AA229/CS239: Decision Making under Uncertainty

Multilevel Monte Carlo Tree Search

Victoria M. Dax

***Abstract*— "Buy low, sell high" is a famous investing adage about taking advantage of the market's propensity to overshoot on the downside and upside. The volatile behavior of stock market prices is an ideal example for an MDP with rare events thht correlate with exceptionally high rewards or penalties. In the area of computational finance, these undulations are often modeled using stochastic processes. Monte Carlo methods are very general and useful approaches for estimating expectations arising from stochastic simulations. However, we not only intend to model them, but to leverage these fluctuations and solve for profitable investment strategies. We will introduce two modifications to the MCTS. To improve its performance on the task, we will use a Multilevel Monte Carlo approach for sampling and, further, append the simulation step of standard MCTS by the splitting method.**

I. Introduction

Monte Carlo methods find application uses in various domains, including solving sequential decision problems with uncertainty in the outcome of an action, which are commonly modeled as Markov decision processes (MDP). If there further is uncertainty in the current state of the environment such problem can be solved as partially observable MPDs (POMDP). Highly dimensional or continuous MDPs and POMDPs are thus often solved using online sampling based algorithms, among which Monte Carlo Tree Search is one of the most successful methods. It finds approximately optimal actions by iteratively building a search tree using Monte Carlo simulations. However, if a catastrophic event occurs, the agent receives a large negative reward associated with it. Assuming the event is rare, the estimate will converge very slowly leading to poor performance. The matter of interest is improving efficiency and performance of MCTS in handling reward structures with rare events. For this matter we will implement a recently developed method, namely Multilevel Monte Carlo (MLMC), for variance reduction. MLMC can be used for sampling and simulating and our results with show a significant increase in performance of MCTS.

Online sampling based solver for POMPDs and Monte Carlo methods has been studied in distinct research areas. While [1] presents a more concise description of Monte Carlo algorithms applied to POMDPs, [2] provides a broad overview of computational methods for decision making problems in general. However, in Monte Carlo methods are as well studied independently in the other domains, such as applied statistics and computational finance for example. Monte Carlo methods and examples are discussed in [3] and various extensions for more specific application uses have been researched too, [4], [5] and [6].

One of these extensions is Multi-Level Monte Carlo performing variance reduction on estimates, [7]. This approach is today mainly researched at the Mathematical Institute at Oxford University.

Within the scope of solving MPDs and POMDPs, this paper presents a variation of Monte Carlo Tree Search (MCTS) and discusses and application example. To better account for rare events in decision making, two main modifications are proposed: First, the Multilevel extension for Monte Carlo methods is introduced to perform variance reduction on the reward estimate. Then, these benefits are leveraged for improvements in MCTS through the splitting method.

A general MLMC solver - which can as well be used for solving SPDEs - has been written for this project. All code was implemented in Julia and is publicly available, along with other examples, at `https://www.github.com/tory-lena/MLMC_method.git`

II. Method

MCTS involves collecting statistics about the expected accumulation of reward after a series of actions. If catastrophic events occur in a problem, sampling may result in large negative rewards. Rare large negative rewards can make the reward estimates converge slowly, leading to poor performance. In the context of MCTS, general variance reduction techniques such as antithetic sampling, conditioning or stratification are often applied. We will focus on a particular kind of variance reduction technique, which is based on the control variates approach and discuss its application to tree search, the splitting method.

A. Multilevel Monte Carlo

Control variates is one of the classic approaches for variance reduction. For a control variate g, with known estimate $\mathbb{E}(g)$, which is well correlated to f, it is feasible to compute an unbiased estimator

$$\mathbb{E}(f) = \frac{1}{N}\sum_{n=1}^{N}\Big\{f(x^{(n)}) - \lambda\big(g(x^{(n)}) - \mathbb{E}(g)\big)\Big\}. \quad (1)$$

To minimize the variance by a factor of $(1-\rho^2)$, the optimal λ is $\lambda^* = \rho\sqrt{\mathbb{V}(f)/\mathbb{V}(g)}$, where ρ is the correlation between f and g. Multi-Level Monte Carlo is an adaption of this method. For L levels, the estimate is

$$\begin{aligned}\mathbb{E}(P_L) &= \mathbb{E}(P_0) + \sum_{l=1}^{L} \mathbb{E}(P_l - P_{l-1}) \\ &= \frac{1}{N_0}\sum_{n=1}^{N_0} P_0^{(l=0,n)} + \sum_{l=1}^{L}\left\{\frac{1}{N_l}\sum_{n=1}^{N_1} P_l^{(l,n)} - P_{l-1}^{(l,n)}\right\},\end{aligned} \tag{2}$$

which leads to a variance of the estimate of $\sum_{l=1}^{L}\frac{\sigma_l^2}{N_l}$.

However, in many infinite-dimensional applications the output P_l on level l is an approximation to a random variable P - referred to as the profit or reward function - which cannot be simulated exactly. So, how to choose the finest level of accuracy? One can base this choice upon desired accuracy, using the optimal number of samples on each level based on an estimate of the variance or use a fixed number of total samples and then perform a simulation at level l with probability p_l. Among other methods, these three are described in [6].

We will for all subsequent simulations use a geometric approach, i.e. the construction of multilevel estimators uses a geometric sequence of grids and usual estimators for $P_l - P_{l-1}$. Alg 1 outlines pseudo code explaining the algorithm implemented in the main routine.

Algorithm 1 MLMC Algorithm

```
1: Set $L = 2$ sample $N_0$ on levels $l = 0, 1, 2$.
2: while $\epsilon > \epsilon_{des}$ do
3:     for all levels l do
4:         evaluate samples
5:     end for
6:     update estimate for $V_l, l = \{0, ..L\}$
7:     find optimal $N_l, l = \{0, ..L\}$
8:     if !(weak convergence) then
9:         L← L+1
10:        $N_0 \leftarrow N_L$
11:    end if
12: end while
```

The optimal N_l is defined as

$$N_l = 2\epsilon^{-2}\sqrt{V_l/C_l}\Big(\sum^{L}\sqrt{V_l C_l}\Big) \tag{3}$$

which ensure that the estimated variance is less than $\frac{1}{2}\epsilon^2$. Here $V_l = \mathbb{V}[P_l - P_{l-1}]$ and C_l is the respectively related computational effort or cost. A test for weak convergence ensures that $\big(\mathbb{E}[P - P_L]\big)^2$ is less than $\frac{1}{2}\epsilon^2$ with

$$\mathbb{E}[P - P_L] = \mathbb{E}[P_L - P_{L-1}]/(2^{\alpha} - 1). \tag{4}$$

These, being the sufficient conditions, bound the mean square error to be less than ϵ^2. Proof and its derivation as well as the a more specific description of the driver and level routines can be found in [6].

B. Adapting MCTS: Splitting

Monte Carlo Tree Search (MCTS) is a one of the most successful online approaches of the recent years, that has been applied to both MDPs and POMDPs. In contrast to other sampling methods, its complexity does not grow exponentially with the horizon. The basic algorithm iteratively builds a search tree from the current state in attempt to find the best possible action. It involves running many simulations from the current state while updating an estimate of the state action value function. There are four stages over which will be iterated successively for the allotted timespan.

1) Starting from the root node, recursively apply the tree policy for descending the search tree until a leaf node is reached. SELECT it.
2) At the leaf node, choose an action w.r.t. the tree policy and EXPAND the tree by linking the new child node.
3) SIMULATE from the new node according to a default policy.
4) BACK-UP the simulation outcome through the search path from the new to the root node. Statistics of states in the path have to be updated accordingly.

As visual aid, fig. 1 represents the method graphically.

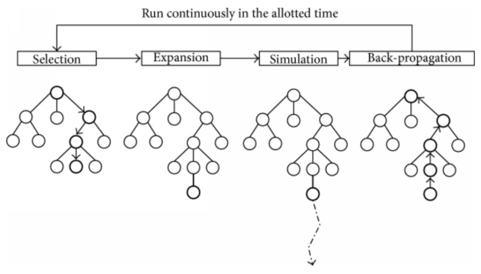

Fig. 1: Monte Carlo Tree Search

The splitting method accelerates the convergence of Monte Carlo simulations by increasing the occurrence of rare events. The method generates more samples of interest by selecting and replicating sample paths. It divides paths into multiple levels based on proximity to the event of interest. If a sample path starting from a level reaches to the next level, the path is copied.Optionally multiple copies could be reproduced. This copy is or these copies are simulated from the current level. If a copy reaches the next level, the replicating procedure is repeated. If not, the simulation of the copy ends and the back-propagation is started. It is a very simple and straightforward approach in which, through repeated replication, generated sample paths are biased toward rare events. Pseudo-code for this method can be found in Alg. 2.

Algorithm 2 Multi Level MCTS

```
$\Gamma, s \leftarrow$ SELECT PATH($T_R$)
$s' \leftarrow$ EXPAND($\Gamma$, $s$)
SIMULATE($s', d$)
**if** PATH $\rightarrow$ NEXT LEVEL **then**
    Replicate PATH
    SIMULATE($s', \Gamma$)
**end if**
BACKUP($T_R, \Gamma, s'$)
```

III. Case Study: Basket Options

"Buy low, sell high" is a famous investing adage about taking advantage of the market's propensity to overshoot on the downside and upside, [8]. Although it sounds very simple, the execution is tricky: It is easy to say whether a certain price is low or high in retrospect, but in the moment, it is monumentally difficult. We will try to tackle this problem using the methods introduced in II-A and II-B

A. Problem Statement

In basket options the value is dependent on a weighted average of J underlying assets,

$$S(t) = \sum_{j=1}^{J} \mu_j S_j(t) \tag{5}$$

each of which satisfies a stochastic differential equation (SDE) of the form

$$dS_j = a(S_j, t)dt + b(S_j, t)W_j(t) \tag{6}$$

driven by Brownian motions $W_j(t)$ with correlation matrix Σ. The task consist in allocating the percentages of an total available asset value in order to maximize the profit. Roughly speaking, impersonating Scrooge McDuck, we aim to find strategic investment actions to take.

As MCTS uses a generative model, we are required to define a stochastic process to model the evolution of each stock. [9], most often assets are modeled through a geometric Brownian motion (GBM)model with a drift $a(X,t) = rX(t)$ and a diffusion $b(X,t) = \sigma x(t)$

$$dX = rXdt + \sigma X dW(t), \qquad 0 < t < T. \tag{7}$$

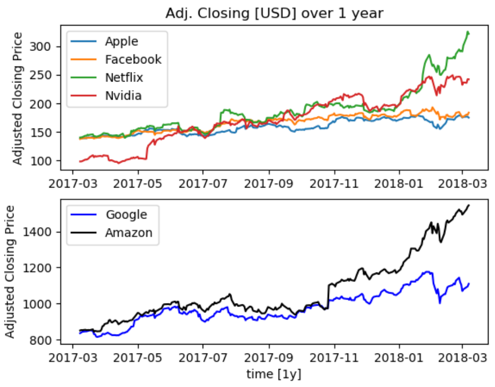

Fig. 2: Financial comparison of assorted stocks
Source: `https://finance.yahoo.com/lookup`

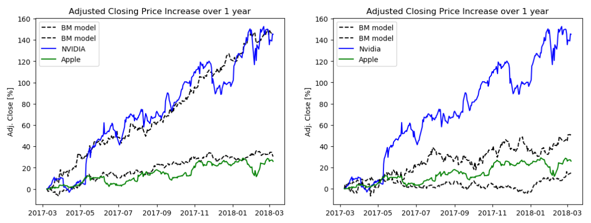

Fig. 3: GBM Simulation

Fig. 3 plots the simulated motion paths resulting from this model. As fig. 3 a seems to be a reasonable approximation, fig. 3 b however is not. Using it would be obsolete.

As GMB does not seem to be an option, the future development will be approximated using a mixture model of data from the past 5 y and the observed changes of the previous days, thus estimating a trend of each stock using the MLMC method outlined in II-A. The simulation results are plotted in fig. 4. Compared to the results in fig.3 the displayed plots look more accurate.

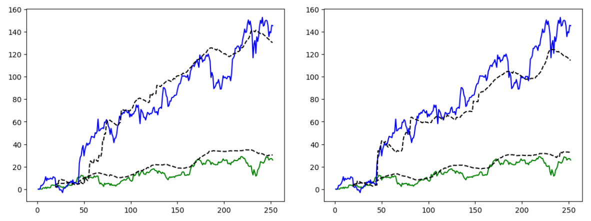

Fig. 4: MLMC and Mixture Model Simulation

Having decided on the "prediction" model, we can will initialize distribution parameters for the all basket stocks options by fitting the fluctuations to Gaussian distributions. The parameters for each assets options are summarized in table I.

	(μ, σ)
Apple	$(0.000763026, 0.0110005)$
Amazon	$(0.00201355, 0.0118056)$
Facebook	$(0.00254726, 0.0109929)$
Google	$(0.00128365, 0.0119121)$
Netflix	$(0.00314721, 0.0235732)$
NVidia	$(0.00674665, 0.0370291)$

TABLE I: Gaussian Distribution Parameters

The implementation of alg. 1 raises two questions: First, how to define the desired accuracy ϵ and, second, how to evaluate the samples based on a given level. Usually the accuracy is either defined w.r.t. to a true model or through repeated convergence test based on the the remaining variance to the computations of the previous level. In this case study, assuming the real model to be the data to compare against would be "cheating" and the computational cost of evaluating all levels is $O(2^l)$ on each level and the increases proportionally to the amount of time steps taken. Thus, instead of defining a desired accuracy ϵ, we base the choice of the l upon the variation in distribution parameters for each stock. This means that if the stock price has been somewhat constant over the past days, it suffices to sample on a shallow level 1 or 2. Though, if the market price has been deviating from our expectations in the recent past, sampling on a level of 6 seems more appropriate.

This same intuitive procedure is used to classify the paths and leaf nodes for splitting.

The sampling proceedure is quite simple: If we choose to make a timestep of Δ days, then we run a simulation Δ simulations of splitting each day into 2^l samples (time increments) after that new data about the true price variation of the stock and our observation can be taken into account to update our belief of the current stock trend.

B. Results

In this section, we will discuss the results for an example portfolio that includes the stocks options of Apple Inc., Google (Alphabet Inc.), Amazon.com Inc., Nvidia, Netflix Inc. and Facebook Inc.. The initial asset value was set to be 1000. After each step of 10 days the agent is able to reallocate the share of its budget as investments in each stock. Thus, the profit made after ten days can be reinvested in the following step. However, the agent is free to choose any repartition of his budget, including not investing at all (if all stock prices were dropping).

Two solvers were used, namely the classical MCTS and double progressive widening (DPW), [10]. All simulations ran with a 100 iterations and a depth of 5. These parameters were iteratively found to produce satisfying results for small portfolios of 3 stock and were then kept constant to facilitate a quantitative comparison of the results.

	GBM/M*+MCTSSolver	GBM/M*+DPW
avg. Profit	1.5% / 21.3%	2% / 25.1%
comp. Time [s]	7.9 / 8.3	6.4 / 6.9
	MLMC+MCTSSolver	MLMC+DPW
avg. Profit	10.45%	12.2%
comp. Time [s]	9.8	10.6

TABLE II: (Apple, Amazon, Facebook)

The results of table II are the average results of 5 runs for each Model-Solver pair. The GBM stands for a geometric Brownian Motion model, as described in III-A and plotted in fig. 3. The M^* denotes the perfect model. Using this model is thus simply cheating. However as we are solving with a sampling based algorithm, and since the action space is quite large, we will not find the exact solution to the problem. The foremost interesting conclusion that can be drawn from the data in table II is that, we were able to significantly increase the profit made over ten times steps, i.e. 100 days. But, the computational cost seems to have increased by 25% for both solvers.

	GBM/M*+MCTSSolver	GBM/M*+DPW
avg. Profit	2.1% / 23.8%	-.5% / 28.1%
comp. Time [s]	81.0 / 83.2	64.1 / 67.8
	MLMC+MCTSSolver	MLMC+DPW
avg. Profit	12.9%	15.2%
comp. Time [s]	95.1	94

TABLE III: (Apple, Amazon, Facebook, Google)

Enlarging the portfolio, thus adding the stock options available to our agent considerable increases the action space from 286 for the initial portfolio to 1001 for the new one. Table III is a summary of the results averaged over 5 runs for each combination. Most apparent is the drastic increase in computation time, which has increased by factor 10, which can be rated to the increased increased action and state spaces. The mean profit for this portfolio seems to be higher than the to the initial one, as the investments can now be partitioned among more stocks and investing in into Alphabet Inc. would have been - in retrospective - a wise decision before January 2018, fig. 2.

The computation time increased again by factor 10, naturally inferring that the computation time seems to be $O(10^n)$, where n is the number of stock in a portfolio. In general the performance of the standard MCTS solver has been inferior to DPW. Also, as expected, the GBM model resulted rather random profits. The fact that the profit has been positive in all runs, is most likely a direct

	GBM/M*+MCTSSolver	GBM/M*+DPW
avg. Profit	1.6% / 25.3%	3.2% / 35%
comp. Time [s]	1090 / 1154	857 / 994
	MLMC+MCTSSolver	MLMC+DPW
avg. Profit	13.45%	17.1%
comp. Time [s]	1298	1302

TABLE IV: (Apple, Amazon, Facebook, Google, Netflix)

consequence from the fact that all stocks displayed an upwards trend until March of this year.

However the MLMC and splitting modifications have proven to yield satisfying results in these simulations.

17.1% profit - such as the MLMC with a DPW solver reached on average for a portfolio of five, table IV - is yielded on average as well when the portfolio comprises Nvidia and Amazon only. These were the stocks with the greatest increase in value in the last year. Note that Nvidia was not included in the previous portfolios, thus there has been a profit potential inaccessible to the agent.

IV. Conclusions

A. Simulation Annotations

MCTS can be applied to MPDs and POMDPs equally well. The approach differs only in that the counts and values are associated with entire histories - composed of actions a and observations o - instead of states. The histories can be organized in trees and during execution the structure expands. Since the state space of the *Basket* example if continuous each state from the simulation will be a "unknown" state. Clustering these states w.r.t. a certain proximity criterion or definition might be a future goal for this project.

A second remark has to be made for completeness. Even though we are dealing with updating our information about every stocks trend after time step Δ. These have not been integrated in the problem definition as "observations". Instead, we have modeled this decision making problem as an hidden model MPD (hmMDP), [11]. A hmMDP describes a POMDP, which has only deterministically changing partially observable state variables. Thus we can model the POMDP as an MDP susceptible to a parameter θ that is updated after each action. For this example θ was a tuple containing distribution characteristics of each stock necessary to defining its trend.

Finally, future work building up on this project should be aiming to speed up the routines. This would enable the usage of this approach with more complex problems.

B. Conclusion

The paper presents two modifications to help improving performance of MCTS for solving MPDs and POMDPs involving a reward model with rare events. Multi-Level Monte Carlo is a recently developed approach, which leverages the idea of control variates for variance reduction and reducing computational cost by performing more low cost in combination with a few high accuracy simulations. Empirical studies, as well as the case study studied in this paper have demonstrated increased performance of the algorithm.

On a larger scale, future work should be directed to magnifying the application use of these techniques. MLMC has been mainly researched in the domain of computational finance, while the splitting method is a convenient modification to the MCTS. Both methods aren't widely spread, though having great potential due to their simplicity of implementation yet yielding significant improvements in accuracy and computational efficiency.

References

[1] S. Thrun, "Monte carlo pomdps," in *Advances in neural information processing systems*, pp. 1064–1070, 2000.

[2] M. Kochenderfer, *Decision Making Under Uncertainty: Theory and Application*. MIT Lincoln Laboratory Series, MIT Press, 2015.

[3] A. B. Owen, *Monte Carlo theory, methods and examples*. 2013.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] Y. Kim, Y. Gur, and M. J. Kochenderfer, "Heuristics for planning with rare catastrophic events," in *Simulation Conference (WSC), 2017 Winter*, pp. 3030–3041, IEEE, 2017.

[6] M. B. Giles, "Multilevel monte carlo methods," *Acta Numerica*, vol. 24, pp. 259–328, 2015.

[7] M. B. Giles, "Multilevel monte carlo method and research." `http://people.maths.ox.ac.uk/~gilesm/`.

[8] Investopedia, "A look at the buy low, sell high strategy." `https://www.investopedia.com/articles/investing`, Jan 12 2018.

[9] M. B. Giles, "Multilevel monte carlo path simulation," *Operations Research*, vol. 56, no. 3, pp. 607–617, 2008.

[10] A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard, "Continuous upper confidence trees," in *International Conference on Learning and Intelligent Optimization*, pp. 433–445, Springer, 2011.

[11] M. Chen, E. Frazzoli, D. Hsu, and W. S. Lee, "Pomdp-lite for robust robot planning under uncertainty," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 5427–5433, IEEE, 2016.

[12] G. Rubino and B. Tuffin, *Rare Event Simulation using Monte Carlo Methods*. Wiley, 2009.

[13] M. wiechowski, H. Park, J. Madziuk, and K.-J. Kim, "Recent advances in general game playing," vol. 2015, p. 986262, 09 2015.

[14] M. B. Giles, "Multilevel monte carlo for basket options," in *Winter Simulation Conference*, pp. 1283–1290, Winter Simulation Conference, 2009.