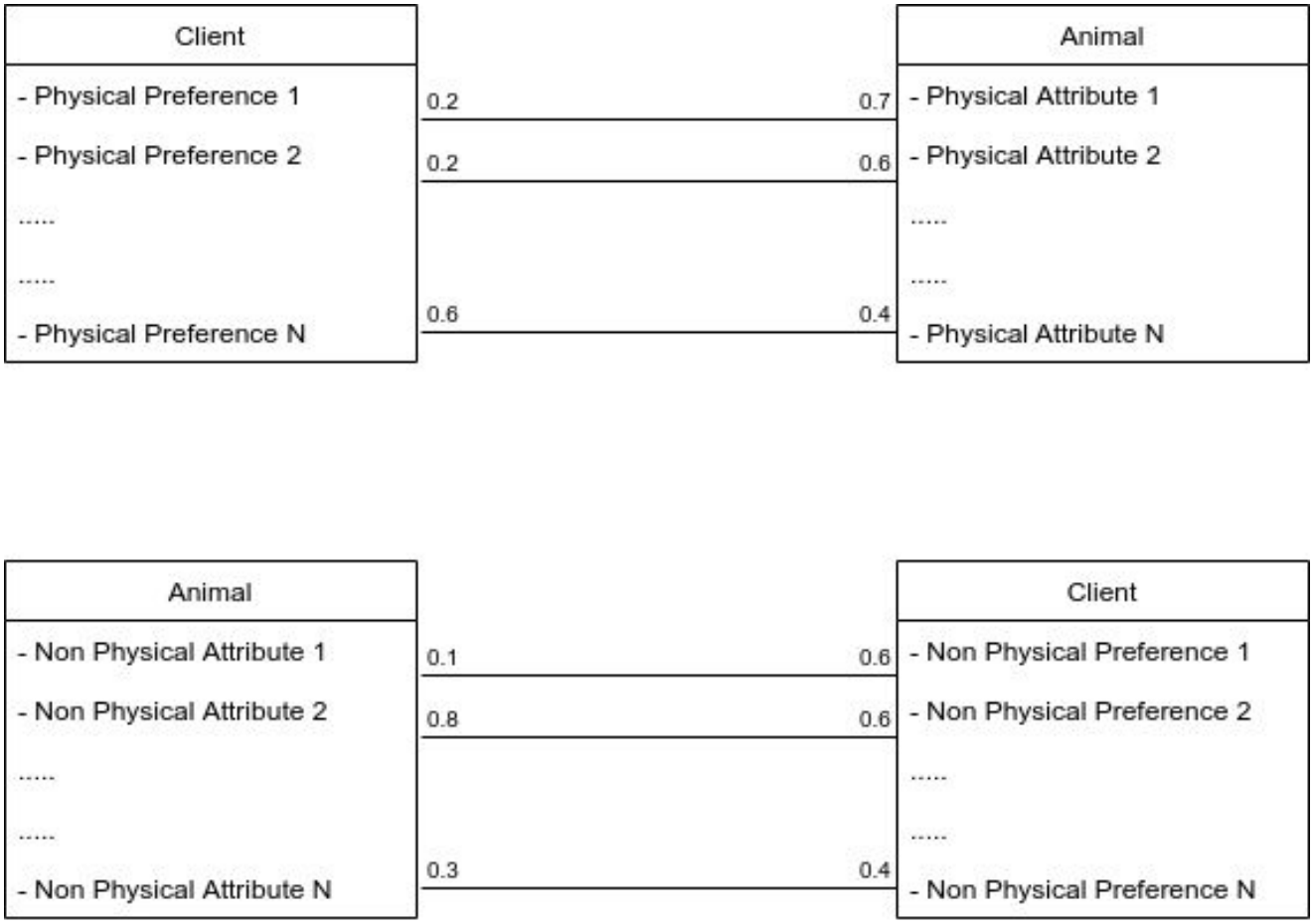
The algorithm is split into two parts, preference ranking and stable pairing

2.1.1 Preference Ranking

Where client physical preferences are compared to animal physical attributes and a list of animals is associated to each client. This list is a sorted by how much a client likes each animal (that means that the animals physical attributes match what the clients are looking for). The same thing is done with the animals non - physical attributes but in this case there is a list of clients is associated to each animal (that means that the animals non - physical attributes or needs match what the clients can offer).

> Preference Ranking Algorithm : Each animal attribute has weights from 0 to 1 (except specified) and the user enters information on their account, which has similar information about them, detailing how important each attribute is to them. This is done by the user entering weights of their own associated with their accounts for each attribute. We then subtract the user input weight from the animal attributes weights for each pair of corresponding weights and return the absolute value. After that each result is added together and the closer the result is to zero (for each client and animal pair) the better of a match they are. So a perfect match would be a client animal pair where their result is zero. If client or animal has rank of of zero for multiple animals or clients then list them in any order for each client or animal.

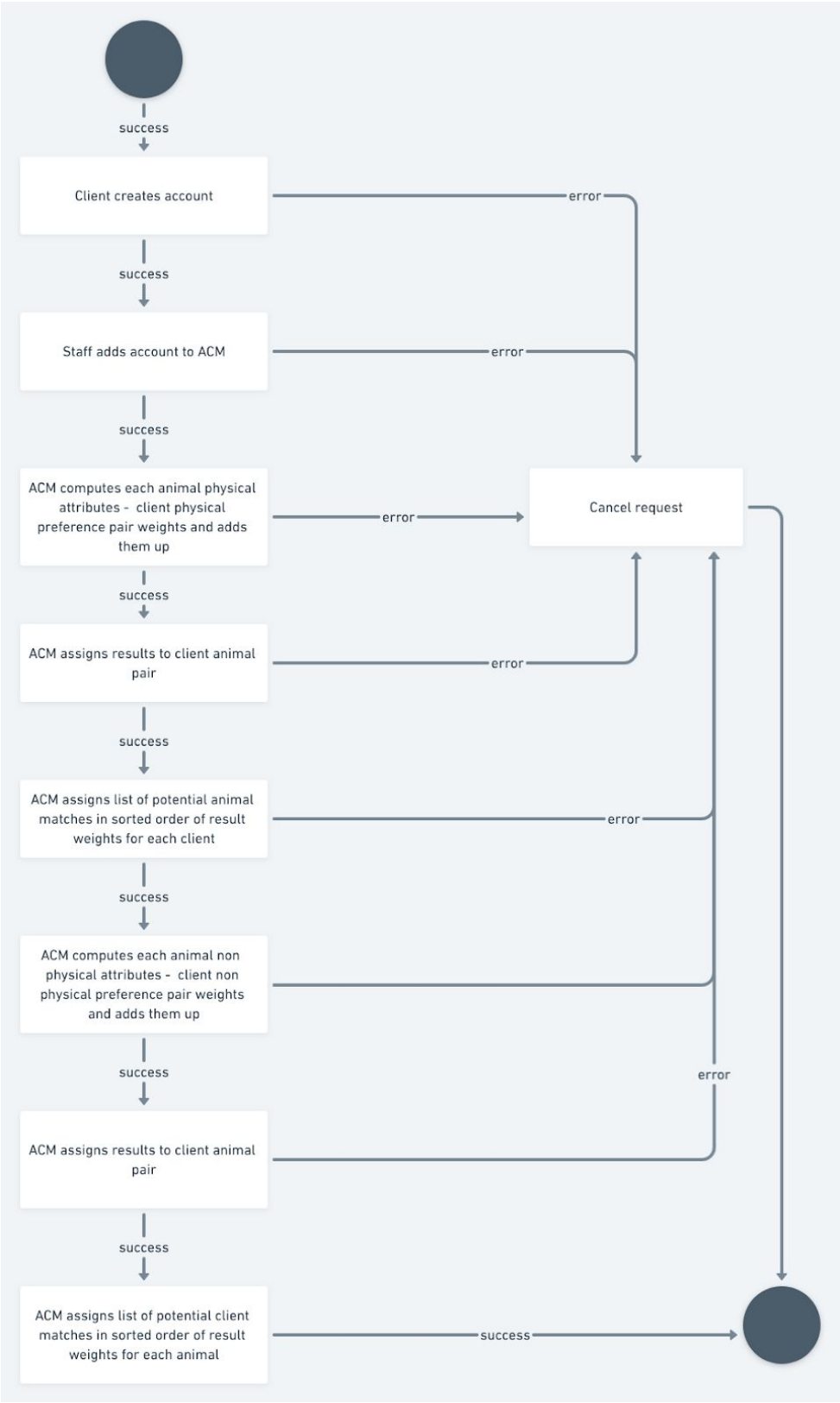
Figure 2.1.1.1 Getting Pair Weights



Pair Weight $= \sum_{N=1}^{N}$ abs(Preference N - Attribute N) for each animal client pair

Then each client has a list of animals sorted by weight associated to them based on non-physical attributes and then each animal has a list of clients sorted by weight associated to them based on non-physical attributes.

Figure 2.1.1.2 - Preference Ranking State Machine Diagram



2.1.2 Stable Pairing

This is where each client and animal preference list information is used to create stable pairs of each animal and client, where each animal is matched with one ideal human or none. This part has two steps:

1. Each animal is initially paired with the client at the top of their list, but the client rejects all animals except for the one that is on that top of his list. This would create a tentative pair.
2. Each animal that was rejected then pairs up with their next best match. And again each client rejects all pairs except for the animal at the top of his list.

Example:

After preference ranking has been done for clients or animals

Table 2.1.2.1 - Clients

Jane	Paul	Max	Sam	Carl	Bob	John	Steve
Dog	Dog	Bird	Cat	Fish	Mouse	Cat	Bird
Cat	Fish	Snake	Horse	Dog	Horse	Fish	Snake
Fish	Mouse	Mouse	Mouse	Snake	Snake	Snake	Mouse
Bird	Horse	Cat	Bird	Cat	Fish	Horse	Fish
Snake	Snake	Dog	Fish	Horse	Dog	Bird	Horse
Mouse	Cat	Fish	Dog	Mouse	Bird	Dog	Cat
Horse	Bird	Horse	Snake	Bird	Cat	Mouse	Dog

Table 2.1.2.2 - Animals

Dog	Cat	Fish	Snake	Horse	Mouse	Bird
Jane	Max	Paul	Bob	Paul	Carl	Jane
Paul	Paul	Jane	Max	Bob	Bob	John
Max	Jane	Sam	Steve	Carl	Steve	Max
Sam	Bob	Steve	Carl	Steve	Paul	Sam
Carl	Steve	Carl	John	Sam	Max	Carl
Bob	Sam	Bob	Jane	John	Sam	Bob
Steve	Johnl	Max	Sam	Jane	Jane	Steve
John	Carl	John	Paul	Max	John	Paul

Table 2.1.2.3 - On first iteration:

Client	Possible matches	Top choice	Current Match	Tentative Match	Tentative match choice level
Jane	Dog, Bird	Dog		Dog	1st
Paul	Fish, Horse	Fish		Fish	1st
Max	Cat	Bird		Cat	4th
Sam		Cat			
Carl	Mouse	Fish		Mouse	6th
Bob	Snake	Mouse		Snake	3rd
John		Cat			
Steve		Bird			

Table 2.1.2.4 - On Second iteration:

Rejected animals go for next best choice and clients accepts highest ranked animal

Client	Possible matches	Top choice	Current Match	Tentative Match	Tentative match choice level
Jane		Dog	Dog	Dog	
Paul		Fish	Fish	Fish	
Max		Bird	Cat	Cat	
Sam		Cat			
Carl		Fish	Mouse	Mouse	
Bob	Horse	Mouse	Snake	Horse	2nd
John	Bird	Cat		Bird	5th
Steve		Bird			

Table 2.1.2.5 - On Third iteration:

If client has an offer from a higher ranked animal he would pair with it and leave whoever he/she is currently paired with

Client	Possible matches	Top choice	Current Match	Tentative Match	Tentative match choice level
Jane		Dog	Dog	Dog	
Paul		Fish	Fish	Fish	
Max		Bird	Cat	Cat	
Sam		Cat			
Carl		Fish	Mouse	Mouse	
Bob		Mouse	Horse	Horse	
John		Cat	Bird	Bird	
Steve	Snake	Bird		Snake	2nd

Table 2.1.2.6 - Stable Pairs

Client	Animal
Jane	Dog
Paul	Fish
Max	Cat
Sam	
Carl	Mouse
Bob	Horse
John	Bird
Steve	Snake

Sam has no match, most likely because there are less animals than clients

2.2 Attributes and Preferences

The client has an attribute called preferences with sub attributes which are the same as the animal requirements. The client preferences sub attributes (which client enters) are compared with animal attributes.

2.2.1 Physical Attributes:

- Age
    - Age will be relative to the kind of animal, but will probably fall into very young (puppy / kitten age), young, middle aged, old
    - Very young, young, middle aged, old.
- Gender
    - Male or Female, match returns 0, and mismatch returns 1
- Species
    - Classified into the following groups, mammals, Birds, Reptiles, Amphibian, insects, fish
- Colour
    - RGB values mapped to 0 - 1 after difference between each client colour value preference and animal attribute colour value is computed
- Fur Level
    - This determines how much furry a pet is. More furry pet requires more brushing, also a more furry pet also sheds more. Reptiles have zero fur level.

2.2.2 Non - Physical Attributes:

All attributes have a range of values from 0 - 1 unless specified

- Activity
    - How active is the animal? Is it low activity (such as spends most of the time sleeping, or sitting idle, ex: snake, a fish). Is it medium activity, where it may require a bit of effort to maintain / control (such as an older cat), or is it very active and requires a lot of effort to maintain / control (like a Puppy).
- Space required
    - Does the animal require a lot of space? Some animals may require very little space (such as a fish, or a bird), does it require a moderate amount of space (such as an entire room, or multiple rooms, like for example a horse), or does it require a lot of space, (such as an outdoor cat/dog or a horse).
- Noise
    - Is the animal loud? Some animals may be very quiet (such as a fish, snake), some animals may be moderately loud (some dogs or cats), and other may be very loud (monkey)
- Training Difficulty
    - Some animals may require different levels of training. Some animals may require 0 training, some animals may require a little amount of training, some animals may require moderate of training, and some animals may require a lot of training and be difficulty to train. Like a fox may need more training than a cat

- Rescued / Bred
    - Some animals in the shelter may be bred or rescued. This will just be preference.
    - Rescued, bred. Return 0 if pair match and one if they don't.
- Maintenance
    - How difficult to maintain. Does it have a specific diet or routine that makes it more difficult to maintain. Maybe daily walks or specific temperature requirements
- Risk
    - Some animals may have some risk associated with them. This can be problematic if you have young children. Animals may have 0 risk, low risk, medium risk, or high risk, e.g. snakes, scorpions
- Cleanliness
    - Some animals may be very messy, some may be moderately messy, some a little messy, and some clean.
    - Clean, little messy, moderately messy, very messy.
- Social
    - Some animals are very social and require a lot of social contact. Some animals require a lot of social interaction so you may need to get more than one ideally
    - No social contact, little social contact, moderate social contact, a lot of social contact.
- Independence
    - Some animals require constant attention while others don't. E.g. an ant farm with a good ecosystem may not require as much attention as a dog that needs to be walked every day
- Rarity
    - How rare / unique is the animal, some people prefer rare exotic animals or species
- Personality
    - This details the types of personality the animal has. For example this could determine whether you want a dog for protection or leisure
    - The options are, timid, aggressive, sensitive, friendly, energetic. The user can enter more than one value and the animal could have more than one personality trait. Each trait of client preference and animal attribute would carry 2 points with both for a maximum of ten. When a matching trait is found subtract points from client and animal. e.g. client choose timid and friendly – 4 points, animal is timid and sensitive – 4 points. Subtract matching points (timid – 2) abs(2 – 4) = 2 points, close match but not perfect

3 Algorithm Philosophy

The idea is to have the client be able to choose generally what kind of animal they want, but they also have to give information on whether they have the ability to take care of the animal as well as information on what kind of animal their lifestyle and personality are a good fit for (depicted by non physical requirements).

The second part of the algorithm pairs a specific animal with a specific client based on the information provided. It is based on the stable marriage problem, where a group of men and women have a list of the prefered potential spouses of the opposite sex. One group (men / women) make proposals to the other group and the other group makes certain decisions. The group that makes the proposals is always guaranteed to get their best choice. And the choice of people being proposed to worsens as the laer the proposal happens. So in this system the animals make the proposals and they get their best possible match. Since the client's ability to take care of the animal is more important that what the client wants physically, the animal match is prioritised (so if you really want a blue fish but aren't able to take care of it we won't let you adopt it). This algorithm is used in the National Resident Matching Program where residents submit a preference list and this and other information is used to match them with hospitals.

3.1 Stable Pairing Proof

To prove that the animal gets his best possible match we can use proof by contradiction. So we first assume there is an animal the that doesn't get their best match. This also means that there is an client that exists that is a better match for a certain animal than who the animal is currently matched with, so we have to show a contradiction in this theory. So we look at the first animal rejected by a possible client show that they can't exist.

A - first animal rejected by possible client
A' - other animal, better match for client
$C_{algorithm}$ - client assigned to animal by algorithm
C_{better} - better client match for animal than current animal match
C^{I}_{better} - better than C_{better}

Premise - Assume C_{better} rejects A
If C_{better} exists then A would attempt to pair
If Premise is true then A' exists and attempts to pair with C_{better}
A' can only attempt to pair with C_{better} if C^{I}_{better} rejects A'
CONTRADICTION

This shows that for the first animal rejected by a possible client to exist there must have been an infinite number of rejections of animals before the first rejection.

3.2 Run time

The entire algorithm has a runtime of:

O(n + nlogn + n^2)

O(n) - time it takes to compute pair weights on average
O(nlogn) - time it takes to sort preference lists (using merge sort ideally)
O(n^2) - stable pairing runtime

This may not be the fastest algorithm, but for this application the time complexity is not that bad. As long as it is run at either the beginning or the end of the work day the system would work seamlessly with the business. Especially since other administrative tasks may take just as long. The clients could give their information at the shelter and be notified of their match the next day. This set up also makes the system more scalable and since the most information is stored as floating point numbers instead of strings or larger data types, the system saves some space. So there are balanced trade offs.

4 Glossary:

Term	Description
ACM	The 'Animal-Client Matching' algorithm which will match animals and prospective clients
Animal	An animal in the shelter that is placed in the cuACS to be adopted.
cuACS	cuACS: The Carleton University Animal Care System.
Client	An individual who is placed in the cuACS with purpose of adopting an animal.
Staff	Individuals who work at the shelter and will manage the adoption process using cuACS.
Preference Ranks	An ordered list of client or animal preferences
Tentative Match	Potential / temporary animal pair match in ACM
Stable Pairing	Making a ideal match pairs out of animal and client preference rank list
Pair Weight:	How closely matched and animal - client pair is based on score before stable