MEETPLANNER

DESIGN DOCUMENT

●IDENTIFICATION

Project Name: MeetPlanner

Project Manager: Peter Grabowski

●

●OVERVIEW

Swim coaches are often faced with a dilemma while planning swim meets. On the one hand, they want to maximize their chances of winning the meet, but they also want to ensure that all swimmers have an opportunity to compete. Trying to find the perfect balance between the two can be tedious and time-consuming ®C many coaches spend as much as twenty hours per week writing meet sheets, on top of their normal coaching schedules.

Swimming is an inherently numeric sport. Each swimmer's performance can be distilled into their times for each event. Similarly, meet planning can be simplified into an algorithmic attempt to maximize a team's score while satisfying certain constraints. However, the software packages currently in wide use (namely HyTek's TeamManager) offer little to no algorithmic assistance to coaches for planning meets.

MeetPlanner will make the meet planning process much simpler by supplementing the functionality offered by TeamManager. It will eliminate the repetitive aspects of planning a swim meet, while still allowing coaches to retain fine-tuned control over their meet sheets. Given a team's roster, the coach's preferred strategy, and (optional) information about the opposing team's swimmers, MeetPlanner will create an optimized, customized meet sheet, in a format byte-compatible with TeamManager. This will help coaches plan meets in significantly less time ®C in some cases, up to fifteen times faster ®C allowing coaches to spend less time planning and more time coaching.

Functionality

Registration

Coaches will begin by creating an account tied with their team. Next, they'll upload their TeamManager database and set certain parameters about their league ®C which events are available, whether it's a high school, college, or age-group league, etc. For each set of parameters, the coach will be able to select from a list of detailed options or specify their own custom settings. Registration will only occur once, although the initial parameters set will be modifiable.

Meet Planning

Coaches will begin by setting certain parameters unique to each meet ®C whether the pool is meters or yards, short course or long course, how many lanes are in the pool, how many teams will be competing in the meet, etc. Next, coaches will be given the opportunity to make sure their team roster is up to date. Coaches will then specify the strategy they want to use ®C if it's a friendly meet, they can maximize the number of events each member of the team is entered in; if it's a competitive meet, every point counts (or perhaps some strategy in between). Coaches will also be given the opportunity to input any entries they are sure of ®C whether a certain swimmer will definitely (or definitely not) swim a particular event.

Coaches will then input any information they have about the other team's roster. This allows our algorithm to tweak its optimal meet sheet based on predictions of the other team's performance. The program will work fine without any information about the other team, but performance will increase when more information is provided. The optimization algorithm will be able to account for any number of opposing teams, whether it's a dual meet or league championships.

Refining the Meet Sheet

The site will then suggest an optimal meet sheet to the coach. The coach will be able to tweak the meet sheet within the website. While refining the meet sheet, the coach will be presented with an intuitive, drag-and-drop interface, visually based on a pool with lanes. The coach will also be given to the option to sort remaining eligible swimmers in a variety of ways and a representation of how the coach's modifications will affect the predicted final score.

For example, if the coach were planning a very competitive meet, s/he might choose to sort swimmers eligible for a given event by their times for that event, fastest swimmer first. Similarly, if a coach were planning a less competitive meet, they might want to make sure that each swimmer has the same number of events or give swimmers a chance to try new events. In this situation, the coach could sort remaining eligible swimmers by their total number of events in the meet, or the last time they swam that event in a previous meet.

Delivery of Results

After the coach is satisfied, the site will output the meet sheet in a format that is byte compatible with MeetManager, to facilitate sharing with other teams. The coach will also have the option to export the meet sheet as a .pdf, making printing easy. Overall, we expect that the meet planning process will take something on the order of a half an hour ®C a sharp reduction from manual planning, which might take upwards of fifteen hours per meet.

Design

Basics

Our web interface will be implemented in Python and Django, and the database will be implemented with MySQL. We are considering using GitHub or SVN for version control, but are currently using DropBox as a simple starting point. Initially, we plan to work with everything hosted locally with Apache, but are willing to consider renting a server from a hosting company.

Front End

We plan on using Django as a framework. We will begin with a relatively simple drop-down menu interface for interacting with the meet sheet, but will aim for a more sophisticated method as time allows. In an ideal interface, for example, coaches would be able to drag swimmers in and out of events, tweaking the meet sheet in real time. This will require that our UI handle things rapidly, so JavaScript/Ajax may be a good choice.

Database

We plan one using MySQL as our database, but Django should insulate us from that decision to a large degree. Each coach will have a unique id (probably in a *Coaches* table). Each swimmer will be linked to the ID of their coach, as well as the ID of the coach that entered data on the particular entry. This will generally refer to the same coach, but they may be different when two coaches using MeetPlanner enter data on each other's teams (to prepare for a meet against one another). Swimmers' times will be modeled as a Kalman filter, which will allow us to abstract expected times as a normal distribution for each event and each swimmer as a mean and variance for each event. For each swimmer, we will store the gender and birthday, as historical data on that swimmer, as well as an up-to-date prediction for each event the swimmer performs in (the mean and deviation output by the Kalman filter). This data will all be entered by the coach, or converted from an uploaded TeamManager database (which is just a Microsoft Access database).

Processing

We will have two general approaches to optimization: strategies for one opposing team, and a strategy for N-opposing teams. For the one-team optimization, we will distribute talent intelligently, putting swimmers in the top X percentile of a given event into that event, and assign less talented swimmers to events that they are experienced with competing. Optionally, coaches may include data on the opposing team, in which our resulting strategy will be more finely tuned to a given meet. For N-Teams, we will initially use a combinatoric brute force method, and will consider more advanced methods such as min-flow/max-cut if better methods are deemed necessary. If data is provided, we will first run our one-team optimization on the opposing team to guess their lineup, and will then run a two-team optimization against that lineup to generate the best meet sheet for our team. If we don't have the other team's data, we will just use the one-team optimization for our team to generate the meet sheet.

Each event will be stored as a tuple, containing the distance and stroke, age of the swimmers eligible for the event, and gender of the swimmer, like ('50 free', '12-14', 'male'). Each lineup will be stored as a dictionary with the key as a tuple like the above, with a value as a tuple of unique keys for the simmers in the event. A meet can be fully represented with two lineups (with initially blank values for each event).

We will initially generate a lineup using the above method, return it using the above representation, and the interface will display the results to the coach. The coach can then make changes in the interface, and

the lineup can be reevaluated to see how the predicted odds of winning have changed (if this is a meet where we have data on the opposing team). The coach can also change some parameters for the meet and regenerate a new lineup.

●MILESTONES

- •Set up version control ®C end of Spring break
- •Get a working MySQL setup going ®C by 3/30/12
- •Implement Kalman Filter ®C by 4/6/12
- •Implement Swimmer class ®C by 4/6/12
- •Implement Single-Team Optimization strategy ®C by 4/16/12
- •Implement Two-Team Optimization strategy ®C by 4/16/12
- •Get above working in Django ®C by 4/20/12
- •Django views for account creation, etc. ®C by 4/20/12
- •Build & expand frontend for optimization ®C by 4/27/12
- •Develop reach features ®C remaining time

Export

Import

Swimmer pages/accounts

- •Come up with a catchier name ®C as soon as possible

●REACH FEATURES

•A web interface for team members. Here, they might be able to indicate how much they are inclined to compete in certain upcoming events, and maybe to view and set their past times. A quick way of doing the former would be to let the coach make a random link he can send to his team, and they can fill it out in a way very similar to Doodle. Or, if time permits, we can make a full implementation with actual swimmer accounts

•Possible analytics to be run once we have a lot of data

•Optimization for 3-team or more meets

•Generalize to other time-based sports

•Include option to score diving

•Include relays and appropriate meet-sheet optimizations

● RISKS AND OPEN ISSUES

- Some group members have limited experience with Python
- Only one member has experience with Django
- Limited experience with web development
- Compatibility with HyTek software could be more complicated than we are anticipating
- Depending on the speed of our algorithms, we may not be able to allow for the user experience we are aiming to create
- Potential browser compatibility problems