The Speech

1 Ladies and gentlemen, I would like to welcome you all for attending my thesis presentation. I am Xiao Yang from Health Informatics, Computer Science. My supervisor is Jesse Hoey. The topic today is Quick and Automatic Selection of POMDP Implementations on Mobile Platform Based on Battery Consumption Estimation. When I was thinking about the title, I just wanted to include all keywords. Didn’t expect it to be such a long one. But after I created this title, it’s hard to cut it down. Because all these “selection”, “POMDP implementations”, “mobile”, “battery”, and “estimation”, as you will see in my presentation, are really important and do constantly occur.

2 My presentation is organized as follows. First talk about what problem we are facing, and then about our special solution. After that, we are gonna go through the experiment part and also the evaluation. In the end, wrap up the presentation and point out some future plan.

3 POMDP is a keyword in our work, partially observable markov decision process. To refresh our knowledge of POMDP, I wanna first talk about MDP, markov decision process. In MDP’s vision of the world, an agent is making sequential decision on discrete time. The point of MDP is to facilitate a better decision making given the surrounding environment. The world is captured by a tuple listed here. We have states, and in every state, the agent can perform an action, which move the agent to some next state with a probability and obtain some rewards accordingly. We want to maximize the long term rewards by performing an action on a state not just for the immediate reward and but also the potential rewards in the future. MDP will compute an optimal policy that maps state to action, serving as a decision plan for the agent.

Partially observable MDP extends from MDP and involves the concept of observations. The external environment is not directly observable but inferrable from the surroundings. For example, the navigation of a robot inside a room, the robot’s vision is blocked by furniture. But when it sees a console desk there, it will have some confidence that it’s near the front door. We would also like to compute a policy that maps history (actions and observations) to actions. The POMDP enables us to encode a new level of uncertainty into the decision making, which is suitable for a broader range of applications, however POMDP is known for its computational complexity. Let’s see how it’s used.

4 POMDP is usually deployed on powerful machines, because it needs that kind of strong resources to support its computation. Then why do people want to put it on mobile? First of all, the development of mobile technology provides the possibility of loading POMDP onto mobile devices. With an emerging generation of advanced devices such as smartphones and tablets, it is now feasible to let mobile devices to carry POMDP. In addition, we can utilize the sensors from mobile, including GPS, accelerometer to enrich POMDP’s perception of outside world. POMDP is an useful model, we can use mobile features to enlarge its availability and make POMDP accessible everywhere. Those are great ideas, but we still have some difficulties in the way, such as weak CPU, restrained memory, limited battery life, etc. To overcome these, many researchers have worked on the theoretical side to twist and improve POMDP structure and algorithms. We wanna try something different today. We study this problem from an engineering angle.

5 From the eyes of an engineer. Since eventually, no matter what algorithms we adopt, no matter what patterns we apply(mobile-alone or c/s), POMDP has to be implemented on mobile devices. It matters how we implement POMDP. Ideally, it should have some inherent rules saying how exactly we should implement this POMDP. But that’s far from practical. They are often too vague to formalize or had it been formalized, too tough to follow. And there are some existing good POMDP packages out there that no one wants to rewrite them. So we don’t do guideline.

We focus on the problem that given a set of POMDP execution implementations, how to select the best for a particular POMDP problem. The execution is underlined here because generally how POMDP works is divided into two parts. First compute a policy and then execute it accordingly. Since after all the mobile features are useful mostly during the actual execution. Selecting the best depends on a number of factors, device model, POMDP problem nature, implementation candidates. It’s OK. That justifies our experiment-based selection, which we will talk about in next slides. But a more important issue is that how we define “best”.

6 We describe our “best” for the good of mobile user experience. First we need to answer the question, what do mobile user care about the most? According to erison study, the top ones are slowness, weak signal strength, crashing down and short battery life. However, apart from short battery life, the other three happen rarely. And according to Nokia’s survey, short battery life is the top gripe among mobile users, and it’s the top one desirable feature users want to see improvement in their next smartphone. Battery is critical. Users don’t care about how much CPU we acquire, how much memory we occupy. As long as the POMDP application will not overwhelm the devices or other applications, the most concerning issue is battery life. Of course those implementations that go to the extreme and would destroy the whole thing should be ruled out from our implementation candidate set. It matters what POMDP implementations we choose. Because different implementations would have different resource usages. Even though we can’t see CPU usage behind, this usage will finally have effects on battery life. Therefore, even though the backend POMDP implementation is transparent to users, it is reflectable in the long run and it reflects on battery life.

Now, we’ve come down to one focus - battery life. We use the battery depletion rate to rank implementations, quantifying the goodness of implementations onto a single number. It’s easy for research purpose because the problem has been largely simplified. The ranking will serves as suggestion for developers, since in some cases the one with least battery depletion rate is not always the most desirable one. For example the client/server mode is selected as the best while we don’t have network connection in some situation. Developers would have to make the final decision. But with the help of our ranking based of BDR, it would be much easier.

7 So, how do we obtain the battery depletion rate. There are two ways. One is through pure experiments. We record battery depletion rate for every implementation candidate. That will require hours of experiment to obtain statistically stable results for each implementation, which is too low-efficient. Another way is to combine the experimental results with analysis to produce an estimated battery depletion rate. We need to do some benchmark experiments and obtain some other data to come up with the estimation. And that is our mobile battery consumption model.

8 The key idea of mobile battery consumption comes from a study conducted by researchers from CMU. In their study, they reveal that battery consumption comes from different components of the mobile device, among which the signal standby, screen display, CPU, WIFI/3G and sensors account for the most part of the consumption. Apparently, the consumption of each component is closely related to their usage. It’s our job to think of a way to translate this usage into battery consumption. Among the top ones listed here, we only consider CPU and WIFI. Because signal standby is always there, and it’s too unpredictable to examine. And screen display and sensors are also ignored because no matter what implementations we choose, we always have to feed in the same input and visualize the output the same way for a particular POMDP problem. Therefore, it doesn’t matter how much screen display and sensors consume, they are the same for all POMDP implementations during one selection.

9 Let’s get into the details of our mobile battery consumption model. Suppose the POMDP makes decision every T time interval, and the actual execution time is t, the rest T-t is idle. We have following formula to estimate battery consumption of a particular POMDP implementation.

When we use the mobile-alone mode, the WIFI is turned off. r equals to r\_CPU times t divided by T plus r\_Base, which is shown in the figure below. The r\_Base is the base line battery depletion rate, which we will explain later. We average r\_CPU on the whole capital T interval and use the average total r as the battery depletion rate. The one that uses WIFI is a little complicated, r equals to average battery depletion rate of baseline + wifi communication and wifi idle status. The important symbols here are r\_CPU, r\_WIFICom, r\_WIFIIdle, and r\_Base. We call it benchmark.

10 To talk about the benchmark. Let’s first define r\_Zero, it’s the battery depletion rate of mobile device doing nothing, serving as the fundamental for future calculation. r\_CPU is the additional battery depletion rate caused by CPU usage of the POMDP implementation, simulated by having full cycle CPU running and use its BDR to deduct the r\_Zero. Similar calculations apply to r\_WIFICom, it’s additional BDR of wifi communication, and r\_WIFIIdle, the additional BDR caused by having WIFI turned on. And the r\_Base is the BDR that caused by supporting the whole structure. Basically, during each round, we have to query sensors input, and call for decision from POMDP package. And the shadow part accounts for the additional BDR captured by r\_Base.

So the whole idea is that, (go back to previous slide). When we have the entire POMDP execution on mobile device, we only count the CPU usage. When we are using client/server mode through WIFI, we ignore the CPU usage and only count the WIFI usage. More precisely, in this situation, the CPU usage is already enclosed in the BDR of WIFICom. That’s all for our mobile battery consumption model.

11 With the help of this mobile battery consumption model, we can obtain BDR in a different way rather than straightforward battery experiments. For a particular POMDP problem and a given set of POMDP implementation candidates, we examine their BDR on three typical time intervals. First we obtain the benchmark through the methods we discussed before. And then simply record the actual execution time t in each round for every implementation candidate. We apply this information to our mobile battery consumption model and generates BDR for every candidate. Finally, use their average BDR on three time intervals as their battery performance indicators to rank implementations. That’s it. That’s how our mobile battery consumption model works.

12 It has some advantages over straightforward battery experiments. It’s really fast. Think about the situation where we have a lot implementations to study. As opposed to hours of experiment for every candidate, we simply need minutes of test that logs their actual execution time during per round, and combined with the benchmark, we can generate the battery depletion rate estimation. People may argue that obtaining benchmark is still pure battery experiment, that costs some time. But the benchmark (r\_CPU, r\_WIFICom, r\_WIFIIdle, r\_Base) is device-determined. It’s independent from whatever POMDP problems we are dealing with. That means, if we are doing experiment on this device, we only need one time benchmark test while if we totally count on pure battery experiments to obtain BDR, we have to re-do all experiment for a new POMDP problem. In the end, we have observed that benchmark can be shared among similar kind of device without disturbing the ranking order. Had it changed the ranking, it only means that the difference between two swapped implementations are very small. In fact we don’t have to care about this little difference for we might have more important factors to deal with such whether one of them can be fitted into memory. Therefore, we apply our mobile battery consumption model to our fulfillment of selection criteria based on battery depletion rate.

13 Next, let’s go through the experiments and evaluation part. We want to do some experiments to evaluate our work. First, what do we have? We have a mobile device, a particular POMDP problem, some POMDP implementation packages, and a mobile software architecture that allows easy plugin of POMDP packages. What do we want to do? We want to compare results from both real and pure battery experiments and estimation from our mobile battery consumption model. What do we expect to see? Rankings are the same, better if the BDRs are close.

14 A little bit details about the experimental setup. (read it)

15 The mobile software architecture. We do it without SIM card. One reason is for simplicity. Also we want to avoid the unpredictable interference from unstable signal strength. And WIFI is on when necessary. Screen brightness is fixed to 50%. We have two threads here, the UI thread also called the main thread is used to interact with user and display useful information. The functional thread created by UI thread is used to handle the invoking and receiving from POMDP package. The POMDP package is just an idea here. It can be a real executable function set or an interface of communication with a remote servers, and we replace it with void function when doing the rBase test. And also to further simplify, we generate observations from a random observation generators, rather than going through that sensor query to avoid interference from additional battery consumption caused by sensors. We log battery changes during the experiments.

16 Let's take a look at the result. First it's the real battery experiment part. We have all four Pomdp packages plugin one by one and then run each of them for three hours, log down time stamp of battery changes meanwhile. We test with three time interval settings 10 sec, 2 sec, 1 sec. Below is the results. We have four implms here. When we shrink the time interval, all the BDR increase. It's reasonable, because that means more decision making per minute, involving more additional battery consumption from the CPU or WIFI usage. We can have a overview here that FSC is the best while the symbolic Perseus consumes the most battery. We will get back to the ranking later.

17 Now, let take a look at the result from battery depletion rate generated by our mobile battery consumption model. We first test the benchmark. We have the mobile software architecture run for three time intervals and get the BDR of each. And then we test the rZero, which is essentially the mobile device doing nothing but has a simple battery info logging service in the background. We obtain rWithCPU by recoding BDR of a calculation of the 1000th Fibonacci number in a forever loop to simulate full cycle CPU usage. And rWithwificom is the BDR of mobile continuously sending and receiving a random integer to and from a remote server through wifi. rWithwifiidle is just the rZero setting with wifi turned on. After these experiments, we calculate the rCPU, rWificom and rWifiidle by subtracting the corresponding value by rZero. And next, we record the avg exe time for each of the four Pomdp implms. In the end, we feed all this information to our mobile battery consumption model and generate the BDRs. They look similar to those from real battery experiments. Let's do some comparison to have a closer look.

18 It’s good to see that ranking from two BDRs are the same. FSC makes it the first with least BDR, followed by Client/Server, then Flat policy and symbolic perseus is the the one with largest BDR.

19 Let’s look at the number. As you can see, the estimated BDRs basically align with the experimental BDRs. Their similarity measured by cosine-similarity is around 0.999. Cosine-similarity measures the closeness of two vectors by calculating their cosine value. It ranges from 0 to 1, where 1 indicates these two vectors are the same. Our estimated BDRs achieve 0.999 cosine-similarity with the experimental BDRs. It means our estimation well predicts the actual numbers.

20 After all these experiments and estimation, we display the ranking suggestion for developers’ reference. It’s organized by the order of BDRs, as you click the drop down menu here, it shows the BDR details. Also, as we discussed before, it lists some additional information because even though BDR is important, developers have to finalize their decision according to the real situation. Memory usage and WIFI requirement are listed here.

21 Conclusion. We’ve done some interesting work. We tackle the deployment of POMDP on mobile from the perspective of engineering, and use experiments to choose POMDP implementations. We quantify the goodness of POMDP implementation on battery depletion rate, and give good explanation about why battery is the most concerning issue in our selection. We propose a mobile battery consumption model to quickly estimate BDRs of POMDP implementation in the long run. Also, we build a framework to conduct battery experiments on mobile devices for POMDP implementations. Those are the primary contributions of the thesis. But beyond that, I think it has more meanings. Because no one has done this before, our work represents an innovative direction that when it comes to real world applications, working on theoretical part is one way, but the implementation is also very important and worth of study. Our work can be generalized for other complex models that have the same dilemma as POMDP when deployed on mobile devices. I believe it’s useful in the research of combining AI and ML models with mobile platforms.

22 As for the future plan, we want to conduct more research about the assumption that benchmark data could be shared among similar devices. Also, we need more experiments to verify our mobile battery consumption model. And probably some generalization of our work to make it suitable for not just POMDP model.

23 That will wrap up my presentation today. I would like to express my sincere appreciation and thanks to my supervisor Jesse, you’ve been a tremendous mentor for me, it’s really great to work with you. I would like to thank Pascal and Dan for your time and energy serving as my thesis readers. Also, I want to thank Marek for his help in preparing some of the experiments. In the end, a special thank to my friends, especially Chengbo Li and Luyuan Lin for their supports and trust all the times. Without any of you, none of this could be possible.