UNIVERSITY OF SOUTHAMPTON

Faculty of Physical Sciences and Engineering
School of Electronics and Computer Science

Bidding Optimization for Display Advertising under Budget Constraint

by

Watthanasak Jeamwatthanachai

September 2015

Supervisor: Dr Enrico H. Gerding
Examiner: Dr Mark J. Weal

A dissertation submitted in partial fulfilment of the degree of
MSc Computer Science

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

MSc Computer Science

by Watthanasak Jeamwatthanachai

This work is all about bidding optimization that can be used in display advertising. We study and investigate the problem of sequential second price auction that provides a partial observation to the winning agent. We also study bidding strategies can be used in a practical situation under budget constraint and uncertainty of market price for a single-sided auction. Besides, we assume the market price is random variable drawing from the gamma distribution. After that, we model and simulate the problem via MATLAB, and also decide to use the integer programming framework for the sake of simplicity of modeling, designing and evaluating the problem. Therefore, we propose our algorithms, heuristic approach, to tackle the problem that are comprised of three algorithms: *Greedy*, *Greedy-Knapsack* and *Improved Greedy-Knapsack* algorithms, which are improved step-by-step respectively. Therefore, we evaluate our algorithms and then compare ours with other existing algorithms such as simple, heuristic and also dynamic programming approach algorithms. The results show that the final version of greedy archive the best performance. We also evaluate the dynamic programming algorithm compared with our algorithms as a special case due to the processing time that take too long in practice. Obviously, our algorithms does not perform well as the dynamic programming does, but the results are not bad since our algorithms can be computed in polynomial time that contrast to the dynamic programming approach that computes in an exponential time.

Contents

List of Figures

Nomenclature

v_a	Bid valuation of agent A
v_{-a}	Bid valuation of other agent, except A
a_t	Auction at a particular time step t
g	Probability Distribution function (PDF)
$g(x)$	Probability of Bidding x
G	Cumulative Distribution function (CDF)
$G(x)$	Probability of Winning given bid x
α	Shape parameter in gamma distribution
β	Scale parameter in gamma distribution
T	Number of auctions
t	Time step
N	Number of the remaining auctions
B	Initial Budget
B'	Remaining Budget
B'_t	Remaining Budget at particular auction a_t
b_t	Bid of our agent at a particular auction a_t
b_t^*	Optimal bid at a particular auction a_t
b_t^{max}	Maximum bound of bid price at a particular auction a_t
$\vec{b}$	Set of series of bid price
$\vec{b}_{bid}$	Series of bid price
p	Payment
p_t	Cost of payment in a particular auction a_t
x_t	Market price in a particular auction a_t
$\vec{A}$	Set of fraction of budget
A_t	Fraction of Budget

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Enrico H. Gerding, who has given a lot of good suggestions thus many time, since this is my first time here study abroad in United Kingdon. It has been honored doing this dissertation under his supervision. He has taught me various of techniques that can be used for future research. Apparently, Any contribution from him is such a good this for all time study in the University of Southampton.

I am also thankful for my family for brought me up for 30 years with lots of assistance to my pursuits to my dream and all their love and encouragement that refill me as an energy for everyday life in United Kingdom. Also, heartily thanks for any supported things from Thailand such as books, foods, and so on so forth.

Lastly, I, also, would like to thank one of my special friend Le Le for forming a team of study for a whole year of studying in MSc Computer Science. Inevitable having to say thank to my friends from other departments and faculties: Vicharinee Saengthong, Jedidiah Wu and others for helping, giving a suggestion of anythings in United Kingdom, and also making my days thus fantastic during the summer time, even though we have known each other in a short period.

Chapter 1

Introduction

In these days, there are lots of people around the world using the internet in their workplace, home and everywhere. Of course, the number of internet activity is raised up exponentially. For that reason, one of well-known business marketing called Display advertising is taken place, which is the marketing that focus on finding the most suitable content given a numerous factor, such as size, position and user's behavioral profile, for instance age, occupation, location, time zone, web browsing history, purchased history, etc.) via Internet cookies and then placing that on the website, also known as a *publisher* who offers available ad spaces for ads, generated by *advertisers* who bid highest in the auction, through *Ad Exchange* (Muthukrishnan, 2009). Ad Exchange is the company who run the auction when an ad is requested. In Figure 1.1, the auction process begins when the user visit website, Ad Exchange announces the available ad spaces to bid managers, which is *demand-sided platform* (DSP) acting on behalf of advertisers. Therefore, the DSP evaluates the advertiser's targeting and decide to whether place bid or not. After that, all bids have been placed on Ad Exchange, and then the winning bid will be determined. The highest bid is the winner, and he pay a second-highest bid since the main mechanism is second-price sealed bid auction, also known as Vickrey auction. After determining the winner, Ad Exchange will announce the winning bid back to the winning advertiser, and the payment has been made, and the generated ad will be transferred to the website (publisher) simultaneously. The state of the ad is rendered on the user's browser is so-called *Impression*. Note that the rendered content could be text, image, video, flash animation, and so on so forth.

Having mentioned thus far that this is why display advertising is significant in these days. Because of the main objective of display advertising is to deliver the message to the consumer directly, and creates an enormous amount of income to advertising companies according to the increasing trend of using the Internet these days. Besides, display advertising ads are generated appropriated content to the user in real time whenever user visit the website regarding given information provided by the website

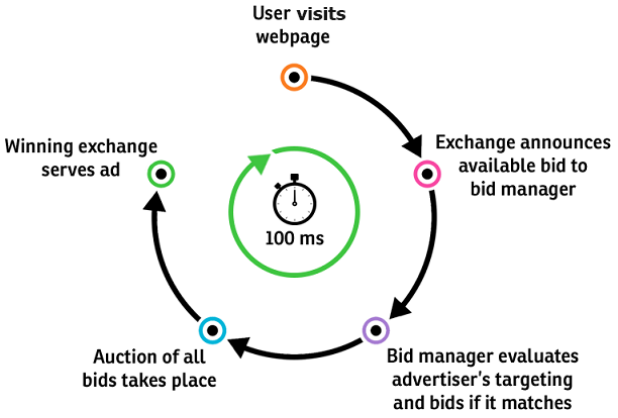


FIGURE 1.1: Why real-time bidding in display advertising is set to become huge? http://searchenginewatch.com/sew/opinion/2293623/why-real-time-bidding-is-set-to-become-huge

(publisher) and user interests. On top of that display advertising does not only make a lot of profit to advertising companies but also make a huge profit to brand companies.

In research aspect, the interesting thing in display advertising that can lead to a research question is how to place bid appropriately under budget constraint in the second price sealed bid auction. Additionally, the number of transactions is extremely high, 100 millions transaction a day approximately, By the way, budget constraint is also the most significant factor as it comes into play as the main parameter in this auction. Advertisers do not want to be out of money in the auction that could make them lose in the rest of remaining auctions. There are many recent studies (Tran-Thanh et al., 2014; Ghosh et al., 2009; Gummadi et al., 2012) have been developed models by adopted various techniques of machine learning since the objective of machine learning is to learn and understand the activity in a complex system in order to define the computational model for future prediction given the information provided by the system.

In our work, we develop a simulation of the second price sealed bid auction and bidding strategy as a point of view of an agent under the budget constraint and uncertainty of market price in single-sided auctions. To be able to learn and understand uncertainty, we deploy Gamma distribution as a market price model that supports the data between 0 to positive infinity (Gamma Distribution, 2015). We, then, adopted machine learning (see Chapter 3 for more detail) to learn market price distribution, which is censored information. The agent will learn if and only if he place the highest bid in the auction and pay the market price, which is the second highest bid in that auction (Tran-Thanh et al., 2014).

1.1 Problem Statement

Given the display advertising definition, in this dissertation, we focus on finding bidding strategy to bid effectively. The problem is taken place that agent[1] have to place bid with the right price under budget constraint when the market price is not known and learning the market price need to be learnt as a must. Also, the auction, in display advertising, is happened thus quickly (100 milliseconds approximately) as it shows in Figure 1.1. We can see that the duration in one single auction is finished in extremely fast. If the agent cannot place their bid in time, he will lose the auction. Also, if he uses a simple strategy, for instance, bidding with his budget. Of course, bidding with the budget, as his valuation v_a, is the good strategy, but it is good where the auction is a single auction. However, for sequential auction, bidding the budget is not a good strategy to go because he might be out of budget in the rest of auctions due to the unknown of other bidder behavior, let's call market price distribution. That could make the agent lose all of the remaining auctions that could also make the agent lose the huge benefits. As a consequence, this leads to the research of real-time bidding optimization.

1.2 Motivation and Challenges

As we already knew the problem from previous Section 1.1, such an enormous benefit brought by display advertising industry. That leads to our motivation of this research that what is such bidding strategy could be suitable using in practice that such environment does not provide a sufficient information, and the budget is limited. There are so many studies of bidding strategy regarding display advertising in the literature of bidding optimization. Unfortunately, most of them did not answer much in the given problem of our work.

Having stated, that lead to our research topic of how to place bid with the right price that maximizes the number of wins under the budget constraint. The challenge here is that the agent does not know the market price distribution. If he wants to learn, he has to pay for the observation. He will never know that his placed bid is going to win or not. That could cost him a lot, and also make him expelled from the remaining auction due to the limited budget in sequential auctions.

[1] We will use the term of agent instead of advertiser within this dissertation.

1.3 Contribution

The main objective in this dissertation is to construct bidding strategy that maximize the number of wins for a given specified budget for sequential auction.[2] We propose the bidding strategy and compare our strategy to other strategies in the literature, and the strategy we build should satisfy the following requirement:

(a) The model follows the essential characteristics of the second price sealed bid auction that is modeled with a single slot in sequential auctions.

(b) The model must be usable and computationally feasible in practice. Also satisfying real-time display advertising auction's process that happens within 100 milliseconds.

(c) For all algorithms, the summation of bids spending for a time horizon must not be over the specified budget, and algorithms must spend all of their budget for a particular number of auction playing in the simulation.

(d) The budget is reset to the initial budget whenever simulation is started.

To be able to achieve the goals we consider the optimization approach instead of game theoretic approach (Bertsimas et al., 2009) since the simulation we produce is a single-sided auction. It is essential that the bidding strategy must be practical since the auction happens within 100 milliseconds. Even though the agent performs the best results, it is useless according the processing time of estimating the right bid is more than 100 milliseconds. That means the agent will lose in the auction, and also lose the rest of the auctions. Nonetheless, our goal is to study the bidding behavior and yet come up with bidding strategy, called Greedy bidding strategy (see also Chapter 3 in more detail, and Chapter 4 for some empirical results compared with other strategies)

In this work, we adopt a machine learning framework as the main part of bidding optimization, since we do not know what the market price distribution looks like. Therefore, market price distribution is defined as a probability distribution function (Tran-Thanh et al., 2014; Bertsimas et al., 2009) that depends on the numbers of market price reported to the agent. The probability distribution is the most significant parameter, playing as a main part of optimization, and that determine what is the right price of the auction in particular time t a_t. Moreover, we have simulated our bidding strategy in a setting where the probability distribution is represented as competitors (market price). Afterward, we show the results by comparing other approaches to see how the performance of our approach is going (see more detail in Section 4.2). However, Our approaches shows that the performance is better that other strategies, except dynamic

[2]Some people called this auction as repeated auction. In the dissertation, we interchangeably use the term of sequential auction and repeated auction afterward.

programming approach. Even though the outcome of our approach is not good as dynamic programming approach, the processing time is faster and within 100 milliseconds. Consequently, we feel like that our approach is suitable and applicable bidding strategy in practice.

Having mentioned that we feel like this dissertation produces the following contributions:

(1) We introduce a bidding strategy for display advertising that **maximizing the number of wins** subject to **the budget constraint** and also satisfying the requirement (a)-(d), stated in previous Section 1.3. The bidding strategy are constructed based on machine learning framework, which is used as a learning technique in part of optimization.

(2) We showed the result of comparison between our approach and other approaches under budget constraint, in simulation, using stochastic market price model brought by gamma distribution in T-repeated auctions which is only single slot each.

(3) We compared the result of our approach with dynamic programming approach, Bellman equation initially presented by (Amin et al., 2012), with extension with using Zeng's estimator instead of KM estimator by (Tran-Thanh et al., 2014). In our work, we consider MLE provided by MATLAB instead of using Zeng's estimator.

Chapter 2

Literature Review

2.1 Second Price Auction and Sequential Auction

Bidding optimization is widely studied and used in many such various auctions in the world. For the most part, bidding optimization is usually used in find the optimal price regarding a given information in that auction. However, in our work, we are only interested in a sealed bid second price auction, also known as Vickrey auction. In Vickrey auction as we already knew that truthfully bidding, place bid as an item valuation v_a, in a single auction, is a dominant strategy (Krishna, 2009). In our work, we consider the problem that budget constraint is included in part of the strategy. Having mentioned, that implies that placing the bid with the budget remaining is a dominant strategy if it is only a single auction since there is only one item that agent is willing to get (Krishna, 2009). However, it does not seem to work on sequential auction (Vetsikas et al., 2013; Amin et al., 2012; Ghosh et al., 2009; Gummadi et al., 2012), also known as repeated auction, since agent has a limited budget. For instance, the agent has a limited budget and place bid equal to the budget he has at a certain time. The result would be that he wins for all auction in the case of he has an infinity of budget. On the other hand, he would lose the auction in the middle of the day if he has a limited budget. The last case it seems realistic to our problem. Thus we can summarize that bidding with all of budget is not a dominant strategy in repeated auctions under the budget constraint (Vetsikas et al., 2013). That leads to the research question of bidding optimization that have been studied in the literature. Since bidding optimization is widely used in the complex system that enable the autonomous agent can be able to learn the bidding behavior in that such environments that provide partial information to the winning agent (Bertsimas et al., 2009; Tran-Thanh et al., 2014; Ghosh et al., 2009).

In our work, there is two part that we focus on **(1)** learning and prediction and **(2)** bidding strategy. For that reason, we adopt machine learning framework as a main part of optimization. After that, we use the past performance, also known as market price

history, that the agent is informed to construct the distribution that is the beginning of our algorithms named *Greedy* algorithm.

2.2 Multi-armed Bandit with Budget Constraint

For all of mentioned reason, we feel like our work is closed to the work of Ghosh et al. (2009) that consider partial observation, which is similar to our work. In this work, the author presents the strategy that can reduce the trade-off between exploration and exploitation. They also propose the different techniques by using multi-armed bandit, where each lever is represented as a bid price can be pulled in the next round. Apparently, that implies that whichever lever is pulled, the agent has to pay for its cost. That problem is related to the literature of multi-armed bandit with fixed cost under budget constraint (Ding et al., 2013; Sen and Ridgway). The study of fixed-cost multi-armed bandit under budget constraint presented by Sen and Ridgway.

In this study, the author presents the algorithm for finding the sequence of arms that is going to be pull, which is similar to our work as we also find the sequence of bid price, $\vec{b}$, considering the highest accumulative probability in time horizon. Interestingly, the main idea of this work is completely opposing to our aspect. Given the budget to the agent and variable cost to each arm, they present the arms selection that always choose the arm that produce a high reward, otherwise, eliminate that arm. The interesting idea in this work is the algorithm named l-split algorithm. This algorithm is to pick one arm, for example arm_1, then determine the next arms, $arm_{t>1}$. Each time that picking arm the budget is deducted according to the cost of that arm. This process continues processing until the budget is zero, or no one arm can be pick up. Repeating this process for each arm until at the end of determining the last arm, it produces the set of arms series. After that, calculating the reward on average, and pick arm that produce the highest reward. We feel like this approach is interesting and similar to the *Online Knapsack Problem* (Lueker, 1998) that we are interested in using so. However, the idea is interesting but we feel like some part we can adopt to our approach, especially that part of determining the series of arms since, in our approach, we consider all possible bid price for the future auction in time horizon. Having mentioned online knapsack problem that the main idea of knapsack problem, in our case, is to place the bid that does not exceed the capacity, which is the remaining budget, B'_t. This technique is used and mixed with the future bid motivated by l-split algorithm (Sen and Ridgway).

2.3 Bidding Strategy under Budget Constraint

By the way, Ghosh et al. (2009) also consider budget constraint as a main parameter using in estimating the optimal bid. They propose an adaptive algorithm using collected

history data in each game, called *Learn-while-Bid* that is closed to our approaches that learning in both exploration and exploitation. Also, they introduce another mechanism that can reduce the trade-off between exploration and exploitation named *Panic*. Panic is an aggressive bidding strategy triggered when the number of impressions is not met when budget remaining is running out. However, for the above-mentioned reason, agent has to perform on repeated auctions that is relevant to the study of Gummadi et al. (2012) who consider the budget constraint and propose an optimal bidding strategy and also equilibria in sponsored search auction. However, the author does not consider only the budget constraint but also take income factor into the consideration that can extend the number of auctions for that agent. Our work does not include in parameter, on the other hand. This work also presents the greedy approach cooperating with Knapsack problem (Lueker, 1998), and also consider the valuation of an item is including the greedy approach, which is not used in our work. Another work that connect to budget-limited auction is belonging to Zhou et al. (2008). This work is a bit similar to our work since they consider budget constraint as a part of online knapsack problem for estimating bids for multiple slots. We also use online knapsack problem as the multiple choice of the bids that using in future bid declaration. For this technique also points to the last work in the next paragraph.

Finally arriving at the last review that is closest to our work belongs to the work of Tran-Thanh et al. (2014). The author evaluates regret bound, which is the factor for online optimization, for the existing bidding strategies in the literature of online advertising auction, repeated second price auction. In this work, the author also considers partial information (observation), also known as censored information is only provided for the winning agent. There are three algorithms presented in this work, and also working under budget constraint with specified time horizon, (1) ϵ-First, (2) Greedy Product Limit (GPL) and (3) LuekerLearn algorithm. For the first two algorithms are dynamic programming (DP) algorithms that adopt Bellman Equation, Equation 3.8, is a part of optimal bid estimation. The difference in both algorithms is a bit different that ϵ-First provides an exploration phase, but no exploration in GPL algorithm, on the other hand. The last algorithm *Lueker Learning* algorithm, the author, claimed that it performs well. The idea of Lueker Learning is quite simple adopting from the N-Average case, originally presented by Lueker (1998). That choose bid defined as expected payment that below $\frac{B'_t}{T-t+1}$, as it show in the Equation 3.7. Also, considering the update distribution which is Maximum-likelihood (MLE). However, the author considers Suzukawa's estimation for ϵ-First and Zeng's estimator for GPL and LuekerLearn algorithms. That is contradicted in our work that use MLE provided in MATLAB. Interestingly, in Bellman equation, the main idea of this equation is to find the optimal bid for all time horizon, in term of maximizing the number of wins, given the updated market price distribution. The result of this equation is completely promising since it is claimed as the optimal bidding. Notwithstanding the excellent outcome, the author proved that the processing

is considerably too long that cannot be used in practice. That work are contradicted to LuekerLearn that can be run in polynomial time.

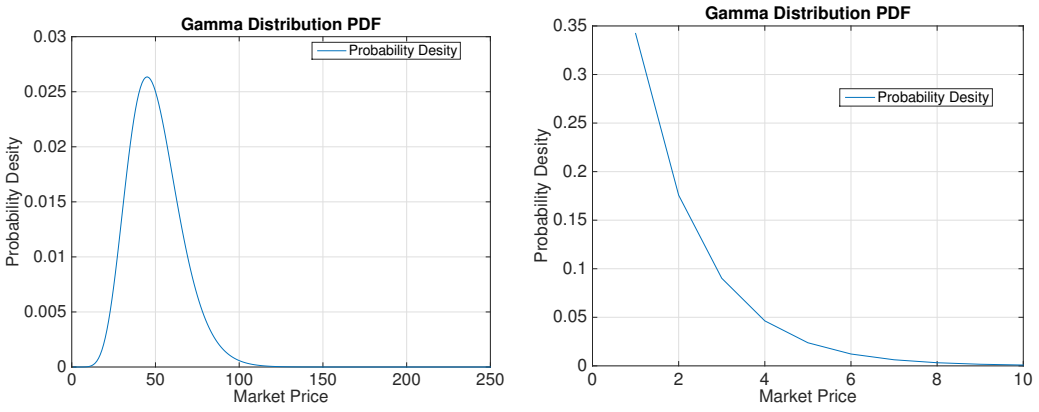
For the reviews of the literature as mentioned earlier, we feel like learning distribution is necessary for any bidding optimization that will be regarding the parameter of distribution estimation that has come into play as a primary role. If the parameter estimation equation is not well-designed, the result might be biased. Considering the bidding strategy for all time horizon is the most significant as it makes a call of what bid price should be placed. If it decides to place the bid too high, it is going to influence the future bid due to the limited budget. For this reasons, we feel like learning market price distribution, distribution's parameter estimation and bidding strategy are the most import thing for our work, and they will be taken into consideration at the most priority.

Chapter 3

Model and Algorithms

3.1 Model Description

In our work, the model is similar to the work of Tran-Thanh et al. (2014). Model is consists of agent A is willing to place repeatedly his bid in a sequential auction with time horizon T of a single item sealed bid second price auction. The agent will place their bid in time step $t \in \{1, ..., T\}$ against the market price as generating bids from a probability distribution g, which is drawn from the gamma distribution given parameters: $\alpha = 10$, $\beta = 5$ and $\alpha = 1$, $\beta = 1.5$, for normal case and special case, respectively (see more detail in Section 4.3).



(a) $\alpha = 10$ $\beta = 5$ (b) $\alpha = 1$ $\beta = 1.5$

FIGURE 3.1: Gamma distribution: Probability Desity Fucntion

Budget is denoted as B for the time period T and remaining budget also is denoted as B', but for the remaining budget at a specified time step t, we use B_t. In the auction, the agent has to pay if he wins the auction. Thus, we define p_t to denote the payment at time step t if he wins. Due to the budget constraint, this means that the total cost of payment should not be over the budget as it shows in a following Equation 3.1:

$$\sum_{t=1}^{T} p_t \;\leq\; B \tag{3.1}$$

In Vickrey auction, the payment is equal to the second highest bid in a certain auction. As we defined market price as a random variable x_t for a single slot that is independent and identically distributed (i.i.d) drawing from the probability density function p brought by gamma distribution with given parameters that we mentioned above. Since the market price are drawn from unknown, but the market price should be more than zero to make the model to be more realistic and reasonable. That means that at the time step t, the budget will be deducted by x_t if the agent place bid b_t is larger than the market price x_t. Otherwise, it will be deducted by nothing since he does not win the auction. This statement implies that agent is permitted to place his bid less than or equal to his budget $b_i \leq B'_t$, which is reasonable and realistic. Note that agent always start with an initial budget $B'_1 = B$.

$$B'_{t+1} = \begin{cases} B'_t - x_t & \text{if } b_t \geq x_t \\ B'_t & \text{otherwise} \end{cases} \tag{3.2}$$

Since the main goal of our strategy is to maximize the number of wins in the time period T. Thus, we develop our strategy called Greedy Bidding strategy. In this strategy, the main idea of the strategy is to design as simple as LuekerLeann bidding policy, as it will be shown in more detail in the next Section 3. Our strategy is completely simple that can be processed in a polynomial time $\mathcal{O}(T)$, where T is the number of items (time horizon).

3.2 Parameter Estimation

As we have mentioned, the most important is learning market price distribution in our model. That is because the agent does not know the market price distribution yet and need to use the distribution in term of finding the optimal bid, which is necessary to be learned to achieve the objective. In our model, we consider a partial observation that is setting that the winner will be informed the market price. Every time the agent win the auction, he will be informed the market price from the mechanism. After that, he updates the distribution given the vector of market price[1]. This update is a so-called online update that is more efficient than offline update[2]. In our approach, the updating process is still happened even thought in exploitation (in more detail see Section 5, 6 and also 7).

[1] Collection of market price that he won auctions in the past, also known as history data

[2] Offline update is updating the distribution only in the exploration phase, then use the knowledge in exploitation without any updating process

To be able to estimate the parameters, *alpha* and *beta*, we adopt Maximum-likelihood Estimation (MLE)[3] in case of calculating that parameter. To use this estimation, we need to be careful since the Maximum likelihood tries to overfit when the number of market price (wins) is not sufficient.

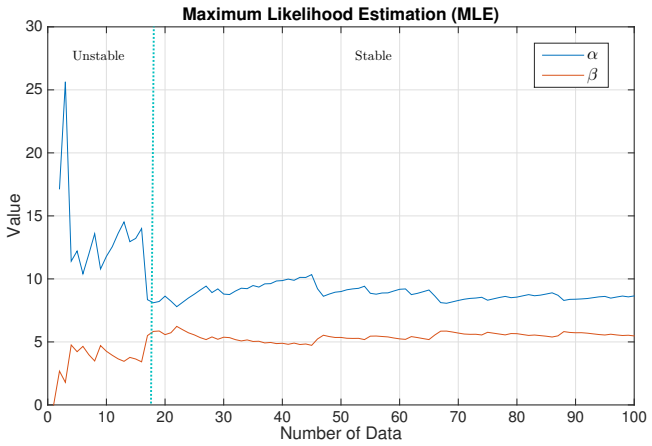


FIGURE 3.2: Maximum Likelihood Estimation Performance

In Figure 3.2 shows the MLE performance regard to the number of data. We can see that when the number of data is not sufficient, both parameters are not stable. That is because, as we mentioned, that MLE try to overfit the data. After the number of data is more than 18, the parameters become stable. This figure suggests that the minimum number of data the agent should have, in term of learning the distribution, is 18 at least. Therefore, we define the number of exploration is 30 ($\epsilon = 30$), which is more than 18. That means that the agent has to win at least 18 times, and if we set $\epsilon = 18$, it is not guaranteed that he is going to win all of them. Thus $\epsilon = 30$ is reasonable.

3.3 Existing Algorithms

Given the model in the Section 3.1, our goal is to find the optimal strategy that maximize the number of win given the limitation of budget, B, in the specificed time horizon, T. Thus, we modified the expectation of number of win accorded to the work of Tran-Thanh et al. (2014) as a following:

$$W^A(B'_t, T) = \mathbb{E}\left[\sum_{t=1}^{T} b^A(B'_t, T) \geq x_t\right] \tag{3.3}$$

In Equation 3.4, we define the number of win, W'_t by using bidding strategy A. Therefore, we run a number of experiments to find the optimal strategy that maximizing the number

[3]In the simulation, we use *gamfit* function to estimate both parameter given the data (market price vector). This function is provided in MATLAB (Gamma Parameters Estimation, 2015).

of wins in each strategy. Finally, the optimal strategy is chosen by a following equation here:

$$A^* = \max_A W^A(B, T) \tag{3.4}$$

Given the problem definition and model description above, we consider existing bidding strategies in the literature in comparing of our strategies, which will be described in Section 3.4. We start with a various of simple strategies and then existing strategy from the literature of Lueker (1998) and Tran-Thanh et al. (2014).

3.3.1 Aggressive Budget Bidding Strategy

Aggressive Budget bidding strategy is a simple bidding strategy that always places bid aggressively with the budget remaining at all the time. The bid price is defined as a following equation:

$$b_t = B'_t \tag{3.5}$$

Algorithm 1: Aggressive Budget

Input : $B'_t > 0$
Output: b_t
1 **for** $t \in \{1, ..., T\}$ **do**
2 | $b_t = B'_t$
3 **end**

FIGURE 3.3: Aggressive Budget Algorithm

The idea of this algorithm is to bid aggressively for the whole period of time horizon with all of the budget, B'_t, he has at the time step t that shows at line 2 of Figure 3.3.

3.3.2 N-Fraction of Budget Bidding Strategy

N-Fraction of Budget bidding strategy is also a simple bidding strategy that split the initial budget B into N fractions, where $N = T$ is the number of auctions in total of time horizon. This strategy is a bit different to the previous algorithm at line 2. Thus, this algorithm will be defined as showing in the Figure 3.4.

$$b_t = \frac{B}{T} \tag{3.6}$$

Algorithm 2: N-Fraction of Budget

Input : $B > 0$ and T
Output: b_t

```
1 for t ∈ {1, ..., T} do
2 |   b_t = B/T
3 end
```

FIGURE 3.4: N-Fraction of Budget Algorithm

3.3.3 LuekerLearn Bidding Strategy

LuekerLearn bidding strategy is a heuristic bidding strategy presented by Tran-Thanh et al. (2014) which adopt online knapsack problem issued by Lueker (1998). The main idea of this strategy is protect agent to no more spent too much money by capping the price according the number of auction remaining N as a fraction parameter by placing bid that does not exceed the maximum bound, $\frac{B'_t}{T-t+1}$, showing at line 2 in Figure 3.5. Also, the expected bid price, b^*_t, is calculated as a payment given bid b, $\sum_{\sigma=1}^{b} g(\sigma)\sigma$. After that, update the distribution is required, $g_{t+1}(x)$, if placed bid is higher than the market price at line 3. Note that this strategy is also known as *N-Average Case*.

$$\begin{aligned} b^*_t &= \max\{b\} \\ \text{s.t.} \quad &\sum_{\sigma=1}^{b} g(\sigma)\sigma \leq \frac{B'_t}{T-t+1} \end{aligned} \tag{3.7}$$

Algorithm 3: LuekerLearn

Input : $B'_t > 0$, t, T, and $g_t(x)$, where $g_1(x)$ is uniform
Output: b^*_t

```
1 for t ∈ {1, ..., T} do
2 |   Place bid b*_t according to Equation 3.7
3 |   Update g_{t+1}(x) by MLE if b*_t ≥ x_t
4 end
```

FIGURE 3.5: LuekerLearn Algorithm (N-Average Case)

Note that in line 4, in the original version of LuekerLean algorithm, the author use Zeng's estimator as a distribution parameters estimation. From now on, we consider MLE function provided by MATLAB since it is comfortably used.

3.3.4 Dynamic Programming Bidding Strategy

This strategy is claimed as an optimal bidding strategy presented by Amin et al. (2012). The Equation 3.8 is used as a part of estimating the optimal bid. It is also known

that Bellman equation in another name is a dynamic programming (Tran-Thanh et al., 2014). The idea of this strategy is to find an optimal bid b_t^* by considering the future bid $b^*_{t+1\ \in[t+1,...,T]}$ as an optimal as well as in a current time t. As same as the previous one, update distribution is also required at line 8 in Figure 3.6.

$$b_t^*(B'_t,t) = \arg\max_{b(B'_t,t)} \left\{ \sum_{\sigma=1}^{b(B'_t,t)} \hat{g}_t(\sigma)[1+G^+(B'_t-\sigma,T-t)] + \hat{F}_t(b(B'_t,t))G^+(B'_t,T) \right\} \tag{3.8}$$

$$G^+(B',T-t) = \sum_{\sigma=1}^{b^+(B'_t,t)} \hat{g}_t(\sigma)[1+G^+(B'_t-\sigma,T-t-1)] + \hat{F}_t(b^+(B'_t,t))G^+(B'_t,T-t-1)$$

However, in our experiment, we modified the original version using MLE instead of Zeng's estimator, at line 8, due to the comfortable applying to our simulation. Considering the time constraint and our limitation on processing this in practice, we also assume that agent should have won at least 10 rounds in term of building the market price distribution, as it showing in line 3-5 as an exploration phase. The main reason for doing so is that we would like to see the different result of Bellman equation compared with our algorithms during the exploitation phase. Therefore, we call this algorithm as ϵ-*First Algorithm*. For the evaluation of this algorithm, we consider this as a special case since it cannot be estimated in polynomial time. We will show in another setting that is suitable for practice, see Section 4.3.

Algorithm 4: Dynamic Programming (DP)

Input : $B'_t > 0$, t, T, $g_t(x)$, and ϵ
Output: b_t^*

```
1  for t ∈ {1, ..., T} do
2      ε = 10
3      if t ≤ ε then
4          Exploration Phase
5          b*_t = B'_t
6      else
7          Place bid b*_t according to Bellman Equation 3.8
8          Update g_{t+1}(x) by MLE
9      end
10 end
```

FIGURE 3.6: Dynamic Programming (DP) Algorithm

3.4 Proposed Algorithms

In this section, we propose three algorithms. 1) *Greedy algorithm* (3.4.1), 2) *Greedy-Knapsack* (3.4.2) and 3) *Improved Greedy-Knapsack* (3.4.3). In the first algorithm *Greedy Algorithm* is the simplest version of our approach, and then we modify the first one into an improved version *Greedy-Knapsack* adopting online knapsack problem-solving technique (Lueker, 1998; Tran-Thanh et al., 2014). The last one, we, finally, consider the parameterized approach toward to *Greedy-Knapsack* to see which number produce the best result, and then we propose the last algorithm called *Improved Greedy-Knapsack* with optimal value as a fraction of budget in order to cap the bid price in particular time step t.

3.4.1 Greedy Algorithm

Greedy algorithm is an online algorithm and heuristic technique that solving the complexity problem by calculating which choice is an optimal choice given the learnable knowledge provided in an environment at each stage. For example, in our work, information of market price when agent win the auction x_t, Budget remains in each time step B'_t, number of remaining auctions N, and so on so forth. We also learn the market price distribution, $g_t(x)$, then calculate the probability of winning given bid, $G_t(x)$. Since the information is limited to our work, we define the simple version of the greedy algorithm that looking the bid that produce the most probability in time horizon as a following Equation 3.9.

$$\begin{aligned} b_t^* = \ & \arg \max_{b_1 \in \{1,\dots,120\}} \sum_{j=1}^{N} G_t(b_j) \\ \text{s.t.} \quad & \sum_{j=1}^{N} b_j = B'_t \\ \text{where} \quad & b_j = \frac{B'_t - b_1}{N-1}, \qquad j \in \{2, ..., N\} \end{aligned} \tag{3.9}$$

The idea of our Greedy algorithm is to find bid b_t from 1[4] to 120[5] that produce the highest probability in time horizon, called the optimal bid b_t^*, and the future bids $b_{j>=2}$ are simply defined as a $N-1$ fraction of the remaining budget B'_t excluding the current bid b_1. Note that the marker price distribution is required for all our algorithms.

[4]Why it does not start from 0? It is because the main goal of our is to maximize the number of wins. If we bid 0 this means we lose the auction that is not satisfied our objective.

[5]This number is the maximum bid price that agent allow to bid, which should be below or equal to the remaining budget, $maxBid \leq B'_t$

Algorithm 5: Greedy

Input : $B'_t > 0$, t, N, $G_t(x)$ and ϵ
Output: b^*_t

1 **if** $t \leq \epsilon$ **then**
2 | *Exploration Phase*
3 | $b^*_t = B'_t$
4 **else**
5 | *Exploitation Phase*
6 | $b^{max}_t = 120$
7 | **if** $b^{max}_t > B'_t$ **then**
8 | | $b^{max}_t = B'_t$
9 | **end**
10 | **for** $bid \in \{1, ..., maxBid\}$ **do**
11 | | $b_1 = bid$
12 | | $b_{j\in[2,...,N]} = \frac{B'_t - b_1}{N-1}$
13 | | $\vec{b}_{bid} = \{b_1, ..., b_N\}$
14 | **end**
15 | $\vec{b} = \{\vec{b}_1, ..., \vec{b}_{b^{max}_t}\}$
16 | Place a bid b^*_t according to Equation 3.9 given a set of $\vec{b}$
17 | Update $G_{t+1}(x)$ using MLE if $b^*_t \geq x_t$
18 **end**

FIGURE 3.7: Greedy Algorithm

Figure 3.7, the idea of this algorithm is to split into two past: exploration and exploitation. In exploration phrase, line 3, we place bid with the remaining budget, b'_t, we have in time step t. After that, in exploitation, we define the optimal bid by looking the bid b_1 varying from 1 to 120, at line 10, while the future bids, $b_{j\geq 2}$, are defined as a fraction of the remaining budget excluding the current bid at line 12. After that, we will get the set of vectors of bid $\vec{b}$, line 15. Then, the optimal bid is chosen by picking one vector in the set that is the highest accumulative probability for a whole period of time horizon according to the Equation 3.9. Note that update market price distribution is necessary in all our approaches.

3.4.2 Greedy-Knapsack Algorithm

Greedy-Knapsack is an improved version of the Greedy algorithm using online knapsack problem proposed by Lueker (1998). This technique is adapted to our approach as budget-limited condition since the main idea of knapsack is to spend money no more than nor exceed the capacity, which is the budget B in this case. We also found that defining the future bid b_j as a $N-1$ fraction of the remaining budget, in the previous equation 3.9 is not reasonable. They could be any number from 1 to B'_t. For more detail, we will give an explanation in Evaluation Section 4.1.2. We, therefore, consider N-Average case as a part of determining the future bid. For this reason, we define the

future bid $b_{j\in[2,...,N]}$ as a fraction of the remaining budget N, where N is the number of auction remaining, at line 13 in Figure 3.8. Therefore, we define the improved version of the Greedy Algorithm as the Equation 3.10.

$$
\begin{aligned}
b_t^* = \ & \arg \max_{b_1 \in \{1,...,120\}} \sum_{j=1}^{N} G_t(b_j) \\
\text{s.t.} \quad & \sum_{j=1}^{N} b_j = B_t' \\
\text{where} \quad & b_j = \frac{B_t' - \sum_{\sigma=1}^{j-1} b_\sigma}{N - j + 1}, \qquad j \in \{2, ..., N\}
\end{aligned}
\tag{3.10}
$$

Algorithm 6: Greedy-Knapsack

Input : $B_t' > 0$, t, N, $G_t(x)$ and ϵ
Output: b_t^*

1 **if** $t \leq \epsilon$ **then**
2 | *Exploration Phase*
3 | $b_t^* = B_t'$
4 **else**
5 | *Exploitation Phase*
6 | $b_t^{max} = 120$
7 | **if** $b_t^{max} > B_t'$ **then**
8 | | $b_t^{max} = B_t'$
9 | **end**
10 | **for** $bid \in \{1, ..., b_t^{max}\}$ **do**
11 | | $b_1 = bid$
12 | | **for** $\sigma \in [2, ..., N]$ **do**
13 | | | $b_\sigma = \frac{B_t' - \sum_{\sigma=1}^{j-1} b_\sigma}{N - \sigma + 1}$
14 | | **end**
15 | | $\vec{b}_{bid} = \{b_1, ..., b_N\}$
16 | **end**
17 | $\vec{b} = \{\vec{b}_1, ..., \vec{b}_{b_t^{max}}\}$
18 | Place a bid b_t^* according to Equation 3.10 given a set of $\vec{b}$
19 | Update $G_{t+1}(x)$ using MLE if $b_t^* \geq x_t$
20 **end**

FIGURE 3.8: Greedy-Knapsack Algorithm

3.4.3 Improved Greedy-Knapsack Algorithm

As we introduced the Greedy-Knapsack algorithm in the previous, we define the last approach motivated by the parameterised version of N-Average Case algorithm, whose results suggest that N-Average Case, where N is the number of auction remaining, is

not the optimal value. We will call this parameterised version of N-Average Case as A-Average Case, in order to avoid the confusion, where A[6] is linear function of fraction of the budget defined as a following Equation 3.11 which is calculated by the Equation 3.12 where A_{start} is 50 and A_{end} is 1. For finding an optimal value of this equation, we will evaluate in next chapter 4.1.3.

$$\begin{aligned} \vec{X} &= \{0, 1, ..., T-2\} \\ \vec{A} &= \{50 + (X_t \frac{1-50}{T-1}), ..., 1\} \\ \text{where} \quad |A| &= T \\ \text{Then} \quad \vec{A} &= \{50, 49.5051, ..., 1\} \end{aligned} \tag{3.11}$$

In another word, A_t can be calculated as a following:

$$\begin{aligned} A_t &= A_{begin} + \left(X_t \times \frac{A_{end} - A_{begin}}{T-1}\right) \\ \text{Then} \quad A_{begin} &= 50 \\ A_{end} &= 1 \end{aligned} \tag{3.12}$$

After considering the result of parameterised version of A-Average Case algorithm, Figure 4.5, we decide to deploy the *$(A = 50)$-Average Case* as a bidding strategy in exploration phase instead of *Aggressive Budget*[7] bidding policy, at line 4 in Figure 3.9. The main reason for deploying this is to avoid bidding too high price that made the agent out of budget afterward as he does in the Figure 4.3(a).

$$b^*_{t \in \{1,...,\epsilon\}} = \frac{B'_t}{A_t} \tag{3.13}$$

After changing the strategy in exploration, Figure 4.6, we also take this into the consideration working under exploitation. Finally arrive at the final improved version of *Greedy-Knapsack* with a maximum bound of bidding price, b_t^{max} in particular time step t integrating into Greedy-Knapsack exploitation phase, at line 7-8 in Figure 3.9, as following Equation 3.14.

[6] In MATLAB, this parameter can be calculated by *linspace* function.

[7] Aggressive Budget is the strategy that always place bid with the remaining budget for each auction.

$$
\begin{aligned}
b_t^* = \ & \arg \max_{b_1 \in \{1,...,b_t^{max}\}} \sum_{j=1}^{N} G_t(b_j) \\
\text{s.t.} \quad & \sum_{j=1}^{N} b_j = B_t' \\
\text{where} \quad & b_t^{max} = \frac{B_t'}{A_t} \\
& \vec{A} = \{50 + (X_t \frac{1-50}{T-1}), ..., 1\} \quad |A| = T \\
& b_j = \frac{B_t' - \sum_{\sigma=1}^{j-1} b_\sigma}{N-j+1}, \qquad j \in \{2, ..., N\}
\end{aligned}
\tag{3.14}
$$

Algorithm 7: Improved Greedy-Knapsack

Input : $B_t' > 0$, t, T, N, $G_t(x)$ and ϵ
Output: b_t^*

$\vec{A} = \{50 + (X_t \frac{1-50}{T-1}), \ ... \ ,1\}$ where $|A| = T$
if $t \leq \epsilon$ **then**
  Exploration Phase
  $b_t^* = \frac{B_t'}{A_t}$
else
  Exploitation Phase
  $b_t^{max} = \frac{B_t'}{A_t}$
  for $bid \in \{1, ..., b_t^{max}\}$ **do**
    $b_1 = bid$
    for $\sigma \in [2, ..., N]$ **do**
      $b_\sigma = \frac{B_t' - \sum_{\sigma=1}^{j-1} b_\sigma}{N-\sigma+1}$
    end
    $\vec{b}_{bid} = \{b_1, ..., b_N\}$
  end
  $\vec{b} = \{\vec{b}_1, ..., \vec{b}_{b_t^{max}}\}$
  Place a bid b_t^* according to Equation 3.14 given a set of $\vec{b}$
  Update $G_{t+1}(x)$ using MLE if $b_t^* \geq x_t$
end

FIGURE 3.9: Improved Greedy-Knapsack Algorithm

In next Chapter 4, we will evaluate our algorithms why they need to be improved. We also compare our algorithm with existing algorithms in the literature.

Chapter 4

Evaluation

In previous Chapter 3, we have developed our approaches: *Greedy*, *Greedy-Knapsack* and *Improved Greedy-Knapsack* algorithm thus far. In this chapter, we focus on the evaluation and performance of our algorithms compared to existing algorithms in the literature as we mentioned from the previous chapter. First, we will look at our performance of proposed greedy algorithms and how well they perform. After that, we will evaluate the greedy performance against other strategies in various settings according to our experiment, which are changeable:

Budget is a given parameter setting up to the agent when the simulation is being started , which can be varied between 500 to 5000 when the number of auction is fixed to 100, $T = 100$. The interesting in this setting is a bit challenge to the agent since he need to manage his budget well and also maximize the number of wins even though the budget is too small. See the result in Section 4.2.1.

Number of Auctions , also, is a given parameter to the agent when the simulation is being started. This parameter is varied from 50 to 300 according to our experiment, and we fixed the initial budget to 4000, $B = 4000$. This challenge is also interesting to see how well agent can manage his bidding under fixed budget when the number of the auction is either small or big number. See the result in Section 4.2.2.

Market Price Parameters are market price distribution's parameter consisting of 2 parameters: α and β. These parameters have directly influent to the algorithms since the market price will be increased when the parameters are set too high. In our work, we consider three settings of these parameters as following:

- $\alpha = 10$ and $\beta = 5$
- $\alpha = 11$ and $\beta = 9$
- $\alpha = 15$ and $\beta = 9$

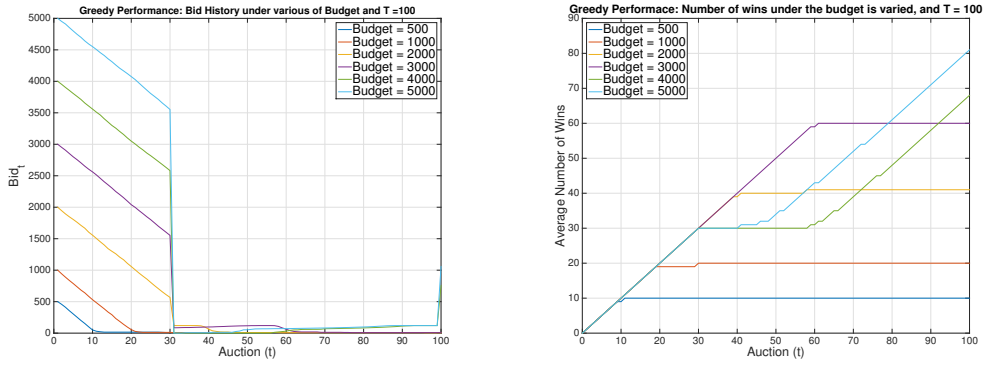
Since we have developed our approaches thus far from the previous Chapter 3, now we turn to performance aspect and evaluate our approaches in various settings such as budget, number of auction and different parameter of market price distribution. To be able to evaluate our approaches, we consider other strategies to compare with ours. The strategies are as following:

4.1 Greedy Algorithms Performance Analysis

In this section, we analyse the performance of our proposed algorithms to see how the agent performs using those strategies. After the analysis, we issue problems in each strategy then try to improve the strategies.

4.1.1 Greedy Algorithm

Given the Algorithm 5 above, we show the results in Figure 4.1. Interestingly, in this results, we observe that when the budget is less than or equal to 3000, he is out of the budget in the rest of auctions, in the middle of time horizon. We investigate that the main cause that made agent lost is because he spent too much money in exploration according to bid the budget, B'_t. Of course, this is bad news for this algorithm that will be solved in *Improved Greedy-Knapsack*, Section 3.4.3. Nonetheless, this algorithm is the simplest version of the greedy algorithm, and the result is not satisfied the number of wins that we are looking for since the results are not good enough. What we can see now in the simplest version of greedy is how to define the future bid $b_{j\geq 2}$, Equation 3.9. Since the future bids, $b_{t>1}$ are defined as a $N-1$-fraction of budget, That does not make any sense when the optimal bid is calculated by placing bid with highest accumulative probability for whole time horizon. As a result, improvement version is needed which we introduce in the next Section 3.4.2.



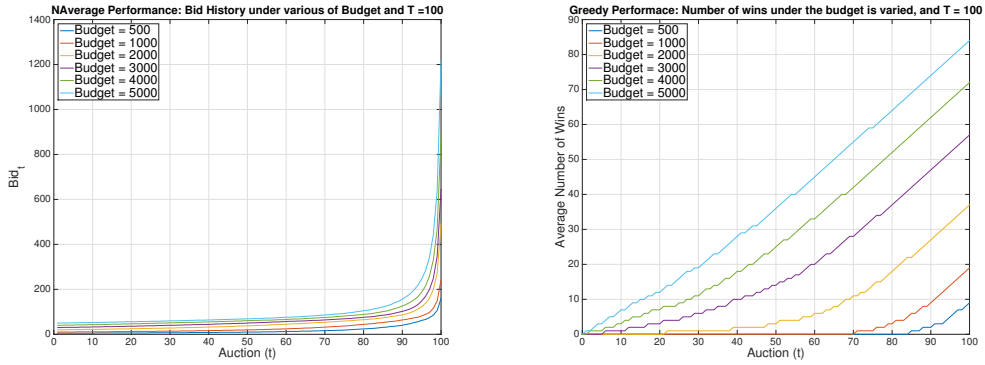
(a) Bid Performance

(b) Number of wins

FIGURE 4.1: Greedy Performance when budget is varied from 500 to 5000 with fixed number of auction T = 100

4.1.2 Greedy-Knapsack

Having mentioned in the Section 3.4.2, we use online knapsack problem solving technique proposed by Lueker (1998). The main reason of using the online knapsack problem is because of the performance is quite stable. There is no sign of out of budget as the original greedy does. Secondly, the number of wins under alternative budgets is also steady, always raised up. The last reason is the budget spent in N-Average case is also admirable, show in the Figure 4.2.



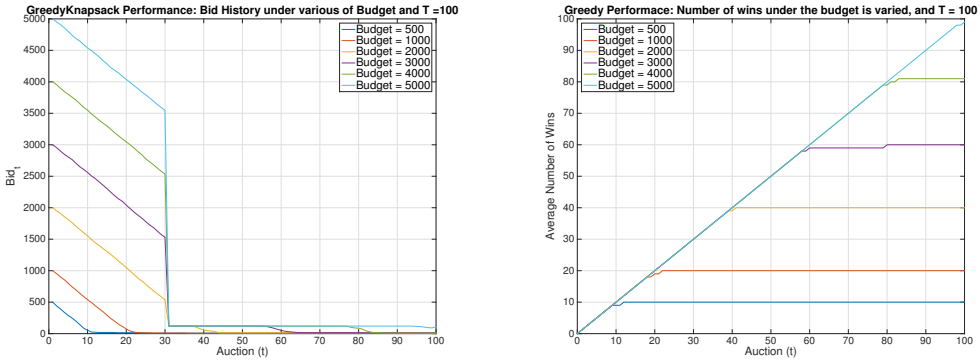
(a) Bid Performance

(b) Number of wins

FIGURE 4.2: N-Average Performance when budget is varied from 500 to 5000 with fixed number of auction T = 100

As we can see that N-Average case is simple strategy, the performance is steady as it should be. We feel like putting this as a part of future bid estimation does give a better result. Therefore, we assume that the performance of Greedy-Knapsack should be increased as well as the N-Average case does. In Figure 4.4, as we made an assumption, The performance of Greedy-Knapsack is promising, entirely performs better than the original greedy algorithm even the budget or number of auction is varied, according the Algorithm 6.

The results also show that the Greedy-Knapsack does better with 20 wins higher than the original version. To sum it up, we can say that changing the estimation equation of future bid influences the number of wins since in our the greedy algorithm we estimate the optimal bid b_t^* according to the accumulative probability in total of time horizon. However, we feel like this improvement is not sufficient in practice as we mentioned that, in exploration, the agent still spent too much budget which is need to be optimized. That leads to the next Section 3.4.3 - *Improved Greedy-Knapsack*: the last improved version of greedy algorithm with take maximum bound of bid price into account as a maximum bound of bid price, b_t^{max}, varies in a specified time step t.



(a) Bid Performance

(b) Number of wins

FIGURE 4.3: Greedy-Knapsack Performance when budget is varied from 500 to 5000 with fixed number of auction T = 100

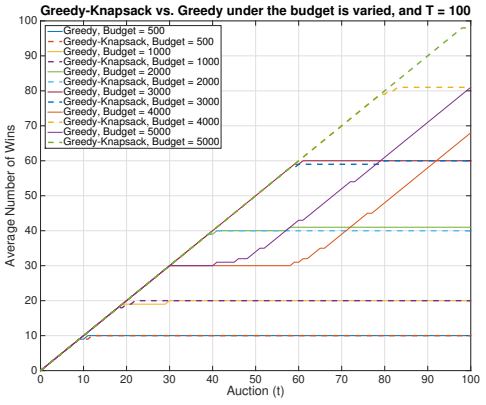
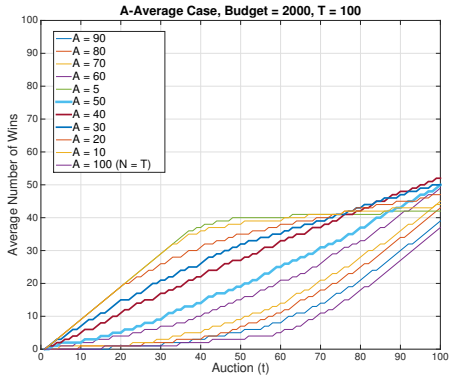


FIGURE 4.4: Greedy-Knapsack vs. Greedy

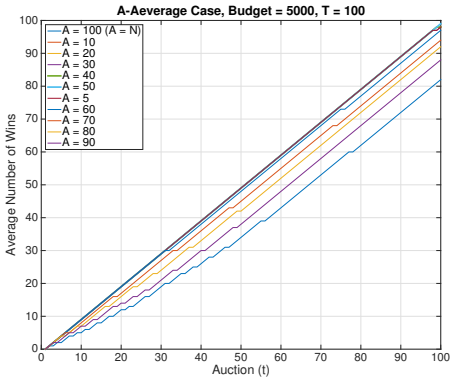
4.1.3 Improved Greedy-Knapsack

In the final version of greedy algorithm is motivated by the parameterised version of N-Average Case algorithm, whose results suggest that N-Average, where N is the number of auction remaining, is not the optimal value. Having said that we, consequently, create an experiment to see what is the optimal value of N, where the budget is varied from 500 to 5000, and the market price parameters are different.

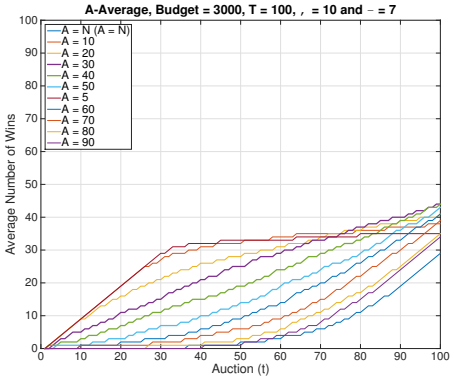
In Figure 4.5, the results suggest that $A = N$ is not an optimal value, as we mentioned above. When we varies this parameter A from 5, 10, 20,..., N, we found that, in a specific setting such as the budget is equal to 4000, the optimal of A, called A^*, is around 40, 50 and 60, however, 50 is the best value since the number of wins in overall always higher than both of 40 and 60. We also found that the number of wins, when the parameter A is equal to the optimal of specific settings, is considerably highest, and better than our Greedy-Knapsack. That is because he places the bid that does not exceed the fraction



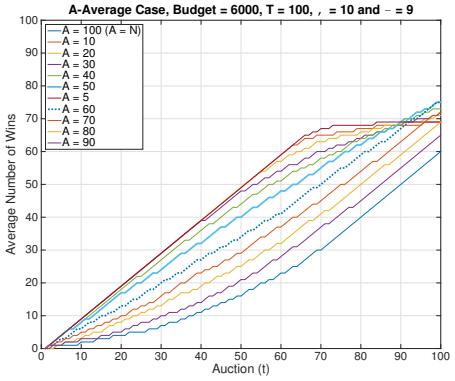
(a) Budget = 2000, T = 100, $\alpha = 10$, $\beta = 5$



(b) Budget = 5000, T = 100, $\alpha = 10$, $\beta = 5$



(c) Budget = 3000, T = 100, $\alpha = 10$, $\beta = 7$



(d) Budget = 6000, T = 100, $\alpha = 10$, $\beta = 9$

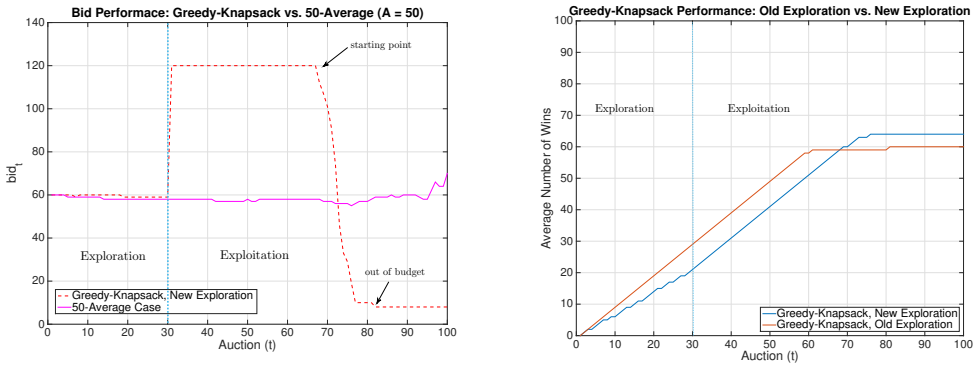
FIGURE 4.5: Parameterised A-Average Case in various situations

of the current budget B'_t. We can see, as a matter of fact, that the results above is completely stable when $A = A^*$.

As a matter of fact, we decide to change our bidding strategy in exploration into A-Average Case instead of using bidding the budget, which is an old bidding strategy, at all the time. The results show that agent does perform better than the former exploration strategy even, though, in exploration, he does not perform well as the previous one. In contrast, he can spare the budget for the remaining budget whose market price is lower than his bid and, of course, that increase the chance of winning in the rest of auctions.

Notwithstanding the increased chance of winning the auction, Figure 4.6(b), there are evidences of out of budget in this algorithm due to placing bid too high price in exploitation as we can see in Figure 4.6(a). The result shows that, after exploration, the agent always bid with the maximum bid price ($b^*_t = 120$) since its probability of winning bidding that price is always highest.

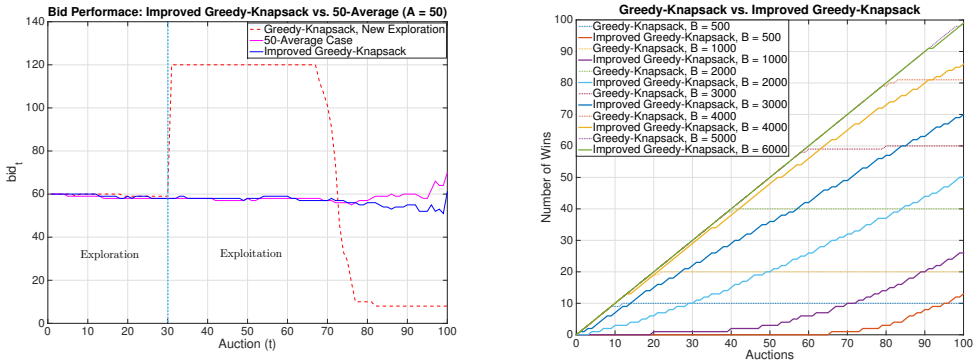
To be able to solve this problem, we decide to deploy A-Average Case on exploitation phase as well as in exploration. After we compare the results in Figure 4.7(a) between the previous Greedy-Knapsack and the new one, we can see the *Improved Greedy-Knapsack* does perform better for all various budget. Even the budget is too small, the agent is



(a) Bid Performance (b) Number of wins

FIGURE 4.6: Greedy-Knapsack Performance: Old Exploration vs. New Exploration under Budget = 3000 and T = 100

also well-organized the budget for the whole time horizon which is contrasted to the previous one that spent too much after exploration phase.



(a) Bid Performance (b) Number of wins

FIGURE 4.7: Improved Greedy-Knapsack Performance under Budget = 3000 and T = 100

In conclusion, we summarize that *Improved Greedy-Knapsack* is now suitable for our case with outstanding performance. This algorithm can also be calculated in polynomial time.

In next Section 4.2, we will compare our proposed algorithms with other existing strategies in the literatures under various of settings such as: budget is varied, number of auctions is varied, and the difference of parameters used in generating the market price.

4.2 Greedy vs. Existing Algorithms

In this section, we focus on how well of our strategies when competing with existing strategies that are mentioned in the Chapter 3. As we already stated that we consider

three parameters: budget, a number of auction and market price distribution parameters.

4.2.1 Budget

In this section we show the performance of our approaches under various budget, $B = \{500, 1000, 2000, 3000, 4000, 5000\}$ with other strategies mentioned above, except GPL bidding strategy, which is dynamic programming. We consider this as a special case since it cannot be calculated in polynomial time. For our analysis, we also consider other different settings of market price distribution's parameters α and β. We consider three difference of alpha and beta as following:

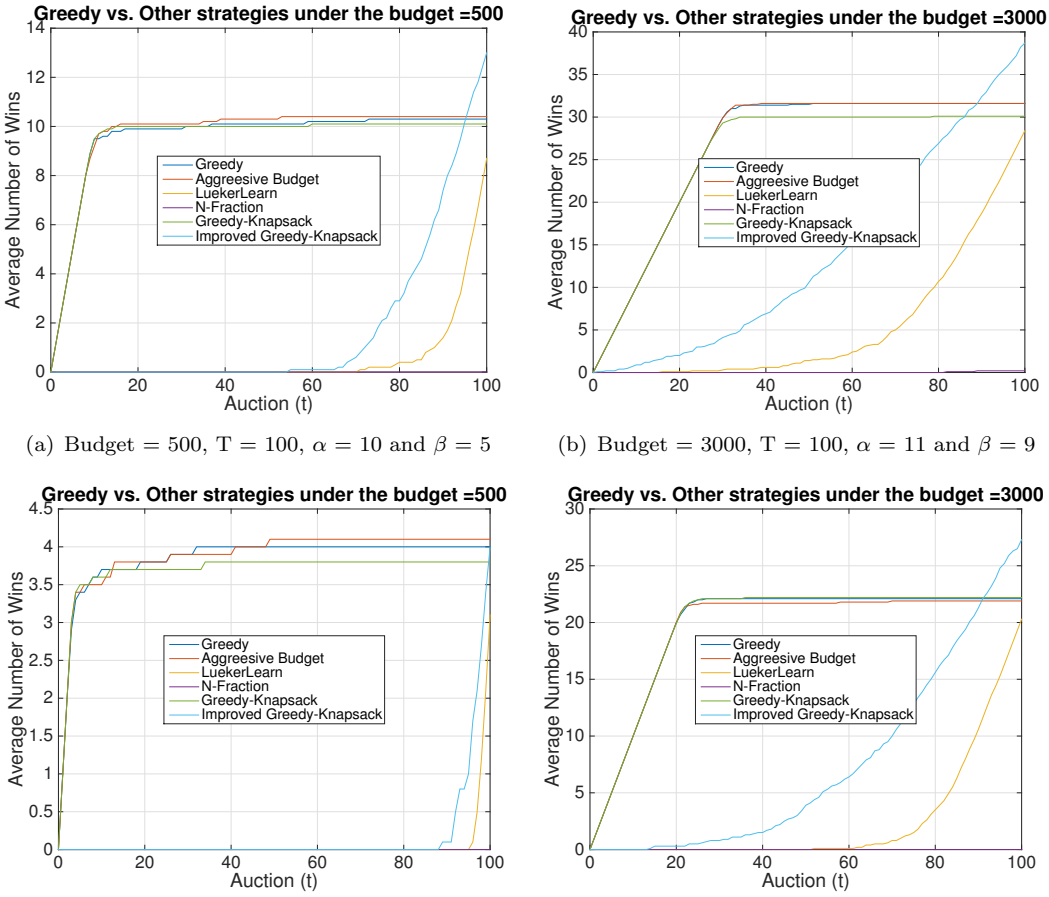
- $\alpha = 10$ and $\beta = 5$
- $\alpha = 11$ and $\beta = 9$
- $\alpha = 15$ and $\beta = 9$

For the results from those settings, in Figure 4.8, show that our *Improved Greedy-Knapsack* does perform very well in provided settings, even though the budget is varied. The point is that in the original *Greedy* algorithm and *Greedy-Knapsack* algorithm that both of them place bid at high price because the chosen bid price is highest accumulative probability for all possible bid in time horizon that is the best choice at a certain time step t. That is contradicted by the final version of the greedy algorithm that is no sign of out of the budget in the Improved Greedy-Knapsack strategy as, on the other hand, Greedy and Greedy-Knapsack perform. As a result, The result shows that *Improved Greedy-Knapsack* perform better than the first version of the greedy algorithm for at least 3% up to 18% in various settings. Totally, when we compare our algorithm with existing algorithm like LuekerLearn algorithm, we claim that our algorithms are entirely better performing against that LuekerLearn algorithm.

On the other hand, *Improved Greedy-Knapsack* does not perform well when budget is too low due to the fraction of budget defined as *50*-Average Case as it uses exploration and exploitation as a maximum bound of bid price. For that reason, it makes the bid is too low as the result will be shown in the Section 4.3, which we consider the market price parameters are a small number. More results of varying budget can be found in Appendix B.

4.2.2 Number of Auctions

Considering the various budget in the previous, it shows the performance of our *Improved Greedy-Knapsack* is outstanding when competes against other strategies. In this section,



(a) Budget = 500, T = 100, $\alpha = 10$ and $\beta = 5$

(b) Budget = 3000, T = 100, $\alpha = 11$ and $\beta = 9$

(c) Budget = 500, T = 100, $\alpha = 15$ and $\beta = 9$

(d) Budget = 3000, T = 100, $\alpha = 15$ and $\beta = 9$

FIGURE 4.8: Greedy Srategies vs. Other strategies in various budget and different parameters of market price distribution, on average number of wins.

we will focus on another aspect when the budget is fixed, $B = 4000$. We consider the number of auction is varied, in this case, to see how well of our approach perform in such environment when the number of auction is equal to, $T = \{50, 100, 150, 200, 250, 300\}$.

In Figure 4.9, the result show that all of the proposed algorithms perform well any number of auctions, and better than other algorithms in average. However, we can see that the ratio of the number of wins and the number of the auction is decreasing when the number of auctions is raised up. That is because the budget is become lower than the market price, as it show in the Figure 4.10. We define the lower and upper bound of market price to indicate that market price is usually drawn from this zone, which is 15 and 65 respectively. Besides, the mean line is the most market price that is usually going to be drawn.

For $T = 100$, we can see that the overall of placed bid is still higher the mean level. That means that agent has a high chance of win the auction. On the other hand, when $T = 300$, the result implies that the budget is being run out starting from $N \geq 160$. For this reason, the placed bid after that time is not going to win since the market price mostly

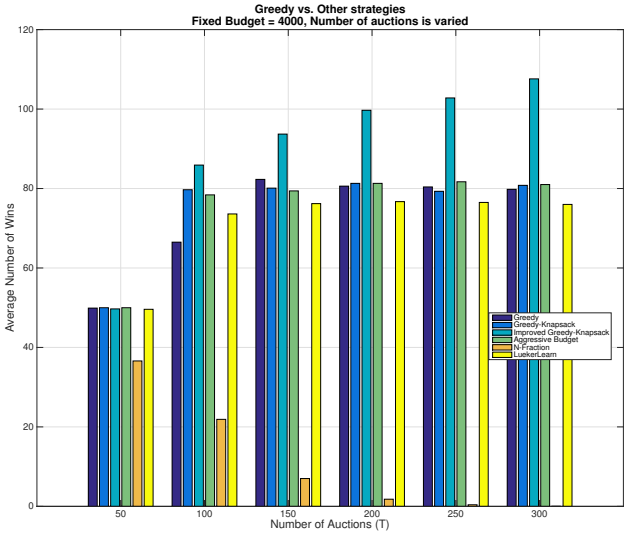


FIGURE 4.9: Greedy vs. Other Strategies when fixed budget and number of auction is varied, $\alpha = 10$ and $\beta = 5$, on average number of wins

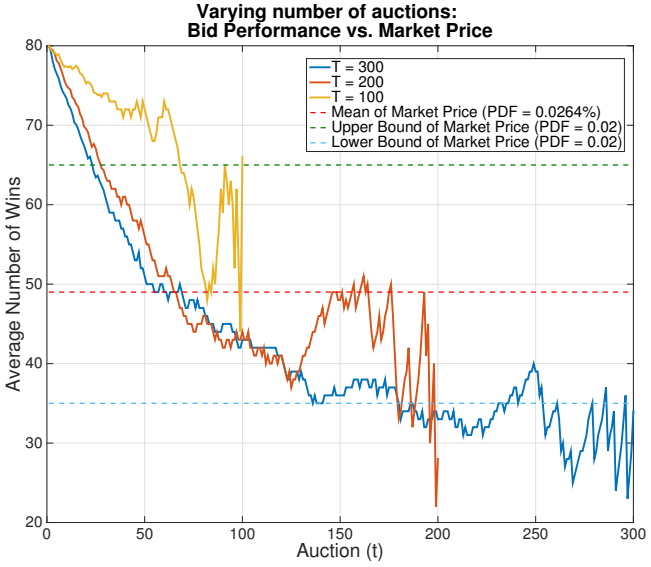
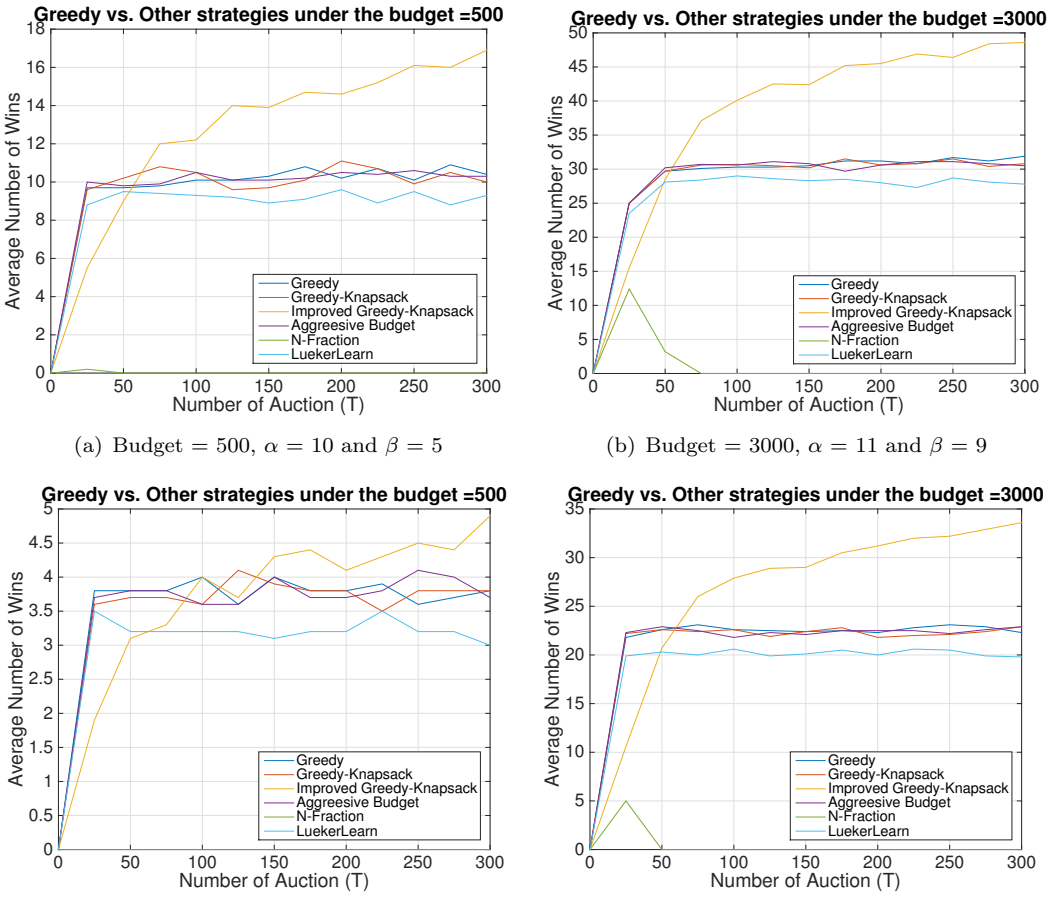


FIGURE 4.10: Comparison of bid performance when the number of auction is varied and given a fixed budget and number of auction is varied, $\alpha = 10$ and $\beta = 5$

higher than his bid, $x_t \geq b_t^*$. As a consequence, when the number of auction is 100, $T = 100$ the winning ratio is considerably high, $\frac{85}{100} = 0.85$ comparing to the winning ratio, $\frac{104}{300} = 0.346$, when the number of auction is set to 300, $T = 300$.

In Figure 4.11, we show the results that when the number of the auction, T, is varied from 50 to 300. The results show that *Improved Greedy-Knapsack* does perform very well in any settings although the given budget is too low. The trend of results is all the same in any settings. Surprisingly that *Greedy-Knapsack* algorithm also performs thus



(a) Budget = 500, $\alpha = 10$ and $\beta = 5$

(b) Budget = 3000, $\alpha = 11$ and $\beta = 9$

(c) Budget = 500, $\alpha = 15$ and $\beta = 9$

(d) Budget = 3000, $\alpha = 15$ and $\beta = 9$

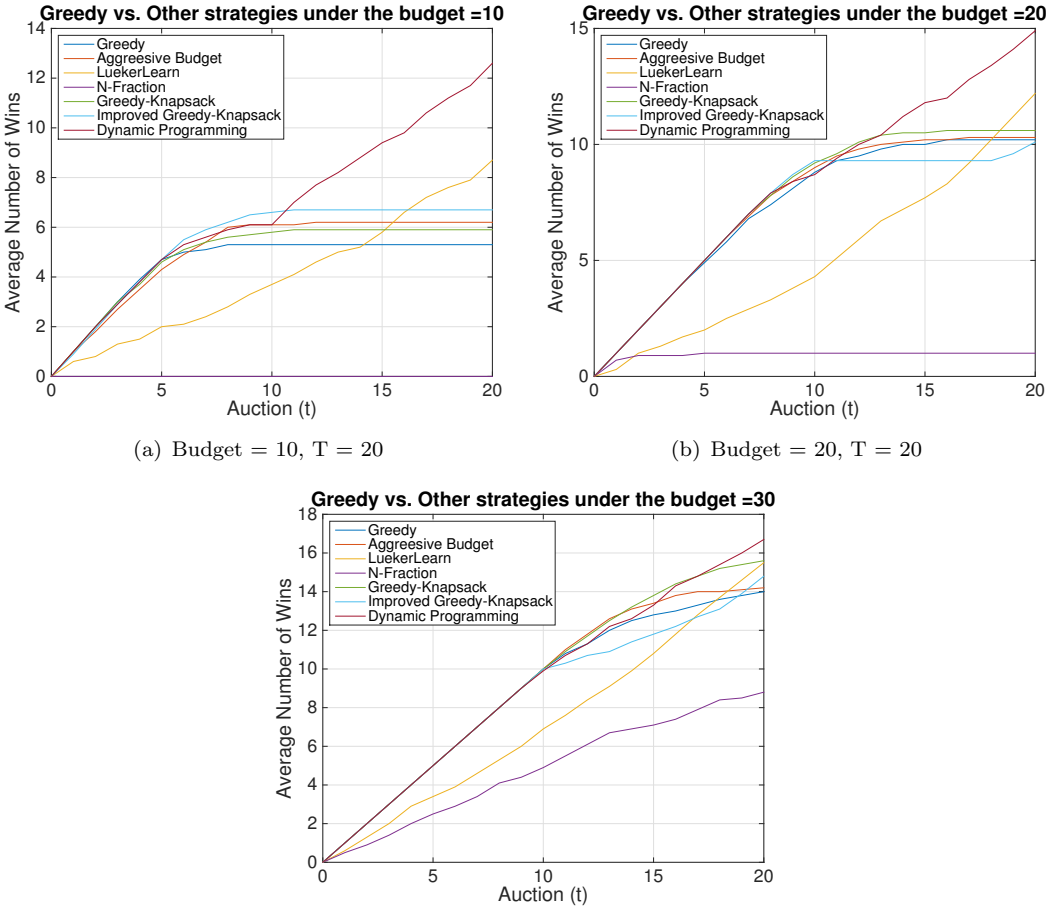
FIGURE 4.11: Greedy Srategies vs. Other strategies in the number of auction is varied and different parameters of market price distribution, on average number of wins.

well as similar as the improved greedy-knapsack algorithm. More results of the varying number of auction can be found in Appendix C.

4.3 Greedy vs. Dynamic Programming

In this section, we focus on a special case that when our algorithms are performing against dynamic programming, which is claimed as an optimal algorithm. In this experiment, we have to define the market price distribution's parameter at a very small number, $\alpha = 1$ and $\beta = 1.5$. Due to the estimation of optimal bid processing in the dynamic programming approach does take too long, as it has been proved (Tran-Thanh et al., 2014).

In Figure 4.12, as dynamic programming claimed, it is clearly that dynamic programming perform very well in any settings even the budget is varied comparing with our algorithms. The main reason is due to the use of 50-Average Case in exploration and exploitation as the maximum bound of the bid price. That makes the bid price estimated



(a) Budget = 10, T = 20

(b) Budget = 20, T = 20

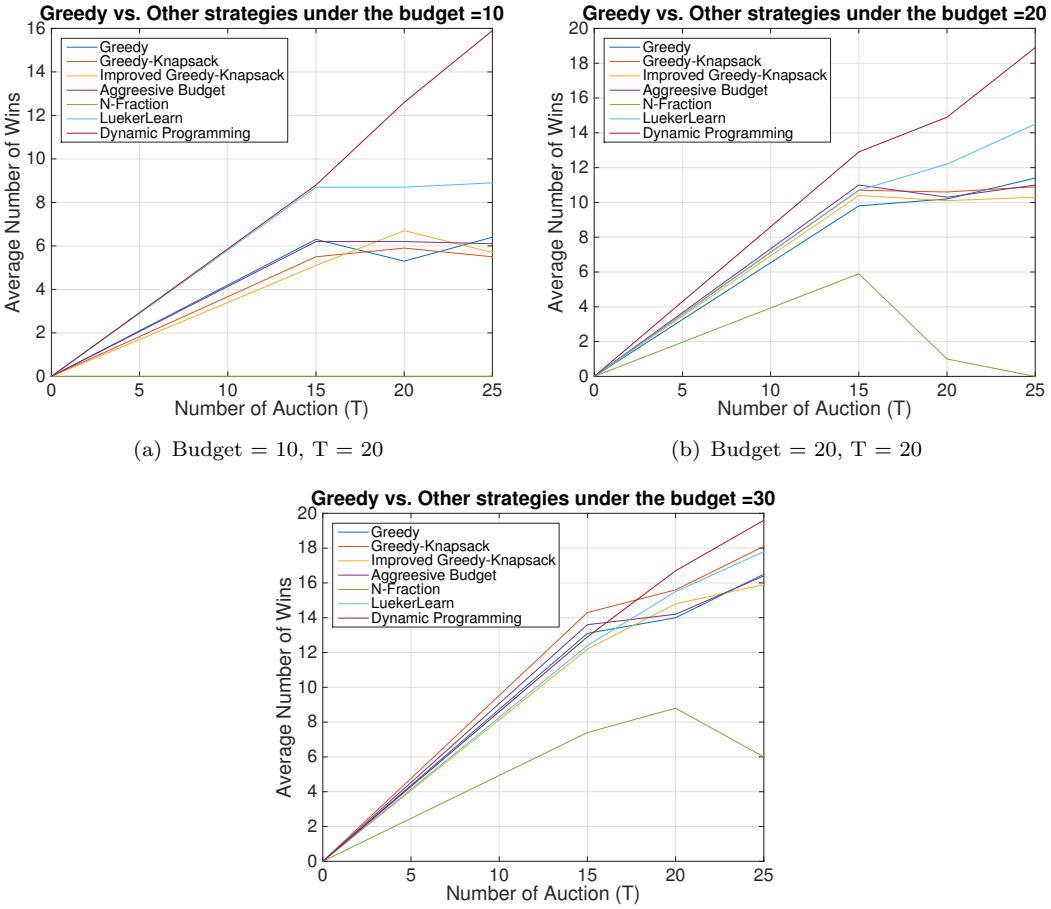
(c) Budget = 30, T = 20

FIGURE 4.12: Greedy Srategies vs. Dynamic Programming algorithm in various settings, $\alpha = 1$ and $\beta = 1.5$, on average number of wins

as such a smallest value, $b_t^* < 1$, and the market price usually drawn equal to 1, which is at highest probability, in Figure 3.1(b). Despite the better result, the processing time used in the simulation is too long as an exponential time. That is contrasted to other strategies that can be processed in polynomial time.

By the way, in Figure 4.13, it shows the performance of dynamic programming when the number of auctions is varied and given a fixed budget. The results show that dynamic programming algorithm performs the best in any settings. However, our approaches do not perform well as we mentioned the cause in the previous paragraph that the budget is too small. On the other hand, the *Greedy-Knapsack algorithm* does perform better than the *Improved Greedy-Knapsack* since it does not have a maximum bound of bid price capping the bid, but it still bid too high that why it does not beat the *LuekerLearn* algorithm. We find out that if we change the maximum bound of bid price parameter $\vec{A}$ to $\vec{A} = \{T + (X_t \frac{1-T}{T-1}), ..., 1\}$ (LuekerLearn's equation), the result will be better.

Finally, for the dynamic programming test case, the number of experiments is just only 10 round for the average results. We feel like that sometimes the results could be biased



(a) Budget = 10, T = 20

(b) Budget = 20, T = 20

(c) Budget = 30, T = 20

FIGURE 4.13: Greedy Strategies vs. Dynamic Programming algorithm in the number of auction is varied, on average number of wins.

since the market price distribution's parameters are a small number. For this case, we think that a 100-rounds would be making the result is more accuracy.

Chapter 5

Cost and Time Analysis

5.1 Cost

As we mentioned in the previous chapters that all of our works towards the completion of the bidding optimization dissertation were done in the simulation via MATLAB. In addition, the data we used in the simulation is generated by random function provided in MATLAB as well. No real data is used in our experiments.

5.2 Project Plan and Time Analysis

The completion of our work such as designing experiments, running experiments plus improvements and producing this dissertation spanned through a twelve week period. To be able to finish on time, a Gantt chart has been created as it shows in Figure D.1. This chart is to plan and monitor the progress of MSc Computer Science project. The Gantt chart provides the step-by-step from the beginning of research until the submission of written dissertation process that can be split into a four groups of process:

Background Research and Literature Review this process was started from week one studying the background of display advertising and existing works in the literature of bidding optimization whose paper investigating budget constraint in sequential auctions. This process was done in one week.

Simulation and Experiments design in this process, we mainly focus on designing the experiments and simulation. We started this process after the literature process was done from week two to week four. In actual progress, this process is still continued, since we customize it during the testing process.

Testing we extremely pay attention to this testing process from week 5 to week 10. The main activity of this process is to integrate our approaches and existing algorithms

in the recent studies into the simulation for generating results. After that, tune-up process was done on week nine to ten.

Disseration writing the dissertation is commenced on week nine, and the draft version was done on week eleven. On the last week, we spend this period on content corrections. After that, the written dissertation was submitted on Friday of the week twelve.

However, comparing the planned progress with actual progress is a bit different since we found the problem on the testing process, and it is necessary to have it fixed before writing the dissertation. That leads to the delay of our progress. The longest period belongs to the testing since we need to tune up the proposed algorithms and also fixed the errors come out from the simulation.

Chapter 6

Conclusions

6.1 Conclusion

Finally we arrive at the conclusion, in this dissertation, we have proposed our three heuristic algorithms, *Greedy*, *Greedy-Knapsack* and *Improved Greedy-Knapsack*, for the problem of sequential[1] second price auction under budget constraint given to the agent. The auction that we are interested in our dissertation is auction that only provides partial observation, for example, market price (second highest bid), to the winning agent. Besides, the market price in our model is drawn from the gamma distribution with fixed parameters.

After that, we model the problem and then proposed algorithms were tested in the simulation with random data drawing from the gamma distribution with three different set of parameters α and β. For each settings we also test the algorithms in various aspect: **(1)** budget is varied, and number of auction is fixed **(2)** number of auction is diverse and budget is fixed. The proposed algorithms clearly outperform in the simulation. We also compare our approaches with the existing algorithms like *LuekerLearn*, the result show that our algorithms are well-performed when competing with other algorithms. Since budget constraint and time constraint are already introduced, we, finally, feel that the last proposed algorithm *Improved Greedy-Knapsack* is suitable for our problem.

In the evaluation, we found our algorithms perform well in any settings, especially *Improved Greedy-Knapack* produces the best result. However, when we evaluated the particular case, we found our algorithms does not perform well when performing against dynamic programming, the literature claimed as an optimal bidding strategy. According to our results, we feel like that when the budget is low our algorithms do not perform very well, especially *Improved Greedy-Knapsack* algorithm when the budget is too low. However, when the budget is increased to infinity our algorithms produce the best result,

[1] Sequential Auction is also known as repeated auction

but still cannot beat the optimal strategy. Notwithstanding the best result produced by dynamic programming strategy, we feel like it cannot be used in practice, as it has been proved in the work of Tran-Thanh et al. (2014) that it took exponential time for estimating the optimal bid at a particular time step regarding the estimation of all possible future optimal bids.

6.2 Future Work

As we have researched sequential second price auction in our case, ideally speaking, we feel like the best improvement of defining the future bid is to use *Subset Sum technique*[2]. Since in our approach, we will looking for the vector that produce the highest accumulative probability given the market distribution in time horizon. Also, we feel like this is not correct way of using MLE as a parameter estimation (*gamfit*) provided by MATLAB. In fact, we do not know what the market price distribution of other bidders in the real situation that look like. We should assume that market price is drawn from the unknown distribution. For that reason, we think that using of Maximum-likelihood Estimation (MLE) that we deployed in the experiment for constructing the market price distribution is not correct. The correct way is presented in the literature by Tran-Thanh et al. (2014) that using Zeng's estimator to draw the distribution.

Another intestering aspect is number of auction. In reality, we do not know how many auctions held in one day due to the number of user serving on the Internet. This aspect is a bit chanllenge to our approach. We feel like the way that we can focus on is to define the targeting revenue agent should accomplish. That work is a bit chanllenge as a future work. The last aspect belongs to simultaneous auction since in the real situation there is not only a single auction took place at a particular time. There could be two more simultaneous auctions. The challenge of this aspect is finding how to manage the budget remaining in the environment that a number of the auction is randomly drawn from unknown distribution. Besides that agent has to participate in other sequential auction in time horizon.

[2]Sub Sum technique is way to find set of vectors that summation of vector is equal to a specified number. To be able to solve this problem, we have to adopt dynamic programming approach to solving this issue. Unfortunately, this approach is not satisfied the time constraint, since it requires at least B^{T-1}. That is impossible to process in our case.

Appendix A

Market Price Distribution

Having mentioned in the evaluation section 4 that we consider three differernce of parameters using in term of market price generating.

- Figure A.1: $\alpha = 10$ and $\beta = 5$
- Figure A.2: $\alpha = 11$ and $\beta = 9$
- Figure A.3: $\alpha = 15$ and $\beta = 9$
- Figure A.4: $\alpha = 1$ and $\beta = 1.5$

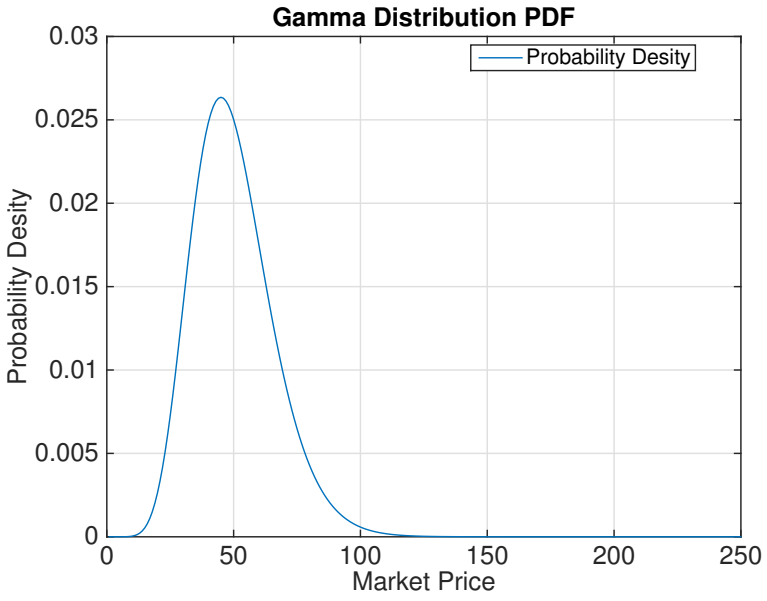


FIGURE A.1: Gamma distribution: Probability Desity Fucntion, $\alpha = 10$ and $\beta = 5$

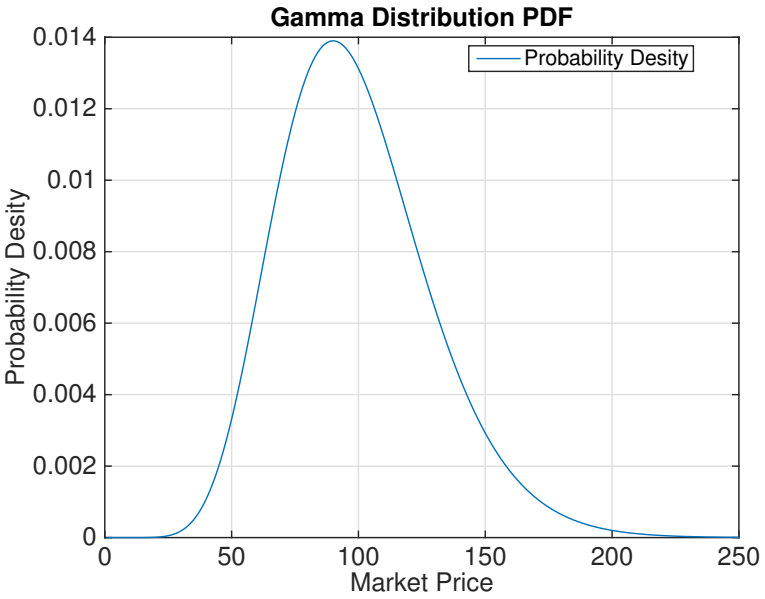


FIGURE A.2: Gamma distribution: Probability Desity Fucntion, $\alpha = 11$ and $\beta = 9$

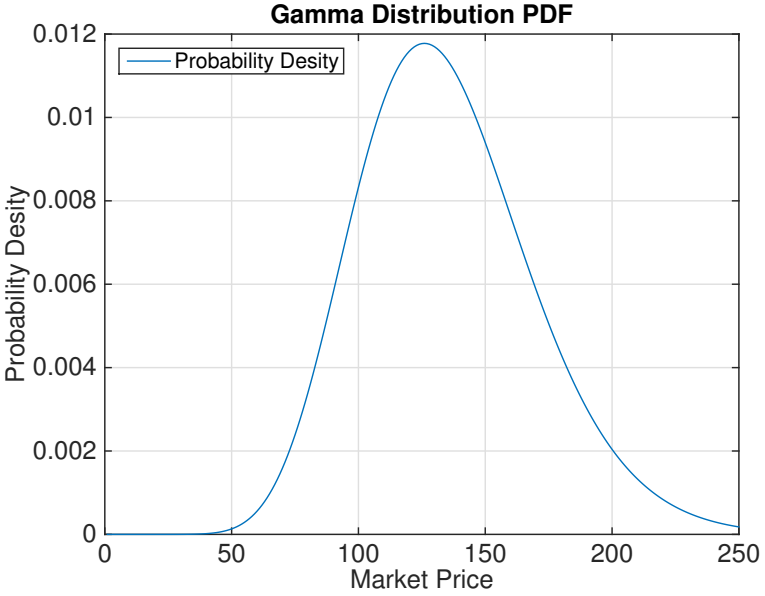


FIGURE A.3: Gamma distribution: Probability Desity Fucntion, $\alpha = 15$ and $\beta = 9$

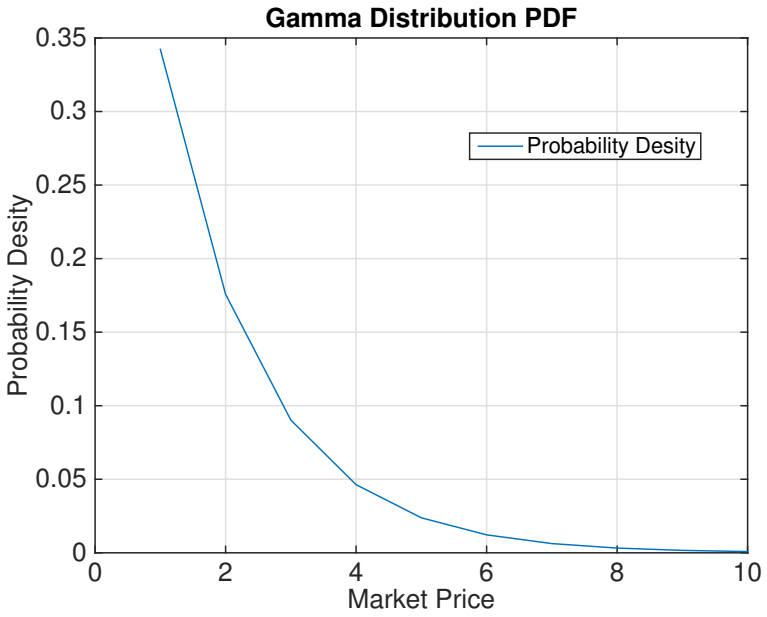


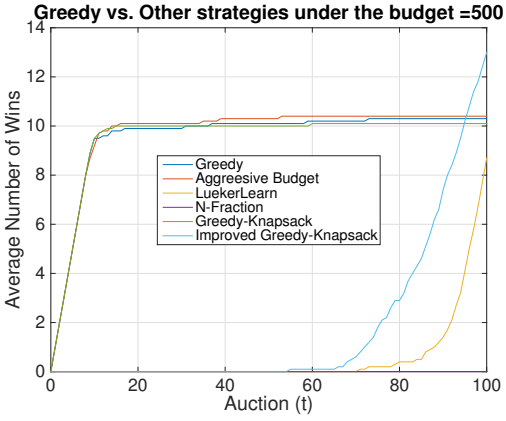
FIGURE A.4: Gamma distribution: Probability Desity Fucntion, $\alpha = 1$ and $\beta = 1.5$.

Appendix B

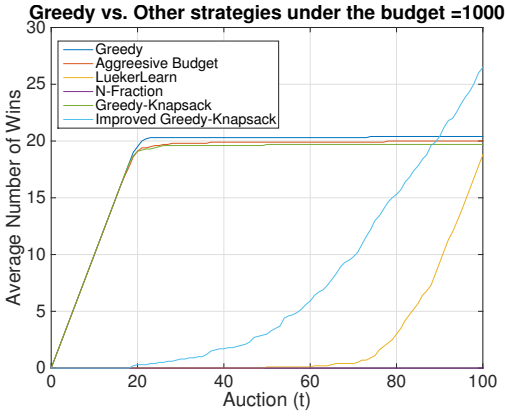
Results: Varying Budget

In this section, we run the experiment of various budget which is comprised of 6 different number of budget: 500, 1000, 2000, 3000, 4000, and 5000. Under the varying budget, we define number of auctions as a fixed number, $T = 100$, since we would like to know that the result is either changed or not if we change the budget. After that, we show the result of our algorithms comparing the dynamic programming algorithm.

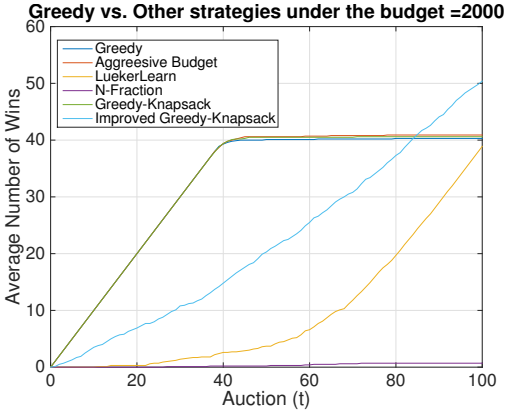
- Figure B.1: $\alpha = 10$ and $\beta = 5$
- Figure B.2: $\alpha = 11$ and $\beta = 9$
- Figure B.3: $\alpha = 15$ and $\beta = 9$
- Figure **??**: $\alpha = 1$ and $\beta = 1.5$, Greedy vs. Dynamic Programming



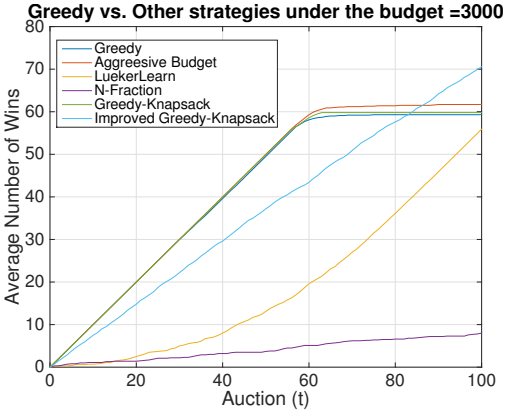
(a) Budget = 500, T = 100



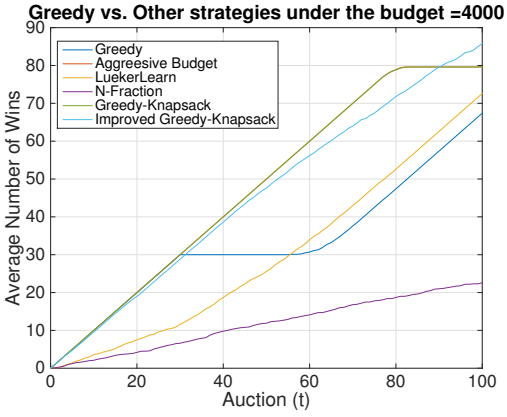
(b) Budget = 1000, T = 100



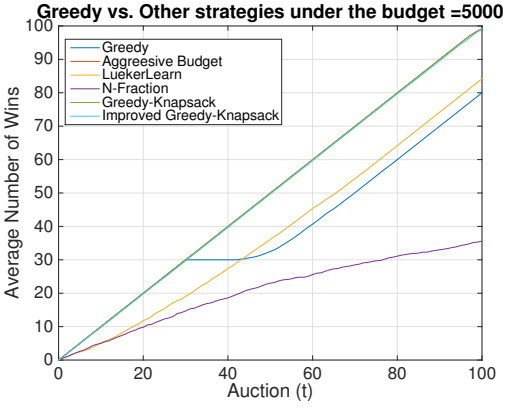
(c) Budget = 2000, T = 100



(d) Budget = 3000, T = 100

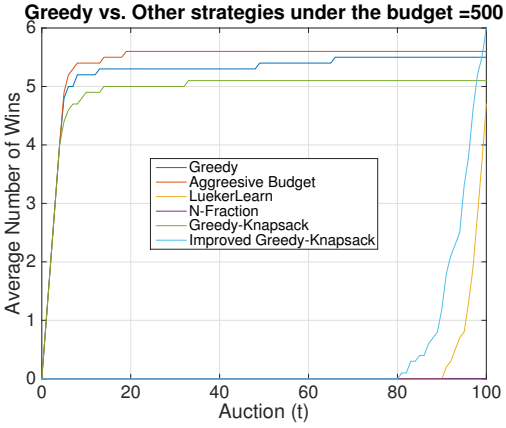


(e) Budget = 4000, T = 100

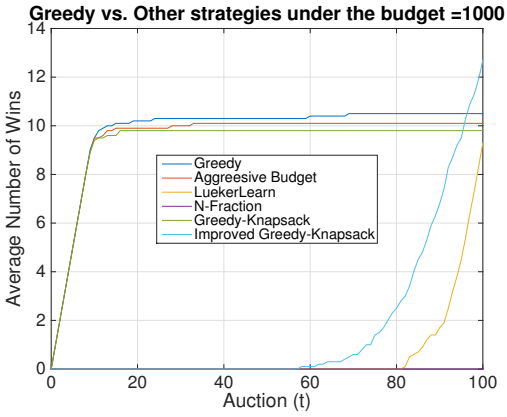


(f) Budget = 5000, T = 100

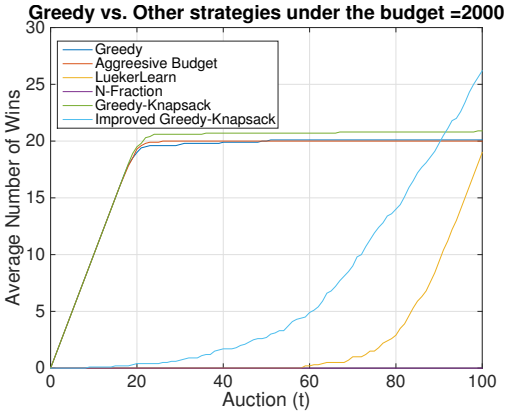
FIGURE B.1: Greedy Strategies vs. Other strategies, on average the number of wins, in various budget and market price is generated by $\alpha = 10$ and $\beta = 5$.



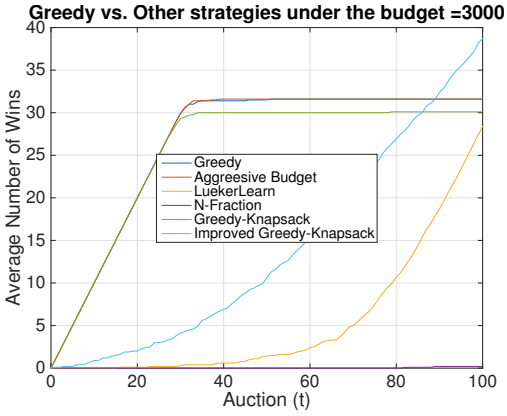
(a) Budget = 500, T = 100



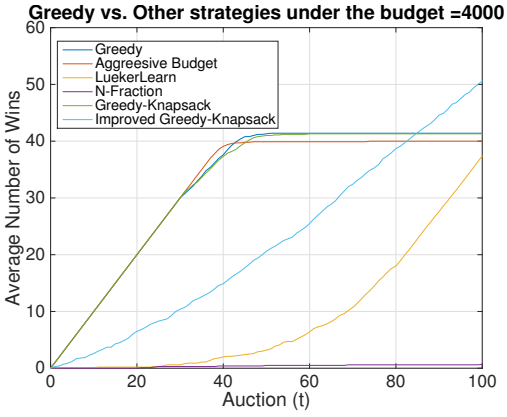
(b) Budget = 1000, T = 100



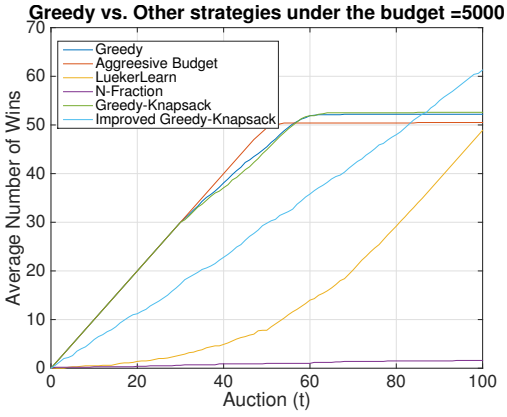
(c) Budget = 2000, T = 100



(d) Budget = 3000, T = 100

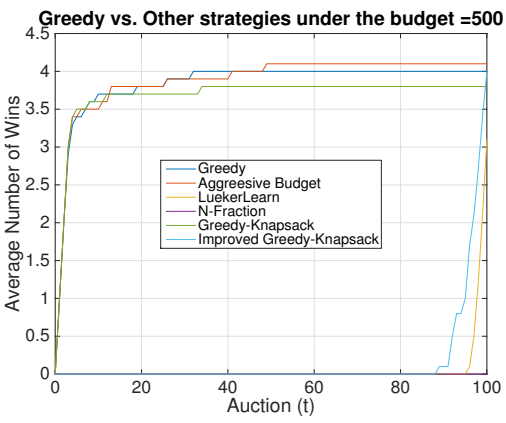


(e) Budget = 4000, T = 100

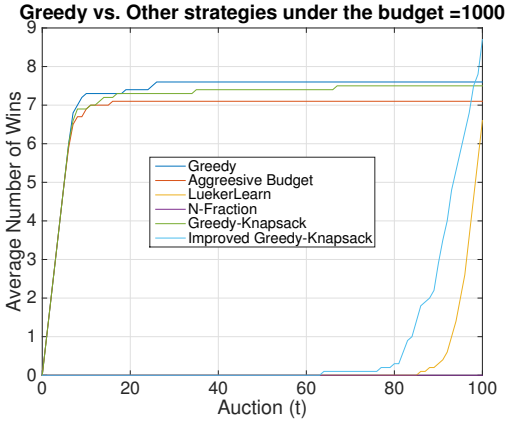


(f) Budget = 5000, T = 100

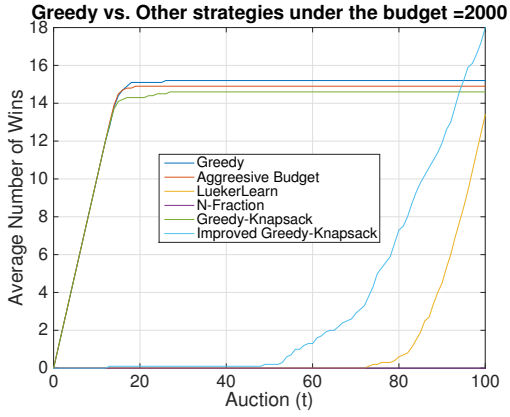
FIGURE B.2: Greedy Strategies vs. Other strategies, on average the number of wins, in various budget and market price is generated by $\alpha = 11$ and $\beta = 9$.



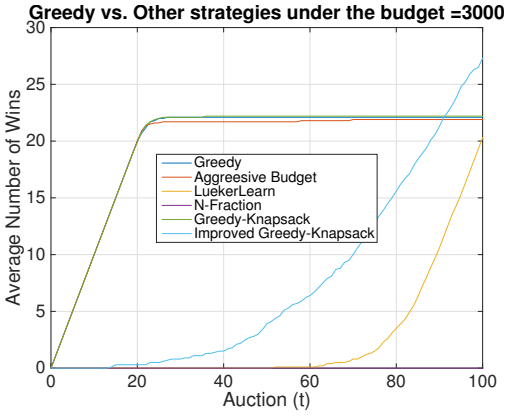
(a) Budget = 500, T = 100



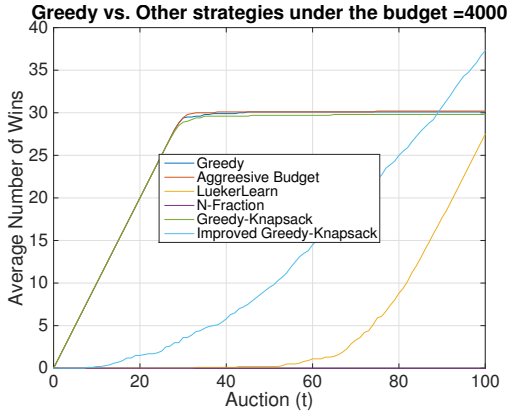
(b) Budget = 1000, T = 100



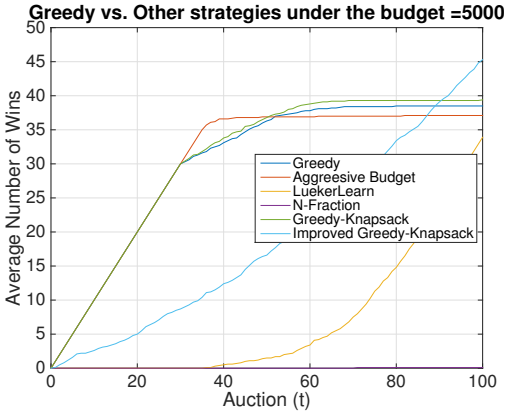
(c) Budget = 2000, T = 100



(d) Budget = 3000, T = 100

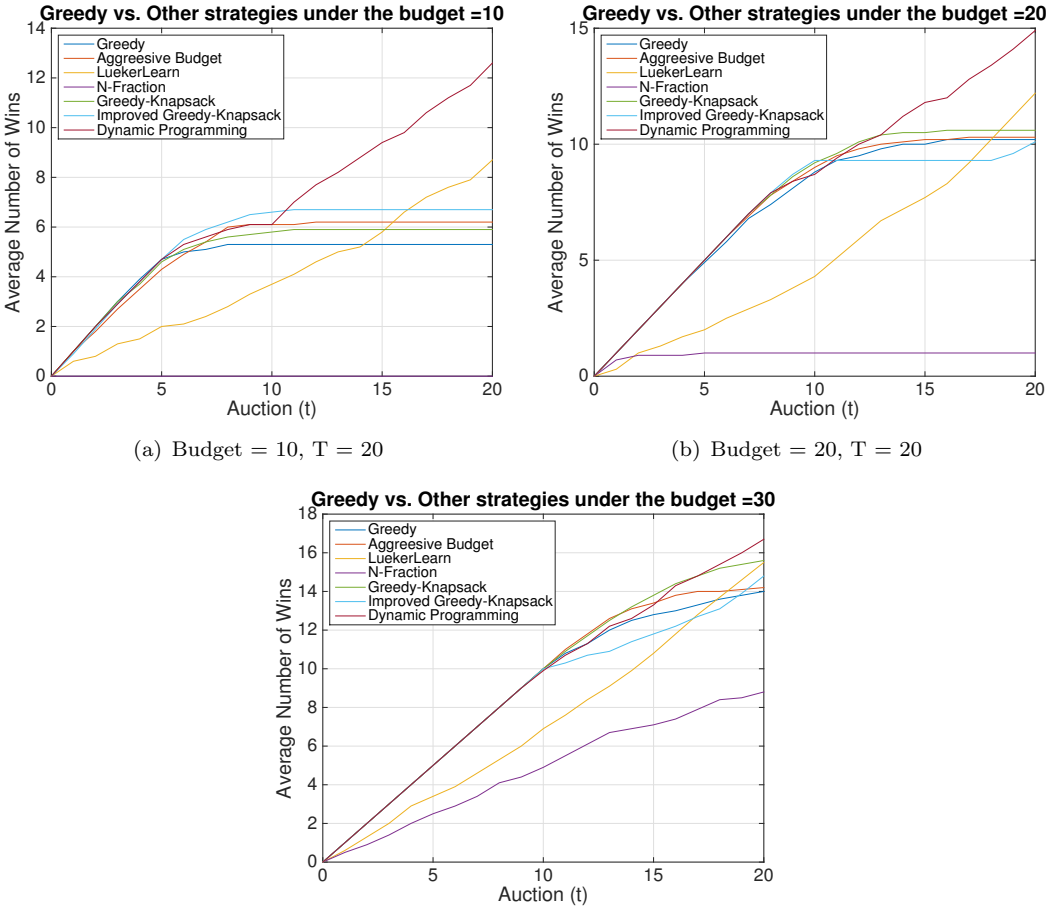


(e) Budget = 4000, T = 100



(f) Budget = 5000, T = 100

FIGURE B.3: Greedy Strategies vs. Other strategies, on average the number of wins, in various budget and market price is generated by $\alpha = 15$ and $\beta = 9$.



(a) Budget = 10, T = 20

(b) Budget = 20, T = 20

(c) Budget = 30, T = 20

FIGURE B.4: Greedy Strategies vs. Dynamic Programming, on average the number of wins, in various budget and market price is generated by $\alpha = 1$ and $\beta = 1.5$.

Appendix C

Results: Varying Number of Auction

In this section, we also run the experiment of varying number of auctions: 50, 100, 150, 200, 250 and 300 when the budget is fixed to 4000, $B = 4000$, when starting the experiment. After that, we show the result of our algorithms performing against the dynamic programming algorithm.

Fixed budget and varying the number of the auction.

- Figure C.1: $\alpha = 10$ and $\beta = 5$
- Figure C.2: $\alpha = 11$ and $\beta = 9$
- Figure C.3: $\alpha = 15$ and $\beta = 9$

Varying the number of auction and budget is also varied.

- Figure C.4: $\alpha = 10$ and $\beta = 5$
- Figure C.5: $\alpha = 11$ and $\beta = 9$
- Figure C.6: $\alpha = 15$ and $\beta = 9$
- Figure C.7: $\alpha = 1$ and $\beta = 1.5$, Greedy vs. Dynamic Programming

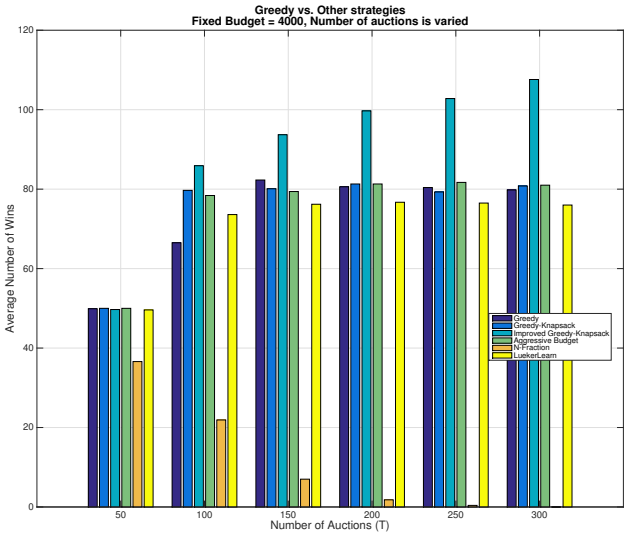


FIGURE C.1: Greedy vs. Other Strategies when fixed budget and number of auction is varied, $\alpha = 10$ and $\beta = 5$

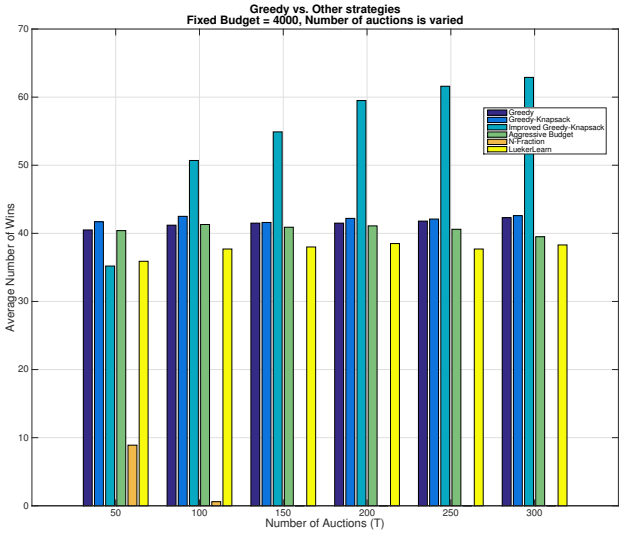


FIGURE C.2: Greedy vs. Other Strategies when fixed budget and number of auction is varied, $\alpha = 11$ and $\beta = 9$

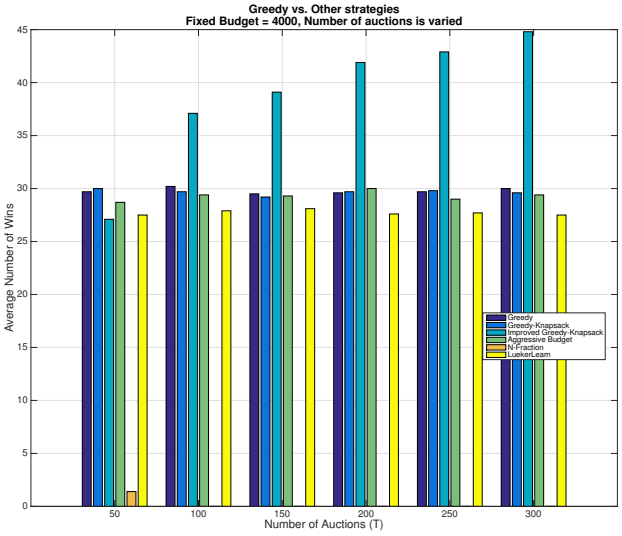
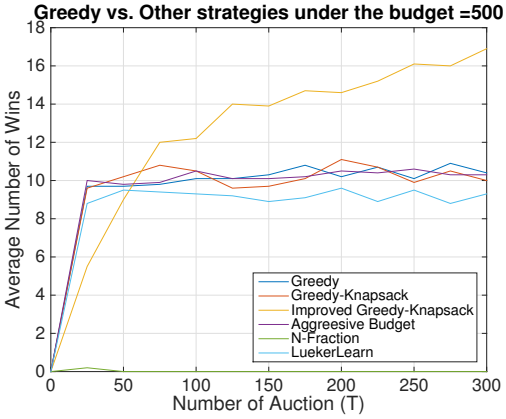
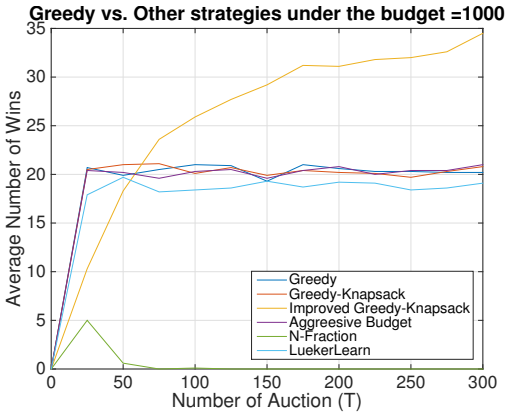


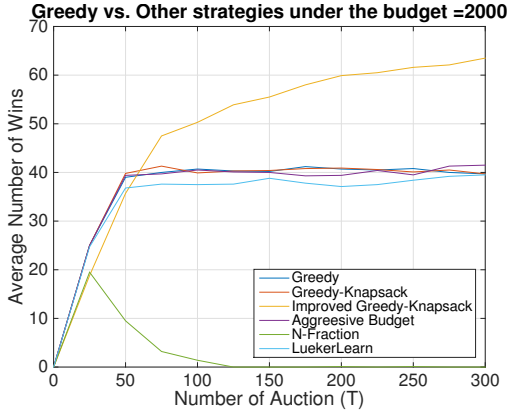
FIGURE C.3: Greedy vs. Other Strategies when fixed budget and number of auction is varied, $\alpha = 15$ and $\beta = 9$



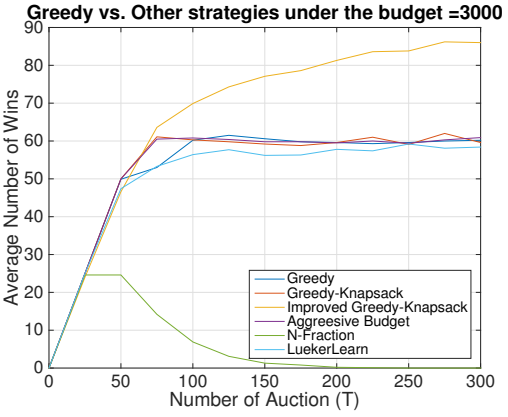
(a) Budget = 500



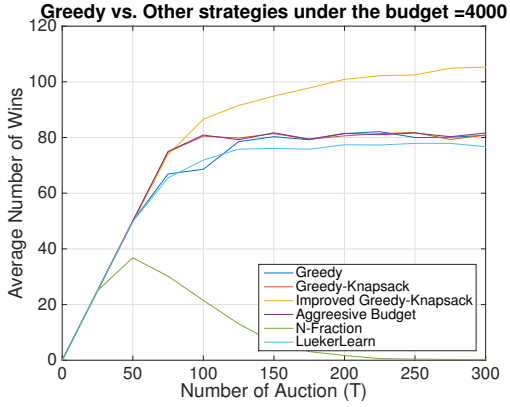
(b) Budget = 1000



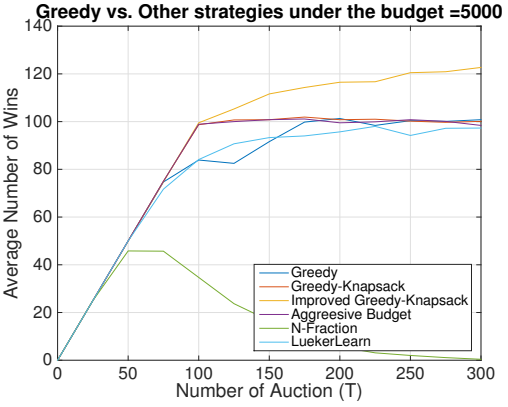
(c) Budget = 2000



(d) Budget = 3000

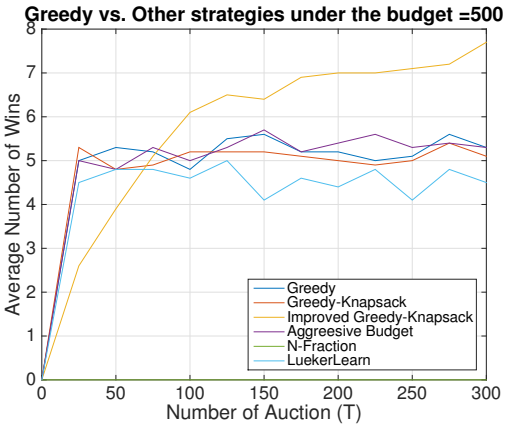


(e) Budget = 4000

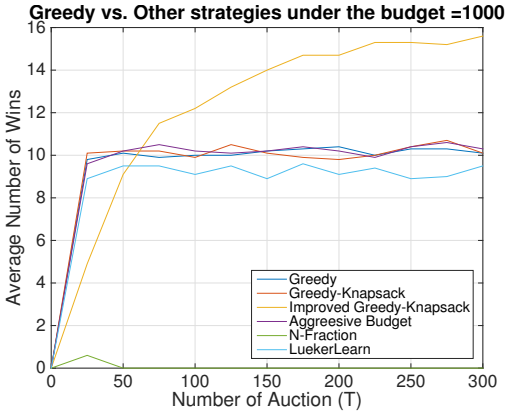


(f) Budget = 5000

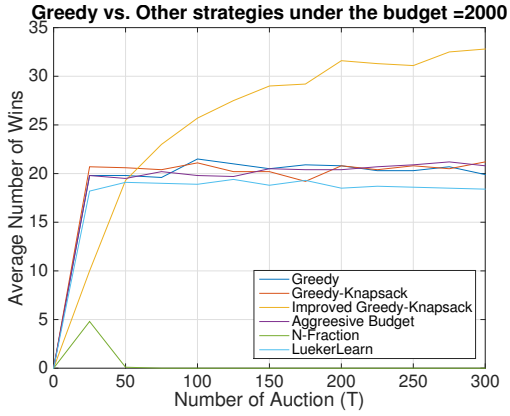
FIGURE C.4: Greedy Strategies vs. Other strategies, on average the number of wins, when the number of auction is varied and specified budget. The market price is generated by $\alpha = 10$ and $\beta = 5$.



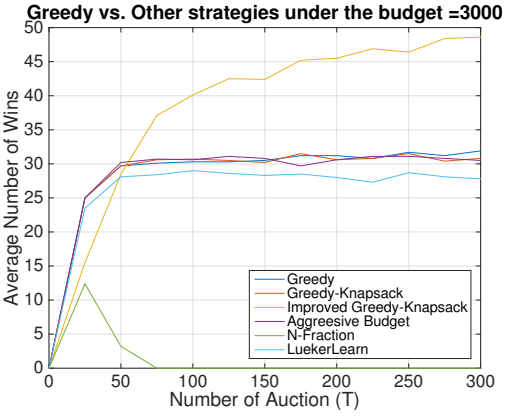
(a) Budget = 500



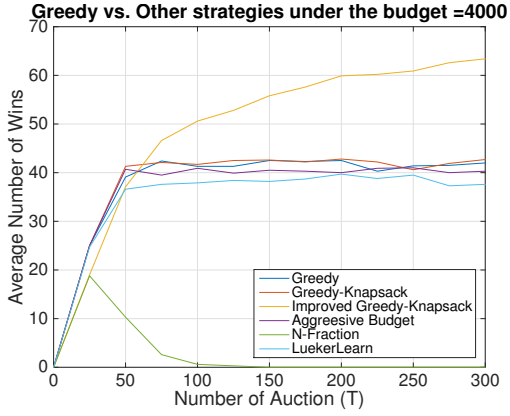
(b) Budget = 1000



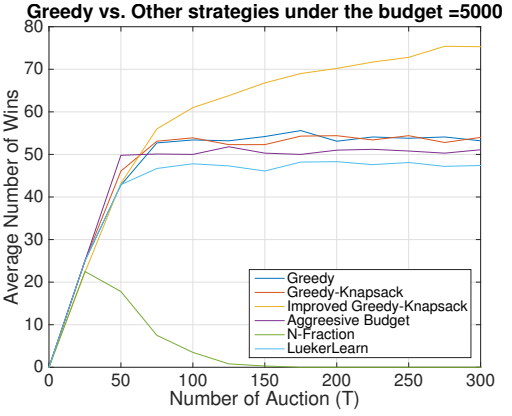
(c) Budget = 2000



(d) Budget = 3000

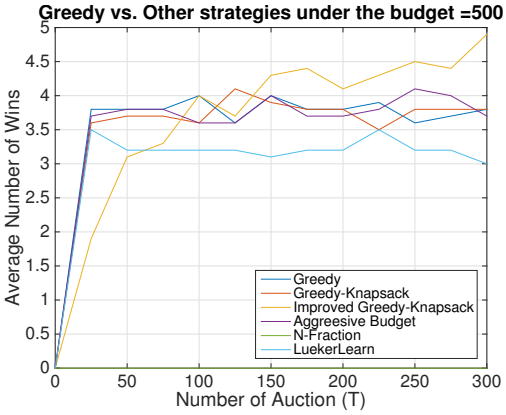


(e) Budget = 4000

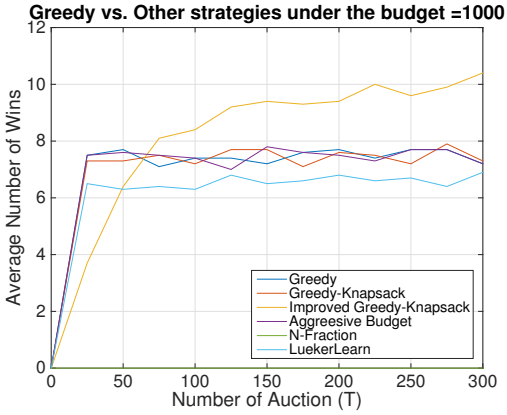


(f) Budget = 5000

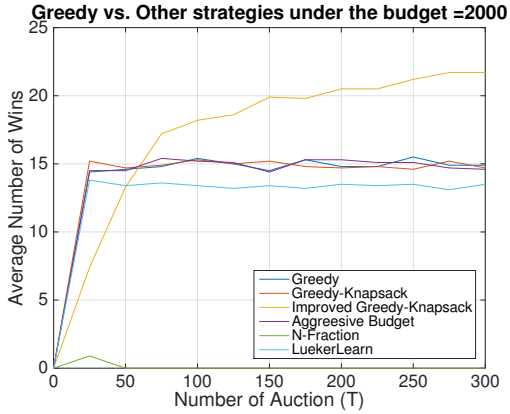
FIGURE C.5: Greedy Strategies vs. Other strategies, on average the number of wins, when the number of auction is varied and specified budget. The market price is generated by $\alpha = 11$ and $\beta = 9$.



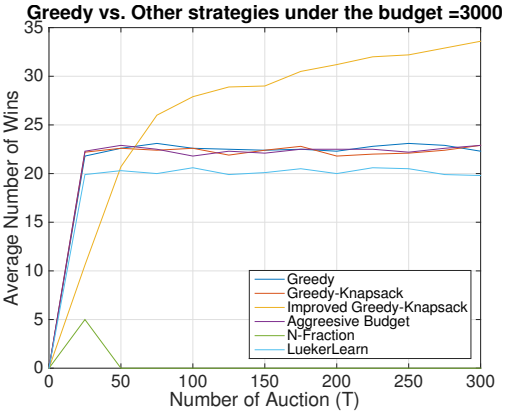
(a) Budget = 500



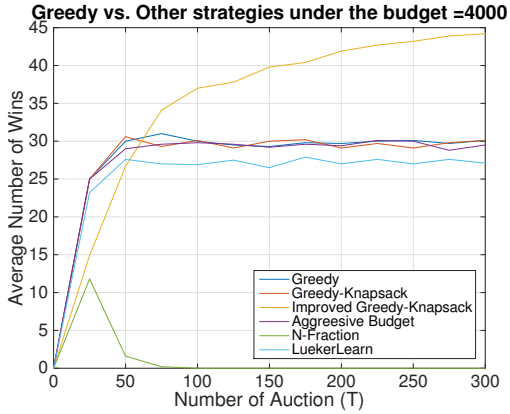
(b) Budget = 1000



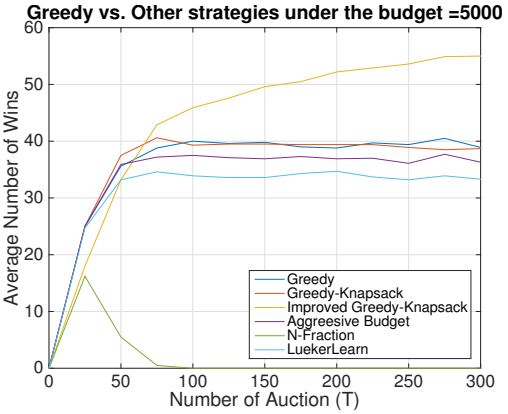
(c) Budget = 2000



(d) Budget = 3000

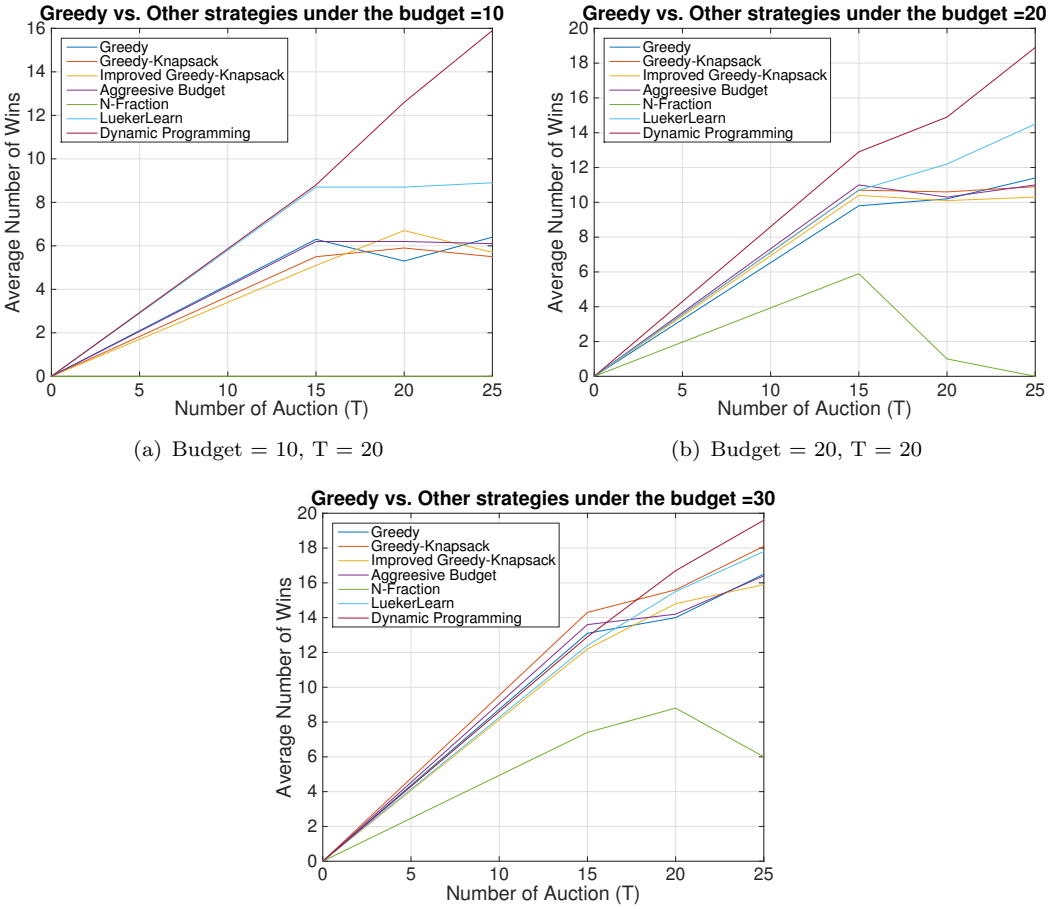


(e) Budget = 4000



(f) Budget = 5000

FIGURE C.6: Greedy Strategies vs. Other strategies, on average the number of wins, when the number of auction is varied and specified budget. The market price is generated by $\alpha = 15$ and $\beta = 9$.



(a) Budget = 10, T = 20

(b) Budget = 20, T = 20

(c) Budget = 30, T = 20

FIGURE C.7: Greedy Strategies vs. Dynamic Programming algorithm in the number of auction is varied and different parameters of market price distribution, on average number of wins.

Appendix D

Project Plan

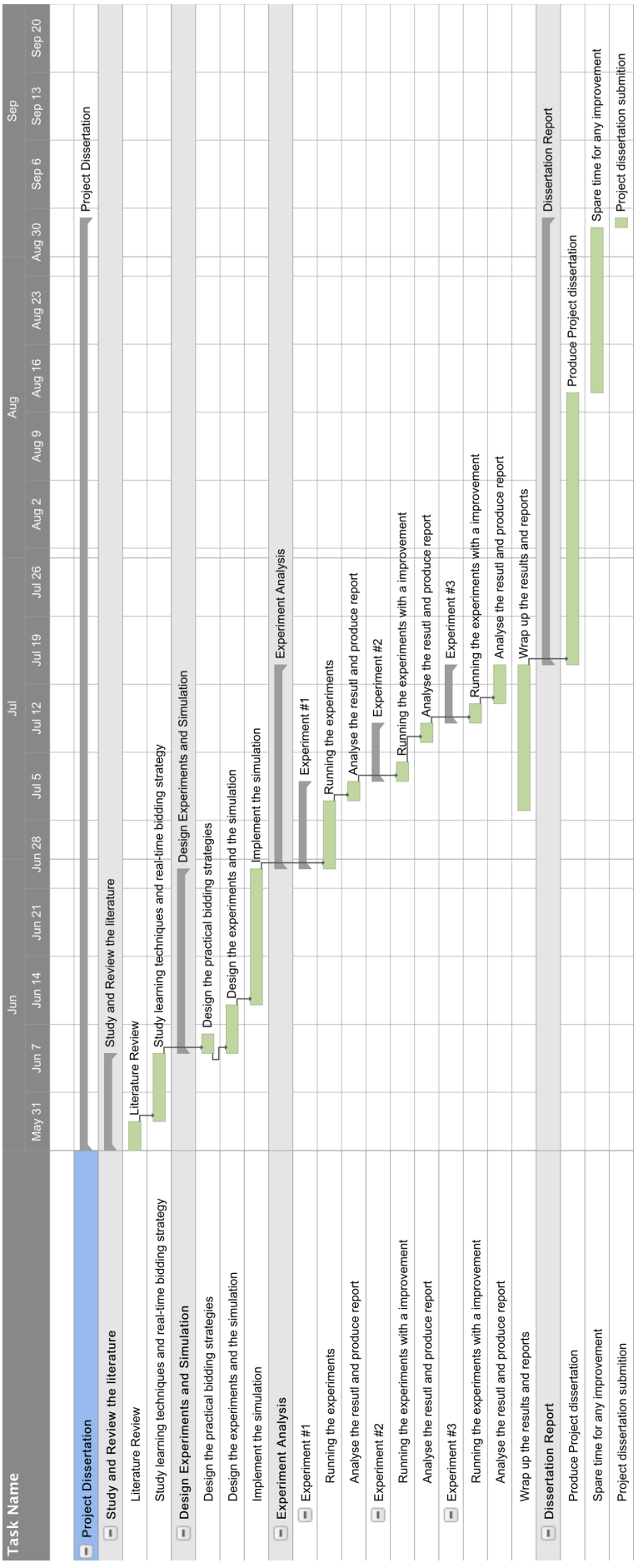


FIGURE D.1: Planned Progress Gantt Chart

Bibliography

Kareem Amin, Michael Kearns, Peter Key, and Anton Schwaighofer. Budget optimization for sponsored search: Censored learning in mdps. *arXiv preprint arXiv:1210.4847*, 2012.

Dimitris Bertsimas, Jeffrey Hawkins, and Georgia Perakis. Optimal bidding in online auctions. *Journal of Revenue & Pricing Management*, 8(1):21–41, 2009.

Wenkui Ding, Tao Qin, Xu-Dong Zhang, and Tie-Yan Liu. Multi-armed bandit with budget constraint and variable costs. In *AAAI*, 2013.

Gamma Distribution. Gamma distribution — Wikipedia, the free encyclopedia, 2015. [Online; accessed 17-Auguest-2015].

Gamma Parameters Estimation. Gamma parameters estimation — matlab gamfit — mathwork united kingdom, 2015. [Online; accessed 19-Auguest-2015].

Arpita Ghosh, Benjamin IP Rubinstein, Sergei Vassilvitskii, and Martin Zinkevich. Adaptive bidding for display advertising. In *Proceedings of the 18th international conference on World wide web*, pages 251–260. ACM, 2009.

Ramakrishna Gummadi, Peter Key, and Alexandre Proutiere. Repeated auctions under budget constraints: Optimal bidding strategies and equilibria. *SSRN working draft*, 2012.

Vijay Krishna. *Auction theory*. Academic press, 2009.

George S Lueker. Average-case analysis of off-line and on-line knapsack problems. *Journal of Algorithms*, 29(2):277–305, 1998.

S Muthukrishnan. Ad exchanges: Research issues. In *Internet and network economics*, pages 1–12. Springer, 2009.

Sandip Sen and Anton Ridgway. Adaptive algorithms for fixed-cost multi-armed bandit problems with budget constraints.

Long Tran-Thanh, Lampros C Stavrogiannis, Victor Naroditskiy, Valentin Robu, Nicholas R Jennings, and Peter Key. Efficient regret bounds for online bid optimisation in budget-limited sponsored search auctions. 2014.

Ioannis Vetsikas et al. Sequential auctions with budget-constrained bidders. In *e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on*, pages 17–24. IEEE, 2013.

Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In *Internet and Network Economics*, pages 566–576. Springer, 2008.