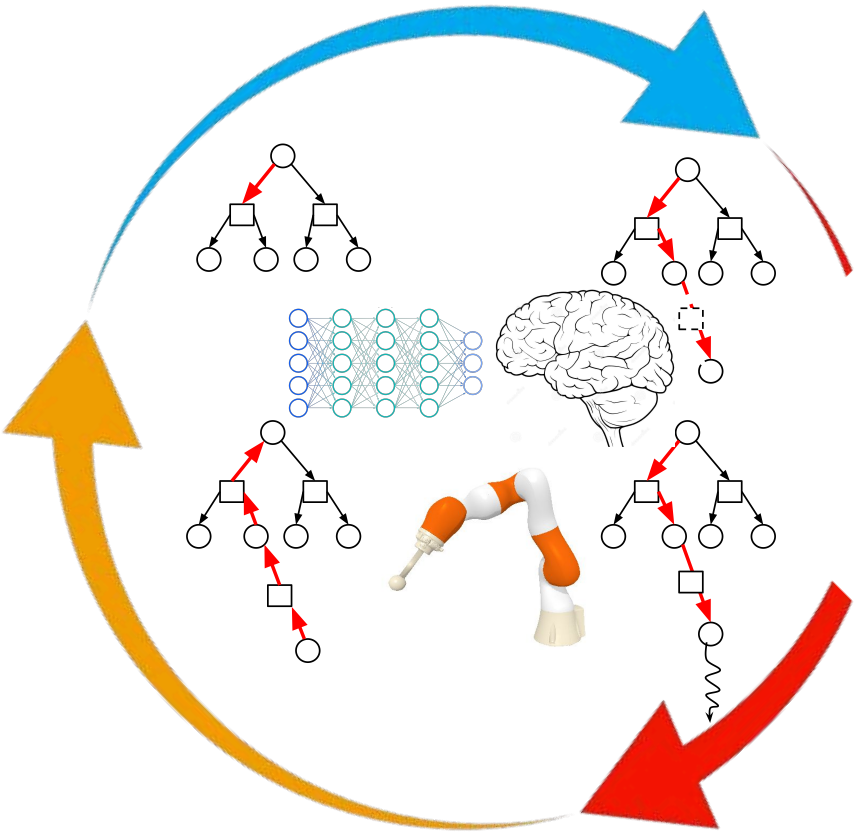
Sample Efficient Monte Carlo Tree Search for Robotics

Zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)
Genehmigte Dissertation von Tuan Dam
Tag der Einreichung: August 11, 2022, Tag der Prüfung: September 22, 2022

1. Gutachten: Prof. Jan Peters, Ph.D
2. Gutachten: Prof. Csaba Szepesvári, Ph.D
3. Gutachten: Prof. Dr. Joni Pajarinen
Darmstadt, Technische Universität Darmstadt,
Jahr der Veröffentlichung der Dissertation auf TUprints: 2023



TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department

Intelligent Autonomous Systems Group

Sample Efficient Monte Carlo Tree Search for Robotics

Submitted doctoral thesis by Tuan Dam

1. Review: Prof. Jan Peters, Ph.D
2. Review: Prof. Csaba Szepesvári, Ph.D
3. Review: Prof. Dr. Joni Pajarinen

Date of submission: August 11, 2022
Date of thesis defense: September 22, 2022

Darmstadt, Technische Universität Darmstadt,
Jahr der Veröffentlichung der Dissertation auf TUprints: 2023

Bitte zitieren Sie dieses Dokument als:
URN: urn:nbn:de:tuda-tuprints-229318
URL: http://tuprints.ulb.tu-darmstadt.de/22931

Dieses Dokument wird bereitgestellt von tuprints,
E-Publishing-Service der TU Darmstadt
http://tuprints.ulb.tu-darmstadt.de
tuprints@ulb.tu-darmstadt.de

For my father

Erklärungen laut Promotionsordnung

§8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, August 11, 2022

T. Dam

Abstract

Artificial intelligent agents that behave like humans have become a defining theme and one of the main goals driving the rapid development of deep learning, particularly reinforcement learning (RL), in recent years. Monte-Carlo Tree Search (MCTS) is a class of methods for solving complex decision-making problems through the synergy of Monte-Carlo planning and Reinforcement Learning (RL). MCTS has yielded impressive results in Go (AlphaGo), Chess(AlphaZero), or video games, and it has been further exploited successfully in motion planning, autonomous car driving, and autonomous robotic assembly tasks. Many of the MCTS successes rely on coupling MCTS with neural networks trained using RL methods such as Deep Q-Learning, to speed up the learning of large-scale problems. Despite achieving state-of-the-art performance, the highly combinatorial nature of the problems commonly addressed by MCTS requires the use of efficient *exploration-exploitation* strategies for navigating the planning tree and quickly convergent value backup methods. Furthermore, large-scale problems such as Go and Chess games require the need for a *sample efficient* method to build an effective planning tree, which is crucial in on-the-fly decision making. These acute problems are particularly evident, especially in recent advances that combine MCTS with deep neural networks for function approximation. In addition, despite the recent success of applying MCTS to solve various autonomous robotics tasks, most of the scenarios, however, are *partially observable* and require an advanced planning method in complex, unstructured environments.

This thesis aims to tackle the following question: *How can robots plan efficiency under highly stochastic dynamic and partial observability?* The following paragraphs will try to answer the question:

First, we propose a novel backup strategy that uses the power mean operator, which computes a value between the average and maximum value. We call our new approach Power Mean Upper Confidence bound Tree (Power-UCT). We theoretically analyze our method providing guarantees of convergence to the optimum. Finally, we empirically demonstrate the effectiveness of our method in well-known Markov decision process (MDP) and partially observable Markov decision process (POMDP) benchmarks, showing

significant improvement in terms of *sample efficiency* and convergence speed w.r.t. state-of-the-art algorithms.

Second, we investigate an efficient exploration-exploitation planning strategy by providing a comprehensive theoretical convex regularization framework in MCTS. We derive the first regret analysis of regularized MCTS, showing that it guarantees an exponential convergence rate. Subsequently, we exploit our theoretical framework to introduce novel regularized backup operators for MCTS based on the relative entropy of the policy update and, more importantly, on the Tsallis entropy of the policy, for which we prove superior theoretical guarantees. Afterward, we empirically verify the consequence of our theoretical results on a toy problem. Eventually, we show how our framework can easily be incorporated in AlphaGo, and we empirically show the superiority of convex regularization, w.r.t. representative baselines, on well-known RL problems across several Atari games.

Next, we take a further step to draw the connection between the two methods, Power-UCT and the convex regularization in MCTS, providing a rigorous theoretical study on the effectiveness of α-divergence in online Monte-Carlo planning. We show how the two methods can be related by using α-divergence. We additionally provide an in-depth study on the range of α parameter that helps to trade-off between *exploration-exploitation* in MCTS, hence showing how α-divergence can achieve state-of-the-art results in complex tasks.

Finally, we investigate a novel algorithmic formulation of the popular MCTS algorithm for robot path planning. Notably, we study Monte-Carlo Path Planning (MCPP) by analyzing and proving, on the one part, its exponential convergence rate to the optimal path in fully observable MDPs, and on the other part, its probabilistic completeness for finding feasible paths in POMDPs (proof sketch) assuming limited distance observability. Our algorithmic contribution allows us to employ recently proposed variants of MCTS with different exploration strategies for robot path planning. Our experimental evaluations in simulated 2D and 3D environments with a 7 degrees of freedom (DOF) manipulator and in a real-world robot path planning task demonstrate the superiority of MCPP in POMDP tasks.

In summary, this thesis proposes and analyses novel value backup operators and policy selection strategies both in terms of theoretical and experimental perspectives to help cope with *sample efficiency* and *exploration-exploitation* trade-off problems in MCTS and bring these advanced methods to robot path planning, showing the superiority in POMDPs w.r.t the state-of-the-art methods.

Zusammenfassung

Künstliche intelligente Agenten, die sich wie Menschen verhalten, sind in den letzten Jahren zu einem bestimmenden Thema und einem der Hauptziele geworden, die die rasante Entwicklung des Deep Learning, insbesondere des Reinforcement Learning (RL), vorantreiben. Monte-Carlo Tree Search (MCTS) ist eine Klasse von Methoden zur Lösung komplexer Entscheidungsprobleme durch die Synergie von Monte-Carlo-Planung und Reinforcement Learning (RL). MCTS hat beeindruckende Ergebnisse bei Go (AlphaGo), Schach (AlphaZero) oder Videospielen erzielt und wurde auch bei der Bewegungsplanung, dem autonomen Fahren von Autos und der autonomen Montage von Robotern erfolgreich eingesetzt. Viele der MCTS-Erfolge beruhen auf der Kopplung von MCTS mit neuronalen Netzen, die mit RL-Methoden wie Deep Q-Learning trainiert werden, um das Lernen von großen Problemen zu beschleunigen. Obwohl MCTS den neuesten Stand der Technik erreicht, erfordert die hochgradig kombinatorische Natur der Probleme, die üblicherweise mit MCTS behandelt werden, den Einsatz effizienter Strategien für die Navigation im Planungsbaum und schnell konvergierende Methoden für die Wertsicherung. Darüber hinaus erfordern große Probleme wie Go- und Schachspiele eine *Sampling-Effizienz*-Methode, um einen effektiven Planungsbaum zu erstellen, der für die fliegende Entscheidungsfindung entscheidend ist. Diese akuten Probleme sind besonders offensichtlich, vor allem bei den jüngsten Fortschritten, die MCTS mit tiefen neuronalen Netzen zur Funktionsannäherung kombinieren. Trotz der jüngsten Erfolge bei der Anwendung von MCTS zur Lösung verschiedener Aufgaben der autonomen Robotik sind die meisten Szenarien jedoch *teilweise beobachtbar* und erfordern eine fortschrittliche Planungsmethode in komplexen, unstrukturierten Umgebungen.

Diese Arbeit zielt darauf ab, die folgenden Fragen zu beantworten: Wie können Roboter unter hoch stochastischer Dynamik und partieller Beobachtbarkeit effizient planen? In den folgenden Abschnitten wird versucht, diese Frage zu beantworten: Zunächst schlagen wir eine neuartige Backup-Strategie vor, die den Power-Mittelwert-Operator verwendet, der einen Wert zwischen dem Durchschnitts- und dem Maximalwert berechnet. Wir nennen unseren neuen Ansatz Power-UCT. Wir analysieren unsere Methode theoretisch und geben

Garantien für die Konvergenz zum Optimum. Schließlich demonstrieren wir empirisch die Effektivität unserer Methode in bekannten MDP- und POMDP-Benchmarks, die eine signifikante Verbesserung in Bezug auf die Stichprobeneffizienz und die Konvergenzgeschwindigkeit im Vergleich zu modernen Algorithmen zeigen.

Zweitens untersuchen wir eine effiziente Explorations-Ausnutzungs-Planungsstrategie, indem wir einen umfassenden theoretischen konvexen Regularisierungsrahmen in MCTS bereitstellen. Wir leiten die erste Regret-Analyse von regularisierten MCTS ab und zeigen, dass sie eine exponentielle Konvergenzrate garantiert. Anschließend nutzen wir unseren theoretischen Rahmen, um neuartige regularisierte Backup-Operatoren für MCTS einzuführen, die auf der relativen Entropie der Politikaktualisierung und, was noch wichtiger ist, auf der Tsallis-Entropie der Politik basieren, für die wir überlegene theoretische Garantien beweisen. Anschließend verifizieren wir die Konsequenz unserer theoretischen Ergebnisse empirisch an einem Spielzeugproblem. Schließlich zeigen wir, wie unser Rahmenwerk leicht in AlphaGo integriert werden kann, und wir zeigen empirisch die Überlegenheit der konvexen Regularisierung im Vergleich zu repräsentativen Baselines bei bekannten RL-Problemen in mehreren Atari-Spielen.

In einem weiteren Schritt stellen wir die Verbindung zwischen den beiden Methoden, Power-UCT und der konvexen Regularisierung in MCTS, her und liefern eine rigorose theoretische Studie über die Effektivität der α-Divergenz in der Online Monte-Carlo Planung. Wir zeigen, wie die beiden Methoden durch die Verwendung der α-Divergenz miteinander verbunden werden können. Darüber hinaus bieten wir eine detaillierte Studie über den Bereich des α-Parameters, der dabei hilft, einen Kompromiss zwischen *exploration-exploitation* in MCTS zu finden, und zeigen somit, wie α-Divergenz in AlphaGo und AlphaZero integriert werden kann, um bei komplexen Aufgaben die besten Ergebnisse zu erzielen.

Schließlich untersuchen wir eine neuartige algorithmische Formulierung des beliebten MCTS-Algorithmus für die Roboterbahnplanung. Insbesondere untersuchen wir die Monte-Carlo-Pfadplanung (MCPP), indem wir zum einen ihre exponentielle Konvergenzrate zum optimalen Pfad in vollständig beobachtbaren MDPs analysieren und beweisen und zum anderen ihre probabilistische Vollständigkeit für das Finden machbarer Pfade in POMDPs (Beweisskizze) unter der Annahme begrenzter Beobachtbarkeit der Entfernung. Unser algorithmischer Beitrag ermöglicht es uns, kürzlich vorgeschlagene Varianten von MCTS mit verschiedenen Explorationsstrategien für die Roboterbahnplanung einzusetzen. Unsere experimentellen Auswertungen in simulierten 2D- und 3D-Umgebungen mit einem Manipulator mit 7 Freiheitsgraden (DOF) und in einer realen Roboter-Bahnplanungsaufgabe zeigen die Überlegenheit von MCPP in POMDP-Aufgaben.

Zusammenfassend lässt sich sagen, dass in dieser Arbeit neuartige Value-Backup-Operatoren und Policy-Selection-Strategien sowohl aus theoretischer als auch aus experimenteller Sicht vorgeschlagen und analysiert werden, um mit den Trade-Off-Problemen *Stichproben-Effizienz* und *Exploration-Exploitation* in MCTS zurechtzukommen und diese fortschrittlichen Methoden in die Roboterbahnplanung einzubringen.

Acknowledgements

Today I sit here writing these acknowledgements to express my deepest gratitude to all who have accompanied, helped, and supported me along the way to completing this dissertation. This study would not have been possible without the support of all of you.

First and foremost, I would like to thank Professor Jan Peters. Thank you for creating a great environment and welcoming me into the IAS family. This is a great honor for me. Thank you for your understanding, your advice, your encouragement, your help, and for providing me with an excellent environment to complete this work. Without it, there would be no success like today.

Next, words cannot express all my deep gratitude to Professor Joni Pajarinen. I want to thank you so much for guiding me throughout this journey. Thank you for your dedicated guidance, understanding, and sympathy that cheered me up throughout the dissertation. Without your help, I probably would not be where I am today. Your great input, vision, and encouragement are difficult to express here. My sincere thanks and deepest gratitude to you.

The next person I would like to acknowledge is Dr. Carlo D'Eramo, who has closely followed, played, studied and worked with me since the first days I came here. Thank you very much for your help. I hope for more opportunities to work together in the future.

The next one is definitely Dr. Georgia Chalvatzaki. I cannot express my gratitude to you for your direct or indirect help to me. Thank you for your kindness and warm heart. You will be much more successful in the future.

Pascal Klink, you are one of my greatest colleagues. You are always great and professional in all the things you do. Having the chance to work with you is my great luck. Thank you for helping me along the way. I hope there will be many, many more good collaborations in the future.

My big thanks and sincere gratitude to all members of IAS family: Boris, Hany, Niklas, Oleg, Puze, Junning, Michael, Joe, Samuele, Simone, Riad, Firas, Davide, Ali, Snehal, An, Tim, Daniel, João, Kay, Kai, Bang, Tianyu, Julen, Marco, Svenja, Doro.... Thank you so much for your help. We have wonderful memories with each other.

I also thank my committee members Csaba Szepesvári and Oskar von Stryk for evaluating my dissertation and participating in the defense.

Thanks to my dear family. My sister, thank you for giving me motivation and encouraging words and for taking care of our mother so that I could continue my journey and finish this thesis. Next, I want to give you my great love, mom. Thank you for your boundless love, advice and encouragement and for giving me a positive outlook on the future.

I love you, mama.

Thank you for everything!

Contents

Abstract **vi**

Zusammenfassung **viii**

Acknowledgements **xi**

1. Introduction **1**
1.1. Motivation . . . 3
1.2. Major Contributions . . . 5
1.2.1. Power Mean Estimation in Monte-Carlo Planning . . . 5
1.2.2. Convex Regularization in Monte-Carlo Tree Search . . . 5
1.2.3. α-divergence in Monte-Carlo Tree Search . . . 6
1.2.4. Monte-Carlo Robot Path Planning . . . 6
1.3. Thesis Outline . . . 7

2. Generalized Mean Estimation in Monte-Carlo Tree Search **9**
2.1. Introduction . . . 9
2.2. Related Work . . . 11
2.3. Background . . . 11
2.3.1. Monte-Carlo Tree Search . . . 12
2.3.2. Upper Confidence bound for Trees . . . 12
2.3.3. Power Mean . . . 13
2.4. Power Mean Backup . . . 14
2.4.1. Power-UCT . . . 14
2.5. Theoretical analysis . . . 16
2.6. Experiments . . . 19
2.6.1. *FrozenLake* . . . 19
2.6.2. Copy Environment . . . 20
2.6.3. Rocksample and PocMan . . . 21

2.7. Conclusion . . . 23

3. Convex Regularization in Monte-Carlo Tree Search 25
3.1. Introduction . . . 25
3.2. Preliminaries . . . 27
3.2.1. Markov Decision Processes . . . 27
3.2.2. Monte-Carlo Tree Search and Upper Confidence bounds for Trees . 27
3.3. Regularized Monte-Carlo Tree Search . . . 28
3.3.1. Legendre-Fenchel transform . . . 29
3.3.2. Regularized backup and tree policy . . . 30
3.3.3. Convergence rate to regularized objective . . . 31
3.4. Entropy-regularization backup operators . . . 31
3.4.1. Regret analysis . . . 32
3.4.2. Error analysis . . . 34
3.5. Empirical evaluation . . . 36
3.5.1. Synthetic tree . . . 37
3.5.2. Entropy-regularized AlphaGo . . . 39
3.6. Related Work . . . 39
3.7. Conclusion . . . 40

4. A Unified Perspective on Value Backup and Exploration in Monte-Carlo Tree Search 43
4.1. Introduction . . . 43
4.2. Related Work . . . 45
4.3. Preliminaries . . . 46
4.3.1. Markov Decision Processes . . . 46
4.3.2. Monte-Carlo Tree Search . . . 46
4.3.3. α-divergence . . . 46
4.4. α-divergence in MCTS . . . 47
4.4.1. α-divergence Regularization in MCTS . . . 47
4.4.2. Connecting Power Mean with α-divergence . . . 48
4.4.3. Regret and Error Analysis of α-divergence in Monte-Carlo Tree Search 49
4.5. Empirical Evaluation . . . 50
4.5.1. Synthetic Tree . . . 50
4.6. Conclusion . . . 52

5. Monte-Carlo Robot Path Planning 53
5.1. Introduction . . . 53

5.2. Related Work 56
5.3. Background 57
5.3.1. RRT* 58
5.4. Problem formulation 59
5.5. Monte-Carlo path planning 60
5.5.1. Fully observable environment 60
5.5.2. Partially observable environment 61
5.5.3. Theoretical analysis 62
MDP 63
POMDP 65
5.6. Experiments 66
5.6.1. Experimental evaluation in simulation 67
5.6.2. Real robot object disentangling task 69
5.7. Conclusions 71

6. Conclusion **72**
6.1. Summary of Contributions 72
6.2. Open Problems 74
6.2.1. Maximum Expected Value Estimation of Power Mean 74
6.2.2. Uncertainty Value Estimation 74
6.2.3. Efficient Monte-Carlo Planning for Autonomous Car Driving 75
6.3. Future Work 75
6.3.1. p-Adaptation 76
6.3.2. Wasserstein Monte-Carlo Tree Search 76
6.3.3. AlphaGo for Autonomous Car Driving 77

A. Appendix **78**
A.1. Generalized Mean Estimation in Monte-Carlo Tree Search 78
A.2. Convex Regularization in Monte-Carlo Tree Search 93
A.3. A Unified Perspective on Value Backup and Exploration in Monte-Carlo Tree Search 104

Curriculum Vitae **107**

List of Figures **112**

List of Tables **115**

List of Algorithms **116**

List of Acronyms 117

List of Symbols 119

Bibliography 122

1. Introduction

> "Driving in Monte Carlo is like riding a bike in your house."
>
> — *Nelson Piquet*

Autonomous agents performing online planning tasks need optimal decisions with an efficient sampling strategy. These are even more important in robotic applications, such as autonomous car driving [1]–[3], robot path planning [4]–[7], and human-robot collaboration [8], [9], where the need for optimal online decisions is crucial. In addition, sampling in real robotic scenarios has been expensive due to the slow movements of robots and expensive queries of physical robot systems [10], preventing the applicability of current methods. Monte-Carlo Tree Search [11] is a powerful tool for online robot planning that, besides having many advantages that can ensure choosing an optimal decision [12] and help with sample efficiency because of the use of a simulation for planning, still poses many open problems.

What is Monte-Carlo Tree Search?

Monte-Carlo Tree Search (MCTS) [11] is an effective online planning strategy that combines Monte-Carlo sampling with forward tree search to find optimal decisions on-the-fly. MCTS uses a black-box model of environments in simulation to build a planning tree. As shown in Fig. 1.1, MCTS consists of four basic steps: Selecting the nodes based on the current statistical information to traverse in the tree, expanding the tree, roll-out in the environments and backup the collected rewards from the environment along the tree. The core nature that determines the success of MCTS planning is an effective value backup operator and an efficient policy search in the tree.

Why do we need Monte-Carlo Tree Search?

By coupling MCTS planning with pre-trained deep neural networks, MCTS has shown great success in various tasks ranging from board games to video games. Notably, with impressive results in the game of Go (AlphaGo) [13] and the game of Chess (AlphaZero) [14] and

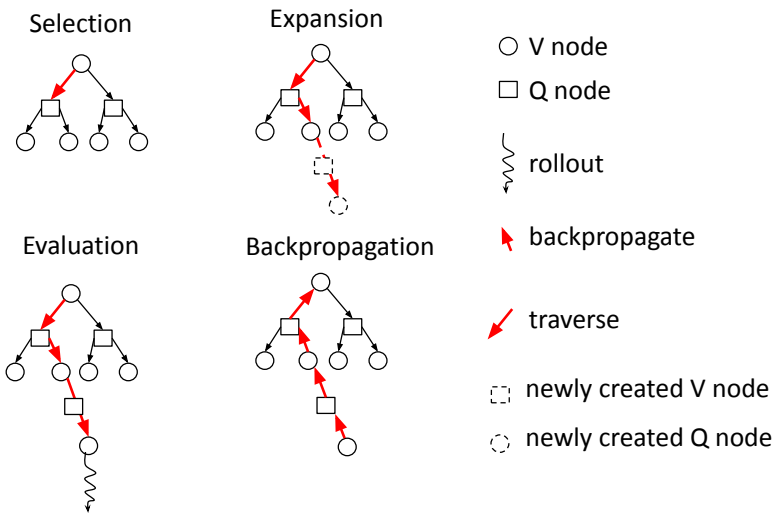


Figure 1.1.: This figure illustrates the four basic steps of Monte-Carlo Tree Search.

recent successful applications of MCTS as an effective planning strategy in autonomous car driving [15], robot assembly [16], and robot motion planning tasks [17], MCTS has been one of the topics that play an increasingly important role in driving the development of autonomous robotics applications.

What are the current Monte-Carlo Tree Search problems?

The use of simulations for planning in MCTS helps to shorten the limitations of sampling in real robots, such as slow movements and expensive queries of physical systems. However, this poses an open research question that how can we efficiently sample in simulations that still ensure choosing the optimum? This problem is called *sample efficiency*. *sample efficiency* refers to the number of times an agent interacts with simulated environments (the number of actions and the corresponding observed states/rewards performed in simulations) to get to the optimum. An algorithmic MCTS strategy is called *sample efficiency* if it makes good use of samples in simulations for the fast convergence of value functions and ensures choosing optimal decisions.

Additionally, the high stochasticity of real robotic tasks requires an effective method to either exploit the current good action or explore other actions hoping to reach higher performance levels. This is called *exploration-exploitation* dilemma. *exploration-exploitation* trade-off in MCTS refers to an effective value function estimation and the corresponding

policy selection in the tree that balances between the exploration of good branches to get more rewards or the exploitation of other branches for faster value function convergence.

In addition, robots acting under uncertainty face many difficulties. Full states of environments are only partially observed, making it hard for decision making. An optimal policy can be obtained based on the belief distributions over the states. However, the belief calculation is computationally expensive, especially in high-dimensional state space. This disclosed problem has been severely implied in robot path planning, in which the current state-of-the-art methods such as Rapidly exploring random trees (RRT)s/RRTs* are only designed for fully observable tasks.

The objective study of this thesis is to investigate Monte-Carlo Planning methods both in terms of theory and experiments to solve these *sample efficiency* and *exploration-exploitation* dilemma problems and transform these advanced methodologies to robot path planning tasks, showing the advantages in *partial observable* settings both in simulations and real robotic scenarios.

1.1. Motivation

With the progressive growth of science and technology, notably artificial intelligence (AI), robots nowadays play an increasingly important role in human life. Humans have developed intelligent robots capable of performing challenging tasks for industrial production and our own daily lives. Fundamental building blocks contributing to this widespread success are the increasing popularity of machine learning (ML), particularly reinforcement learning (RL) and deep learning (DL), which have driven the development of robotics.

DL has recently gained wide adoption in both academia and industry and has been successfully applied in a wide range of applications from computer vision [18]–[21] and natural language processing [22]–[24] to robotics [25]–[27]. Contributing to these successes are the outcomes of AlphaGo [28] and AlphaZero [14] that can defeat humans in the Go and Chess games. The profound success of AlphaGo, AlphaZero lies in the synergistic combination of Monte-Carlo Tree Search (MCTS) [11], an effective method that combines a random sampling strategy with an online tree search to determine the optimal decision, with pre-trained neural networks using Reinforcement Learning (RL) [29] methods such as Deep Q-Learning [30]. These achievements become even more remarkable due to the high branching factor and the large-scale properties of Go and

Chess games. However, despite these successes, both AlphaGo and AlphaZero suffer from poor sample efficiency mostly because of the polynomial convergence rate of the employed state-of-the-art-method Upper Confidence bound Tree (UCT) [12] or Polynomial Upper Confidence bound Tree (PUCT) [31], which uses the average reward backup operator. The inefficiency of the average backup operator is well-known for the issue of underestimating the optimum [32]. One can think of using the maximum backup operator [32], but it overestimates the optimal branch.

MCTS has further shown surprising results in videos game [33]. Notable examples are the outcome achievements of Atari games [34], which unify a forward tree search with a prior deep Q-learning network. However, the high stochasticity of Atari games [34] requires an efficient exploration strategy for sufficient accumulation of rewards from environments to speed up the convergence of value estimation in the tree.

These issues pose an open research question to find an effective backup operator for sample efficiency in combination with an efficient exploration-exploitation strategy.

One of a concerned topic in robotic community is robot path planning. Robot path planning is typically addressed by sampling-based methods [35], [36]. Rapidly exploring random trees (RRTs*) [37] is the chosen state-of-the-art method that ensures the probabilistic completeness to find a feasible path in fully observable environments [38]. However, most of the real robotic tasks are in partially observable environments because robots typically observe information about environments from laser sensors [39], camera images [40], and sensory feedback [41], which typically contain noise. Therefore, robots normally operate in environments under uncertainty. The need for a strategic online planning method in unknown, unstructured environments becomes critical.

This thesis tackles following open MCTS problems: *sample efficiency, exploration-exploitation*. We take a further step to show how can MCTS help with *partial observability*. in robotic tasks. First, backup operator and planning policy in the MCTS tree in both MDP and POMDP environments are crucial to break the issue of sample efficiency. Second, entropy regularization in online Monte-Carlo planning helps to balance exploration and exploitation. Finally, a novel sampling-based robot path planning method based on MCTS has been proposed to show the advances over the state-of-the-art RRT*, notably in partial observability environments.

1.2. Major Contributions

This dissertation proposes advanced methods for one of a powerful tools for online planning in robotics, MCTS, and brings these advanced methods as new sampling-based techniques to solve robot path planning tasks, showing the superiority of the methods in POMDPs both in simulations and real robots. The contributions of this dissertation are fourfold: (i) Demonstrate that the use of power mean (Power-UCT) as a new backup operator can help with the sample efficiency problem in online Monte Carlo planning. Provide an asymptotic convergence guarantee for the optimum. (ii) Introduce the use of convex regularization framework in MCTS with an exponential convergence guarantee. Investigate different types of entropy-based regularization and show the advantages for solving the exploration-exploitation problem in Monte Carlo planning. (iii) Provide a unified view of α-divergence in MCTS. Combining the two methods (Power-UCT and the convex regularization framework) and giving a rigorous theoretical analysis. (iv) Propose to use MCTS for robot path planning. Provide an exponential convergence guarantee to find the optimal path in MDPs, and probabilistic completeness to find a feasible path in POMDPs, showing the advantage over RRT* in POMDPs. Each of these contributions will be presented separately in four main chapters. Here we list the main contributions.

1.2.1. Power Mean Estimation in Monte-Carlo Planning

We show how the power mean backup operator (Power-UCT) can help overcome the limitations of regular methods like UCT [12], which is well-known for the problem of underestimating the tree's optimal branch in terms of a novel backup operator and sampling strategies [32]. Mainly, Power-UCT uses power mean as a new backup operator and Upper Confidence Bound (UCB) [42] as a policy selection in MCTS, generalizing the well-known state-of-the-art method UCT, which solves the problem of the underestimate of the average backup operator in UCT. Furthermore, we prove that Power-UCT ensures a theoretically polynomial convergence guarantee to the optimum. We additionally analyze the effectiveness of p-constant value that balances exploration and exploitation in some MDP and POMDP environments, showing the benefits of Power-UCT over baseline methods. This approach will be presented in chapter 2.

1.2.2. Convex Regularization in Monte-Carlo Tree Search

We additionally provide a comprehensive study of the use of convex regularization framework in MCTS (Extended Empirical Exponential Weight (E3W)). Subsequently, we prove

that E3W ensures an exponential convergence rate and derive a first regret analysis of regularized MCTS methods. Finally, we narrow down the scope of the study to entropy-based regularizers and show performance benefits of the Tsallis entropy-based regularizer (TENTS) over other methods in some Atari games. This approach will be presented in chapter 3.

1.2.3. α-divergence in Monte-Carlo Tree Search

We provide a rigorous theoretical study of α-divergence in MCTS. In detail, we exploit that entropy regularization in MCTS can be achieved by employing the α-divergence function as the regularizer, therefore to relatively derive the maximum entropy, the relative entropy of the policy update, and, more importantly, derive the Tsallis entropy of the policy those has been proposed in E3W. Furthermore, we generalize the definition of average mean by considering the α-divergence function as the probability distance and deriving power mean, which has been used as a novel backup operator in Power-UCT. Finally, we show how α-divergence can be integrated into MCTS and the effectiveness of α parameters in the Synthetic Tree problem. This approach will be presented in chapter 4.

1.2.4. Monte-Carlo Robot Path Planning

Next, sampling-based approaches such as RRTs* [37] or probabilistic roadmap planning are commonly used in robot path planning. However, these approaches are limited to problems where the robots observe the environment fully. This assumption is not satisfied in practice since many robotics systems operate under partial observability. We propose a forward search-based path planning approach using MCTS with continuous actions and observations with an exponential convergence rate guarantee in fully observable MDPs and probabilistic completeness guarantee in partially observable POMDPs. Existing MCTS methods for continuous spaces do not provide convergence rate proofs and have not been demonstrated in POMDP tasks. Moreover, the MCTS planner takes advantage of recently proposed Power-UCT [43] and Tsallis Entropy Monte-Carlo Planning (TENTS) [44] MCTS approaches, which have not previously been used in robotics. Experiments show that our approach outperforms RRTs in challenging MDP tasks and demonstrates the approach in a real-world robotic POMDP physical object disentangling task where existing path planning methods do not converge to an optimal solution. This approach will be presented in chapter 5.

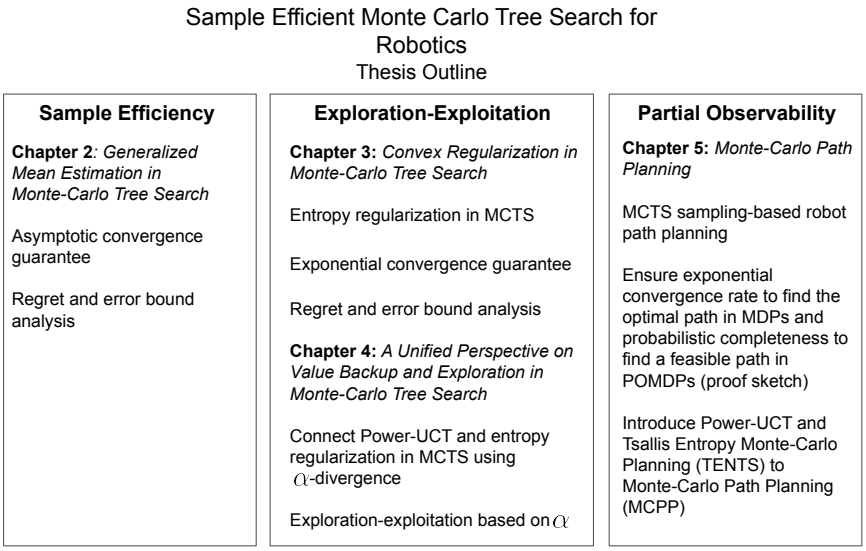


Figure 1.2.: This figure illustrates the outline of the thesis.

1.3. Thesis Outline

In this section, we attempt to give a brief overview of how the rest of the thesis is structured. For the big picture, please refer to the Fig. 1.2.

In Chapter 1: ”Introduction”, we provide motivation for our work and briefly outline contents of the thesis.

In Chapter 2: ”Generalized Mean Estimation in Monte-Carlo Tree Search”, we demonstrate the benefits of using power mean as the backup operator to replace the average backup operator of UCT. We call our method Power-UCT and indicate that Power-UCT ensures the same polynomial convergence rate as UCT and demonstrate the advantages over baselines in both MDP and POMDP tasks.

In Chapter 3: ”Convex Regularization in Monte-Carlo Tree Search”, we introduce the use of convex regularization in MCTS and study the properties of various entropy-based regularizers both in terms of theory and experiment in MCTS.

In Chapter 4: ”A Unified Perspective on Value Backup and Exploration in Monte-Carlo Tree Search”, we introduce α-divergence and show the connection of how to use α-divergence

to derive Power-UCT and E3W, providing the regret bound of Power-UCT and entropy regularisation methods with respect to α parameters.

In Chapter 5: ”Monte-Carlo Path Planning”, we propose to use MCPP as a sampling-based robot path planning method in high dimensional continuous state and action space. Shows the convergence rate in MDPs, probabilistic completeness in POMDPs (proof sketch), and furthermore shows the benefit of the method in POMDPs compared to the baseline RRT*.

In Chapter 6: ”Conclusion”, we review the contributions, open problems and future work of the Ph.D. thesis.

2. Generalized Mean Estimation in Monte-Carlo Tree Search

"Expectation is the mother of all frustration."

— *Antonio Banderas*

We consider Monte-Carlo Tree Search (MCTS) applied to Markov Decision Processes (MDPs) and Partially Observable MDPs (POMDPs), and the well-known Upper Confidence bound for Trees (UCT) algorithm. In UCT, a tree with nodes (states) and edges (actions) is incrementally built by the expansion of nodes, and the values of nodes are updated through a backup strategy based on the average value of child nodes. However, it has been shown that with enough samples the maximum operator yields more accurate node value estimates than averaging. Instead of settling for one of these value estimates, we go a step further proposing a novel backup strategy which uses the power mean operator, which computes a value between the average and maximum value. We call our new approach Power-UCT, and argue how the use of the power mean operator helps to speed up the learning in MCTS. We theoretically analyze our method providing guarantees of convergence to the optimum. Finally, we empirically demonstrate the effectiveness of our method in well-known MDP and POMDP benchmarks, showing significant improvement in performance and convergence speed w.r.t. state of the art algorithms.

2.1. Introduction

Monte-Carlo Tree Search (MCTS) [32] is an effective strategy for combining Monte-Carlo search with an incremental tree structure. MCTS is becoming increasingly popular in the community, especially after the outstanding results recently achieved in the game of Go [28]. In the last years, the MCTS research has mainly focused on effective ways of expanding the tree, performing rollouts, and backing up the average reward computed from rollouts to the parent nodes. We consider the Upper Confidence bound applied

to Trees (UCT) algorithm [12], which combines tree search with the well-known UCB1 sampling policy [45], as an effective way of dealing with the action selection to expand the tree. In UCT, the estimate of the value of each node is computed by performing multiple rollouts starting from the node, and updating the node's value as the average of the collected rewards; then, the node's value is backed up to the parent nodes that are updated with the average of the value of the children nodes. Since the action selection policy tends to favor the best actions in the long run, UCT has theoretical convergence guarantees to the optimal value. However, it has been shown that using the average reward for backup leads to an underestimation of the optimal value, slowing down the learning; on the other hand, using the maximum reward leads to an overestimation causing the same learning problems, especially in stochastic settings [32]. This problem is also evinced in the well-known Q-Learning algorithm [46], where the maximum operator leads to overestimation of the optimal value [47]. Some variants of Q-Learning based on (weighted) mean operators have been successfully proposed to address this issue [48], [49].

In this chapter, we introduce a novel backup operator based on a power mean [50] that, through the tuning of a single coefficient, computes a value between the average reward and the maximum one. This allows to balance between the negatively biased estimate of the average reward, and the positively biased estimate of the maximum reward; in practice, this translates in balancing between a safe but slow update, and a greedy but misleading one. In the following, we propose a variant of UCT based on the power mean operator, which we call Power-UCT. We theoretically prove the convergence of Power-UCT, based on the consideration that the algorithm converges for all values between the range computed by the power mean. We empirically evaluate Power-UCT w.r.t. UCT and the recent MENTS algorithm [31] in classic MDP and POMDP benchmarks. Remarkably, we show how Power-UCT outperforms the baselines both in terms of quality and speed of learning. Thus, our *contribution* is twofold:

1. We propose a new backup operator for UCT based on a power mean, and prove the convergence to the optimal values;
2. We empirically evaluate the effectiveness of our approach comparing it with UCT in well-known MDPs and POMDPs, showing significantly better performance.

The rest of this chapter is organized as follows. First we describe related work. Next, we discuss background knowledge of MCTS, UCB and UCT. Then, we describe the power mean operator and introduce our Power-UCT algorithm. We derive theoretical results and prove convergence to the optimum for Power-UCT. Finally, we present empirical results in both MDP and POMDP problems, showing that Power-UCT outperforms baselines in MCTS.

2.2. Related Work

Several works focus on adapting how UCB1 [45] is applied to MCTS. For this purpose UCB1-tuned [45] modifies the upper confidence bound of UCB1 to account for variance in order to improve exploration. [51] propose a Bayesian version of UCT, which obtains better estimates of node values and uncertainties given limited experience. However, the Bayesian version of UCT is more computation-intensive. While most work on bandits in MCTS focuses on discrete actions, work on continuous action MCTS also exists [52]. Since our MCTS algorithm is based on the UCT algorithm, which is an extension of UCB1, our method could be applied to all of these MCTS algorithms.

Many heuristic approaches based on specific domain knowledge have been proposed, such as adding a bonus term to value estimates based on domain knowledge [53]–[57] or prior knowledge collected during policy search [58]–[62]. We point out that we provide a novel node value backup approach that could be applied in combination with all of these methods.

To improve upon UCT algorithm in MCTS, [63] formalizes and analyzes different on-policy and off-policy complex backup approaches for MCTS planning based on techniques in the Reinforcement Learning literature. [63] propose four complex backup strategies: MCTS(λ), MaxMCTS(λ), MCTS$_\gamma$, MaxMCTS$_\gamma$. [63] report that MaxMCTS(λ) and MaxMCTS$_\gamma$ perform better than UCT for certain setup of parameter. [64] proposed an approach called SARSA-UCT, which performs the dynamic programming backups using SARSA [65]. Both [63] and [64] directly borrow value backup ideas from Reinforcement Learning in order to estimate the value at each tree node. However, they do not provide any proof of convergence.

Instead, our method provides a completely novel way of backing up values in each MCTS node using a power mean operator, for which we prove the convergence to the optimal policy in the limit. The recently introduced MENTS algorithm [31], uses softmax backup operator at each node in combination with EXT3 policy, and shows better convergence rate w.r.t. UCT. Given its similarity to our approach, we empirically compare to it in the experimental section.

2.3. Background

In this section, we first discuss an overview of Monte Carlo Tree Search method. Next, we discuss UCB algorithm and subsequently an extension of UCB to UCT algorithm. Finally,

we discuss the definition of Power Mean operator and its properties.

2.3.1. Monte-Carlo Tree Search

MCTS combines tree search with Monte-Carlo sampling in order to build a tree, where states and actions are respectively modeled as nodes and edges, to compute optimal decisions. MCTS requires a generative black box simulator for generating a new state based on the current state and chosen action. The MCTS algorithm consists of a loop of four steps:

- **Selection:** start from the root node, interleave action selection and sampling the next state (tree node) until a leaf node is reached
- **Expansion:** expand the tree by adding a new edge (action) to the leaf node and sample a next state (new leaf node)
- **Simulation:** rollout from the reached state to the end of the episode using random actions or a heuristic
- **Backup:** update the nodes backward along the trajectory starting from the end of the episode until the root node according to the rewards collected

In the next subsection, we discuss UCB algorithm and its extension to UCT.

2.3.2. Upper Confidence bound for Trees

In this work, we consider the MCTS algorithm UCT (Upper Confidence bounds for Trees) [12], an extension of the well-known UCB1 [45] multi-armed bandit algorithm. UCB1 chooses the arm (action a) using

$$a = \arg\max_{i \in \{1 \ldots K\}} \overline{X}_{i,T_i(n-1)} + C\sqrt{\frac{\log n}{T_i(n-1)}}. \tag{2.1}$$

where $T_i(n) = \sum_{t=1}^{n} \mathbf{1}\{t = i\}$ is the number of times arm i is played up to time n. $\overline{X}_{i,T_i(n-1)}$ denotes the average reward of arm i up to time $n-1$ and $C = \sqrt{2}$ is an exploration constant. In UCT, each node is a separate bandit, where the arms correspond to the actions, and the payoff is the reward of the episodes starting from them. In the

backup phase, value is backed up recursively from the leaf node to the root as

$$\overline{X}_n = \sum_{i=1}^{K} \left(\frac{T_i(n)}{n}\right) \overline{X}_{i,T_i(n)}. \tag{2.2}$$

[12] proved that UCT converges in the limit to the optimal policy.

2.3.3. Power Mean

In this work, we introduce a novel way of estimating the expected value of a bandit arm ($\overline{X}_{i,T_i(n-1)}$ in (2.1)) in MCTS. For this purpose, we will use the *power mean* [66], an operator belonging to the family of functions for aggregating sets of numbers, that includes as special cases the Pythagorean means (arithmetic, geometric, and harmonic means):

Definition 1. *For a sequence of positive numbers $X = (X_1, ..., X_n)$ and positive weights $w = (w_1, ..., w_n)$, the power mean of order p (p is an extended real number) is defined as*

$$M_n^{[p]}(X, w) = \left(\frac{\sum_{i=1}^{n} w_i X_i^p}{\sum_{i=1}^{n} w_i}\right)^{\frac{1}{p}}. \tag{2.3}$$

With $p = 1$ we obtain the weighted arithmetic mean. With $p \to 0$ we have the geometric mean, and with $p = -1$ we have the harmonic mean [66] Furthermore, we get [66]

$$\mathrm{M}_n^{[-\infty]}(X, w) = \lim_{p \to -\infty} \mathrm{M}_n^{[p]}(X, w) = \mathrm{Min}(X_1, ..., X_n), \tag{2.4}$$

$$\mathrm{M}_n^{[+\infty]}(X, w) = \lim_{p \to +\infty} \mathrm{M}_n^{[p]}(X, w) = \mathrm{Max}(X_1, ..., X_n), \tag{2.5}$$

The weighted arithmetic mean lies between $\mathrm{Min}(X_1, ..., X_n)$ and $\mathrm{Max}(X_1, ..., X_n)$. Moreover, the following lemma shows that $\mathrm{M}_n^{[p]}(X, w)$ is an increasing function.

Lemma 1. *$M_n^{[p]}(X, w)$ is an increasing function meaning that*

$$M_n^{[1]}(X, w) \leq M_n^{[q]}(X, w) \leq M_n^{[p]}(X, w), \forall p \geq q \geq 1 \tag{2.6}$$

For the proof, see [66].

2.4. Power Mean Backup

As previously described, it is well known that performing backups using the average of the rewards results in an underestimate of the true value of the node, while using the maximum results in an overestimate of it [32]. Usually, the average backup is used when the number of simulations is low, for a conservative update of the nodes due to the lack of samples; on the other hand, the maximum operator is favoured when the number of simulations is high. We address this problem proposing a novel backup operator for UCT based on the power mean (Equation 2.3)

$$\overline{X}_n(p) = \left(\sum_{i=1}^{K} \left(\frac{T_i(n)}{n} \right) \overline{X}_{i,T_i(n)}^{p} \right)^{\frac{1}{p}} . \tag{2.7}$$

This way, we bridge the gap between the average and maximum estimators with the purpose of getting the advantages of both. We call our approach Power-UCT and describe it in more detail in the following.

2.4.1. Power-UCT

Indeed, the Power-UCT pseudocode shown in Algorithm 2.1 is almost identical to the UCT one, with only few differences highlighted for clarity. MCTS has two types of nodes: V_Nodes corresponding to state-values, and Q_Nodes corresponding to state-action values. An action is taken from the V_Node of the current state leading to the respective Q_Node, then it leads to the V_Node of the reached state. The introduction of our novel backup operator in UCT does not require major changes to the algorithm. In Power-UCT, the expansion of nodes and the rollouts are done in the same way as UCT, and the only difference is the way the backup of returns from Q_nodes to V_nodes is computed. In particular, while UCT computes the average of the returns, Power-UCT uses a power mean of them. Note that our algorithm could be applied to several bandit based enhancements of UCT, but for simplicity we only focus on UCT. For each state s, the backup value of corresponding V_node is

$$V(s) \leftarrow \left(\sum_{a} \frac{n(s,a)}{N(s)} Q(s,a)^p \right)^{\frac{1}{p}} \tag{2.8}$$

```
s: state
a: action
N(s): number of simulations of V_Node of state s
n(s,a): number of simulations of Q_Node of state s and action a
V(s): Value of V_Node at state s. Default is 0
Q(s,a): Value of Q_Node at state s, action a. Default is 0
τ(s,a): transition function
γ: discount factor
ϵ > 0:

R = Rollout(s, depth)
    if γ^depth < ϵ then
        return 0
    a ~ π_Rollout(.)
    (s', r) ~ τ(s,a)
    return r + γRollout (s', depth + 1)

a = SelectAction(s)
    return arg max_a Q(s,a) + C sqrt(log N(s) / n(s,a))

a = Search(s)
    while Time remaining do
        SimulateV (s, 0)
    return SelectAction(s)

SimulateV(s, depth)
    a = SelectAction (s)
    SimulateQ (s, a, depth)
    N(s) ← N(s) + 1
    V(s) ← ( Σ_a (n(s,a)/N(s)) Q(s,a)^p )^(1/p)

SimulateQ(s, a, depth)
    (s', r) ~ τ(s,a)
    if Node s' not expanded then
        Rollout(s', depth)
    else
        SimulateV (s', depth + 1)
    n(s,a) ← n(s,a) + 1
    Q(s,a) ← ((Σ_a r_{s,a}) + γ.Σ_{s'} N(s').V(s')) / n(s,a)

MainLoop
    while resource budget remains do
        a = Search(s)
```

Algorithm 2.1: Pseudocode of Power-UCT.

where $N(s)$ is the number of visits to state s, $n(s,a)$ is the number of visits of action a in state s. On the other hand, the backup value of Q_nodes is

$$Q(s,a) \leftarrow \frac{(\sum_a r_{s,a}) + \gamma . \sum_{s'} N(s').V(s')}{n(s,a)} \tag{2.9}$$

where γ is the discount factor, s' is the next state after taking action a from state s, and $r_{s,a}$ is the reward obtained executing action a in state s.

2.5. Theoretical analysis

In this section, we show that Power-UCT can smoothly adapt to all theorems of UCT [12]. The following results can be seen as a generalization of the results for UCT, as we consider a generalized mean instead of a standard mean as the backup operator. Our main results are Theorem 6 and Theorem 7, which respectively prove the convergence of failure probability at the root node, and derive the bias of power mean estimated payoff. In order to prove them, we start with Theorem 1 to show the concentration of power mean with respect to i.i.d random variables X. Subsequently, Theorem 2 shows the upper bound of the expected number of times when a suboptimal arm is played. Theorem 3 bounds the expected error of the power mean estimation. Theorem 4 shows the lower bound of the number of times any arm is played. Theorem 5 shows the concentration of power mean backup around its mean value at each node in the tree.

Theorem 1. *If $X_1, X_2, ..., X_n$ are independent with* **Pr**$(0 \leq X_i \leq 1) = 1$ *then for any $\epsilon > 0$, $p \geq 1, \exists C_p > 0$ that*

$$\mathbf{Pr}\left(\left|\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}} - \mathbb{E}\left[\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}}\right]\right| > \epsilon\right) \leq 2\exp\left(-C_p n \epsilon^2\right)$$

Theorem 1 is derived using an estimation of variance of power mean operator, and Chernoff's inequality. Note that this result can be considered a generalization of the well-known Hoeffding inequality to power mean. Next, given i.i.d. random variables X_{it} (t=1,2,...) as the payoff sequence at any internal leaf node of the tree, we assume the expectation of the payoff exists and let $\mu_{in} = \mathbb{E}[\overline{X_{in}}]$. We assume the power mean reward drifts as a function of time and converges only in the limit, which means that

$$\mu_i = \lim_{n \to \infty} \mu_{in}.$$

Let $\delta_{in} = \mu_i - \mu_{in}$ which also means that

$$\lim_{n\to\infty} \delta_{in} = 0.$$

From now on, let $*$ be the upper index for all quantities related to the optimal arm. By assumption, the rewards lie between 0 and 1. Let's start with an assumption:

Assumption 1. *Fix $1 \leq i \leq K$. Let $\{F_{it}\}_t$ be a filtration such that$\{X_{it}\}_t$ is $\{F_{it}\}$-adapted and $X_{i,t}$ is conditionally independent of $F_{i,t+1}, F_{i,t+2}, \ldots$ given $F_{i,t-1}$. Then $0 \leq X_{it} \leq 1$ and the limit of $\mu_{in} = \mathbb{E}[\overline{X_{in}}(p)]$ exists, Further, we assume that there exists a constant $C > 0$ and an integer N_c such that for $n > N_c$, for any $\delta > 0$, $\triangle_n(\delta) = C\sqrt{n\log(1/\delta)}$, the following bounds hold*

$$\mathbf{Pr}(\overline{X}_{in}(p) \geq \mathbb{E}[\overline{X}_{in}(p)] + \triangle_n(\delta)/n) \leq \delta, \tag{2.10}$$

$$\mathbf{Pr}(\overline{X}_{in}(p) \leq \mathbb{E}[\overline{X}_{in}(p)] - \triangle_n(\delta)/n) \leq \delta. \tag{2.11}$$

Under Assumption 1, a suitable choice for the bias sequence $c_{t,s}$ is given by

$$c_{t,s} = 2C\sqrt{\frac{\log t}{s}}. \tag{2.12}$$

where C is an exploration constant.
Next, we derive Theorems 2, 3, and 4 following the derivations in [12]. First, from Assumption 1, we derive an upper bound on the error for the expected number of times suboptimal arms are played.

Theorem 2. *Consider UCB1 (using power mean estimator) applied to a non-stationary problem where the pay-off sequence satisfies Assumption 1 and where the bias sequence, $c_{t,s}$ defined in (2.12). Fix $\epsilon \geq 0$. Let $T_k(n)$ denote the number of plays of arm k. Then if k is the index of a suboptimal arm then Each sub-optimal arm k is played in expectation at most*

$$\mathbb{E}[T_k(n)] \leq \frac{16C^2 \ln n}{(1-\epsilon)^2 \triangle_k^2} + A(\epsilon) + N_c + \frac{\pi^2}{3} + 1. \tag{2.13}$$

Next, we derive our version of Theorem 3 in [12], which computes the upper bound of the difference between the value backup of an arm with μ^* up to time n.

Theorem 3. *Under the assumptions of Theorem 2,*

$$\left|\mathbb{E}\left[\overline{X}_n(p)\right] - \mu^*\right| \leq |\delta_n^*| + \mathcal{O}\left(\frac{K(C^2 \log n + N_0)}{n}\right)^{\frac{1}{p}}.$$

A lower bound for the times choosing any arm follows

Theorem 4. ***(Lower Bound)*** *Under the assumptions of Theorem 2, there exists some positive constant ρ such that for all arms k and n, $T_k(n) \geq \lceil \rho \log(n) \rceil$.*

For deriving the concentration of estimated payoff around its mean, we modify Lemma 14 in [12] for power mean: in the proof, we first replace the partial sums term with a partial mean term and modify the following equations accordingly. The partial mean term can then be easily replaced by a partial power mean term and we get

Theorem 5. *Fix an arbitrary $\delta \leq 0$ and fix $p \geq 1$, let $\triangle_n = (\frac{9}{4})^{p-1}(9\sqrt{\frac{1}{C_p} n \log(2/\delta)})$. Let n_0 be such that*

$$\sqrt{n_0} \leq \mathcal{O}(K(C^2 \log n_0 + N_0(1/2))). \tag{2.14}$$

Then for any $n \geq n_0$, under the assumptions of Theorem 2, the following bounds hold true

$$\mathbf{Pr}(\overline{X}_n(p) \geq \mathbb{E}[\overline{X}_n(p)] + (\triangle_n/n)^{\frac{1}{p}}) \leq \delta \tag{2.15}$$

$$\mathbf{Pr}(\overline{X}_n(p) \leq \mathbb{E}[\overline{X}_n(p)] - (\triangle_n/n)^{\frac{1}{p}}) \leq \delta \tag{2.16}$$

Using The Hoeffding-Azuma inequality for Stopped Martingales Inequality (Lemma 10 in [12]), under Assumption 1 and the result from Theorem 4 we get

Theorem 6. ***(Convergence of Failure Probability)*** *Under the assumptions of Theorem 2, it holds that*

$$\lim_{t \to \infty} \mathbf{Pr}(I_t \neq i^*) = 0. \tag{2.17}$$

And finally, the following is our main result showing the expected payoff of our Power-UCT.

Theorem 7. *Consider algorithm Power-UCT running on a game tree of depth D, branching factor K with stochastic payoff at the leaves. Assume that the payoffs lie in the interval [0,1]. Then the bias of the estimated expected payoff, $\overline{X_n}$, is $\mathcal{O}(KD(\log(n)/n)^{\frac{1}{p}} + K^D(1/n)^{\frac{1}{p}})$. Further, the failure probability at the root convergences to zero as the number of samples grows to infinity.*

Proof. (Sketch) As for UCT [12], the proof is done by induction on D. When $D = 1$, Power-UCT corresponds to UCB1 with average mean backup at the leaf node, and the

proof of convergence follows as the result of Hoeffding's inequality, the expected payoff is guaranteed directly from Theorem 1, Theorem 3 and Theorem 6. Now we assume that the result holds up to depth $D-1$ and consider the tree of depth D. Running Power-UCT on root node is equivalent to UCB1 on non-stationary bandit settings, but with power mean backup. The error bound of running Power-UCT for the whole tree is the sum of payoff at root node with payoff starting from any node i after the first action chosen from root node until the end. This payoff by induction at depth $D-1$ in addition to the bound from Theorem 3 when the drift-conditions are satisfied, and with straightforward algebra, we can compute the payoff at the depth D, in combination with Theorem 6. Since our induction hypothesis holds for all nodes at a distance of one node from the root, the proof is finished by observing that Theorem 3 and Theorem 5 do indeed ensure that the drift conditions are satisfied. This completes our proof of the convergence of Power-UCT. Interestingly, the proof guarantees the convergence for any finite value of p. □

2.6. Experiments

In this section, we aim to answer the following questions empirically: Does the Power Mean offer higher performance in MDP and POMDP MCTS tasks than the regular Mean? How does the value of p influence the overall performance? How does Power-UCT, our MCTS algorithm based on the Power Mean, compare to state-of-the-art methods in tree-search? We chose the recent MENTS algorithm [31] as a representative state-of-the-art method.

For MENTS we find the best combination of the two hyper-parameters by grid search. In MDP tasks, we find the UCT exploration constant using grid search. For Power-UCT, we find the p-value by increasing it until performance starts to decrease.

2.6.1. *FrozenLake*

For MDPs, we consider the well-known *FrozenLake* problem as implemented in OpenAI Gym [67]. In this problem, the agent needs to reach a goal position in an 8x8 ice grid-world while avoiding falling into the water by stepping onto unstable spots. The challenge of this task arises from the high-level of stochasticity, which makes the agent only move towards the intended direction one-third of the time, and into one of the two tangential directions the rest of it. Reaching the goal position yields a reward of 1, while all other outcomes (reaching the time limit or falling into the water) yield a reward of zero. As can

Table 2.1.: Mean and two times standard deviation of the success rate, over 500 evaluation runs, of UCT, Power-UCT and MENTS in *FrozenLake* from OpenAI Gym. The top row of each table shows the number of simulations used for tree-search at each time step.

Algorithm	4096	16384	65536	262144
UCT	0.08 ± 0.02	0.23 ± 0.04	0.54 ± 0.05	0.69 ± 0.04
p=2.2	0.12 ± 0.03	0.32 ± 0.04	$\mathbf{0.62 \pm 0.04}$	$\mathbf{0.81 \pm 0.03}$
p=max	0.10 ± 0.03	0.36 ± 0.04	0.55 ± 0.04	0.69 ± 0.04
MENTS	$\mathbf{0.28 \pm 0.04}$	$\mathbf{0.46 \pm 0.04}$	$\mathbf{0.62 \pm 0.04}$	0.74 ± 0.04

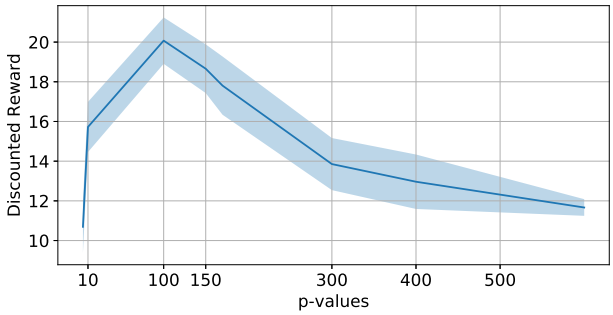


Figure 2.1.: Evaluating Power-UCT w.r.t. different p-values: The mean discounted total reward at 65536 simulations (shaded area denotes standard error) over 100 evaluation runs.

be seen in Table 2.1, Power-UCT improves the performance compared to UCT. Power-UCT outperforms MENTS when the number of simulations increases.

2.6.2. Copy Environment

Now, we aim to answer the question of how Power-UCT scales to domains with a large number of actions (high branching factor). We use the OpenAI gym Copy environment where the agent needs to copy the characters on an input band to an output band. The agent can move and read the input band at every time-step and decide to write a character from an alphabet to the output band. Hence, the number of actions scales with the size of the alphabet.

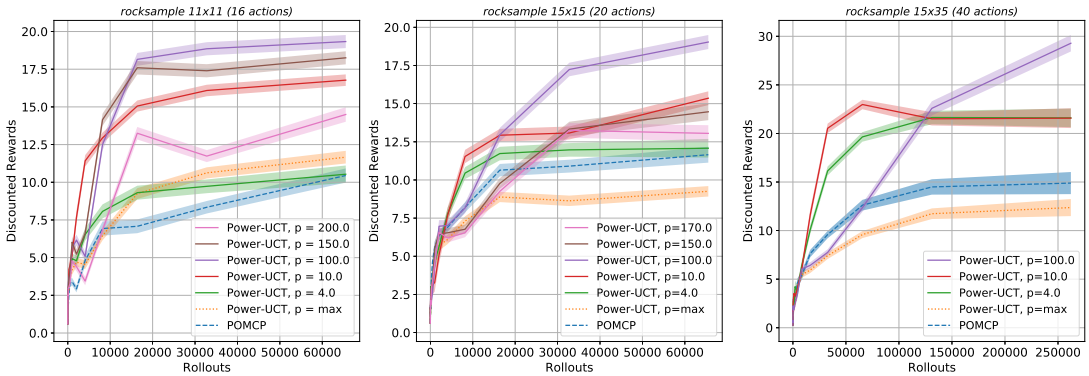


Figure 2.2.: Performance of Power-UCT compared to UCT in *rocksample*. The mean of total discounted reward over 1000 evaluation runs is shown by thick lines while the shaded area shows standard error.

Contrary to the previous experiments, there is only one initial run of tree-search and afterwards, no re-planning between two actions occurs. Hence, all actions are selected according to the value estimates from the initial search. The results in Tables 2.2 and 2.2 show that Power-UCT allows solving the task much quicker than regular UCT. Furthermore, we observe that MENTS and Power-UCT for $p = \infty$ exhibit larger variance compared to Power-UCT with a finite value of p and are not able to reliably solve the task, as they do not reach the maximum reward of 40 with 0 standard deviation.

2.6.3. Rocksample and PocMan

In POMDP problems, we compare Power-UCT against the POMCP algorithm [68] which is a standard UCT algorithm for POMDPs. Since the state is not fully observable in POMDPs, POMCP assigns a unique action-observation history, which is a sufficient statistic for optimal decision making in POMDPs, instead of the state, to each tree node. Similar to fully observable UCT, POMCP chooses actions using the UCB1 bandit. Therefore, we modify POMCP to use the power mean identically to how we modified fully observable UCT and get a POMDP version of Power-UCT. We also modify POMCP similarly for the MENTS approach. Next, we discuss the evaluation of the POMDP based Power-UCT, MENTS, and POMCP, in the *rocksample* and *pocman* environments [68].

Rocksample. The *rocksample (n,k)* ([69]) simulates a Mars explorer robot in an n *x* n grid containing *k* rocks. The task is to determine which rocks are valuable using a

Table 2.2.: Mean and two times standard deviation of discounted total reward, over 100 evaluation runs, of UCT, Power-UCT and MENTS in the copy environment with 144 actions (top) and 200 actions (bottom). Top row: number of simulations at each time step.

Algorithm	512	2048	8192	32768
UCT	2.6 ± 0.98	$9. \pm 1.17$	34.66 ± 1.68	$\mathbf{40. \pm 0.}$
$p = 3$	3.24 ± 1.17	$\mathbf{12.35 \pm 1.14}$	$\mathbf{40. \pm 0.}$	$\mathbf{40. \pm 0.}$
$p = \max$	2.56 ± 1.48	9.55 ± 3.06	37.52 ± 5.11	39.77 ± 0.84
MENTS	$\mathbf{3.26 \pm 1.32}$	11.96 ± 2.94	39.37 ± 1.15	39.35 ± 0.95

(a) 144 Actions

Algorithm	512	2048	8192	32768
UCT	1.98 ± 0.63	6.43 ± 1.36	24.5 ± 1.56	$\mathbf{40. \pm 0.}$
$p = 3$	$\mathbf{2.55 \pm 0.99}$	$\mathbf{9.11 \pm 1.41}$	$\mathbf{36.02 \pm 1.72}$	$\mathbf{40. \pm 0.}$
$p = \max$	2.03 ± 1.37	6.99 ± 2.51	27.89 ± 4.12	39.93 ± 0.51
MENTS	2.44 ± 1.34	8.86 ± 2.65	34.63 ± 5.6	39.42 ± 0.99

(b) 200 Actions

long range sensor, take samples of valuable rocks and finally leave the map to the east. There are $k + 5$ actions where the agent can move in four directions (North, South, East, West), sample a rock, or sense one of the k rocks. Rocksample requires strong exploration to find informative actions which do not yield immediate reward but may yield high long term reward. We use three variants with a different number of actions: *rocksample* (11,11), *rocksample* (15,15), *rocksample* (15,35) and set the exploration constant as in [68] to the difference of the maximum and minimum immediate reward. In Fig. 2.2, Power-UCT outperforms POMCP for almost all values of p. For sensitivity analysis, Fig. 2.1 shows the performance of Power-UCT in *rocksample* (11x11) for different p-values at 65536 simulations. Fig. 2.1 suggests that at least in *rocksample* finding a good p-value is straightforward. Fig. 2.3 shows that Power-UCT significantly outperforms MENTS in *rocksample* (11,11). A possible explanation for the strong difference in performance between MENTS and Power-UCT is that MENTS may not explore sufficiently in this task. However, this would require more in depth analysis of MENTS.

Pocman. We further evaluate our algorithm in the *pocman* problem [68]. In *pocman*, an agent called PocMan must travel in a maze of size (17x19) by only observing the local neighborhood in the maze. PocMan tries to eat as many food pellets as possible.

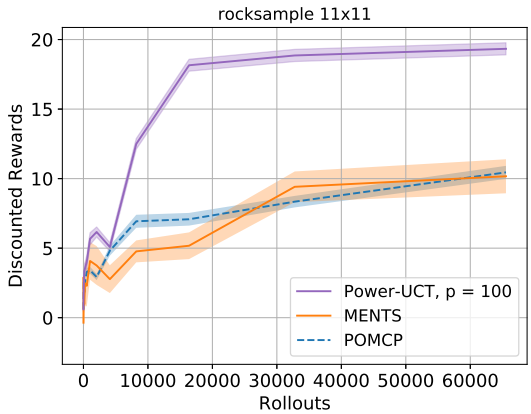


Figure 2.3.: Performance of Power-UCT compared to UCT and MENTS in *rocksample* 11x11. The mean of discounted total reward over 1000 evaluation runs is shown by thick lines while the shaded area shows standard error.

Four ghosts try to kill PocMan. After moving initially randomly the ghosts start to follow directions with a high number of food pellets more likely. If PocMan eats a power pill, he is able to eat ghosts for 15 time steps. PocMan receives a reward of -1 at each step he travels, $+10$ for eating each food pellet, $+25$ for eating a ghost and -100 for dying. The *pocman* problem has 4 actions, 1024 observations, and approximately 10^{56} states. Table 2.3 shows that Power-UCT and MENTS outperform POMCP.

2.7. Conclusion

We proposed to use power mean as a novel backup operator in MCTS, and derived a variant of UCT based on this operator, which we call Power-UCT. We theoretically prove the convergence of Power-UCT to the optimal value, given that the value computed by the power mean lies between the average and the maximum. The empirical evaluation on stochastic MDPs and POMDPs, shows the advantages of Power-UCT w.r.t. other baselines.

Possible future work includes the proposal of a theoretically justified or heuristic approach to adapt the greediness of power mean. Moreover, we are interested in analysing the bias and variance of the power mean estimator, or analysing the regret bound of Power-UCT in MCTS. Furthermore, we plan to test our methodology in more challenging Reinforcement Learning problems through the use of parametric function approximators, e.g. neural networks.

Table 2.3.: Discounted total reward in *pocman* for the comparison methods. Mean $\pm$ standard error are computed from 1000 simulations except in MENTS where we ran 100 simulations.

	1024	4096	16384	65536
POMCP	30.89 ± 1.4	33.47 ± 1.4	33.44 ± 1.39	32.36 ± 1.6
$p = max$	14.82 ± 1.52	14.91 ± 1.52	14.34 ± 1.52	14.98 ± 1.76
$p = 10$	29.14 ± 1.61	35.26 ± 1.56	44.14 ± 1.60	$\mathbf{53.30 \pm 1.46}$
$p = 30$	28.78 ± 1.44	33.92 ± 1.56	42.45 ± 1.54	49.66 ± 1.70
MENTS	$\mathbf{54.08 \pm 3.20}$	$\mathbf{55.37 \pm 3.0}$	$\mathbf{53.90 \pm 2.86}$	51.03 ± 3.36

3. Convex Regularization in Monte-Carlo Tree Search

"Only entropy comes easy."

— *Anton Chekhov*

Monte-Carlo planning and Reinforcement Learning (RL) are essential to sequential decision making. The recent AlphaGo and AlphaZero algorithms have shown how to successfully combine these two paradigms to solve large-scale sequential decision problems. These methodologies exploit a variant of the well-known UCT algorithm to trade off the exploitation of good actions and the exploration of unvisited states, but their empirical success comes at the cost of poor sample efficiency and high computation time. In this chapter, we overcome these limitations by introducing the use of convex regularization in Monte-Carlo Tree Search (MCTS) to drive exploration efficiently and improve policy updates. First, we introduce a unifying theory on using generic convex regularizers in MCTS, deriving the first regret analysis of regularized MCTS and showing that it guarantees an exponential convergence rate. Second, we exploit our theoretical framework to introduce novel regularized backup operators for MCTS based on the relative entropy of the policy update and, more importantly, on the Tsallis entropy of the policy, for which we prove superior theoretical guarantees. Third, we empirically verify the consequence of our theoretical results on a toy problem. Finally, we show how our framework can easily be incorporated in AlphaGo, and we empirically show the superiority of convex regularization, w.r.t. representative baselines, on well-known RL problems across several Atari games.

3.1. Introduction

Monte-Carlo Tree Search (MCTS) is a well-known algorithm to solve decision-making problems through the combination of Monte-Carlo planning and an incremental tree

structure [32]. MCTS provides a principled approach for trading off between exploration and exploitation in sequential decision making. Moreover, recent advances have shown how to enable MCTS in continuous and large problems [28], [70]. Most remarkably, AlphaGo [28] and AlphaZero [14], [71] couple MCTS with neural networks trained using Reinforcement Learning (RL) [29] methods, e.g., Deep Q-Learning [30], to speed up learning of large scale problems. In particular, a neural network is used to compute value function estimates of states as a replacement for time-consuming Monte-Carlo rollouts, and another neural network is used to estimate policies as a probability prior to the therein introduced PUCT action selection strategy, a variant of well-known UCT sampling strategy commonly used in MCTS for exploration [12]. Despite AlphaGo and AlphaZero achieving state-of-the-art performance in games with high branching factor like Go [28] and Chess [14], both methods suffer from poor sample-efficiency, mostly due to the polynomial convergence rate of PUCT [31]. This problem, combined with the high computation time to evaluate the deep neural networks, significantly hinder the applicability of both methodologies.

In this chapter, we provide a theory of the use of convex regularization in MCTS, which proved to be an efficient solution for driving exploration and stabilizing learning in RL [72]–[75]. In particular, we show how a regularized objective function in MCTS can be seen as an instance of the Legendre-Fenchel transform, similar to previous findings on the use of duality in RL [76]–[78] and game theory [79], [80]. Establishing our theoretical framework, we can derive the first regret analysis of regularized MCTS, and prove that a generic convex regularizer guarantees an exponential convergence rate to the solution of the regularized objective function, which improves on the polynomial rate of PUCT. These results provide a theoretical ground for the use of arbitrary entropy-based regularizers in MCTS until now limited to maximum entropy [31], among which we specifically study the relative entropy of policy updates, drawing on similarities with trust-region and proximal methods in RL [72], [81], and the Tsallis entropy, used for enforcing the learning of sparse policies [82]. Moreover, we provide an empirical analysis of the toy problem introduced in [31] to evince the practical consequences of our theoretical results for each regularizer. Finally, we empirically evaluate the proposed operators in AlphaGo, on several Atari games, confirming the benefit of convex regularization in MCTS, and in particular the superiority of Tsallis entropy w.r.t. other regularizers.

3.2. Preliminaries

Background knowledge on Markov Decision Processes, Monte-Carlo Tree Search and the state-of-the-art method Upper Confidence bounds for Trees (UCT) will be presented in this section.

3.2.1. Markov Decision Processes

We consider the classical definition of a finite-horizon Markov Decision Process (MDP) as a 5-tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the finite discrete action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition kernel, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi \in \Pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a probability distribution of the event of executing an action a in a state s. A policy π induces a value function corresponding to the expected cumulative discounted reward collected by the agent when executing action a in state s, and following the policy π thereafter $Q^\pi(s, a) \triangleq \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{i+k+1} | s_i = s, a_i = a, \pi\right]$, where r_{i+1} is the reward obtained after the i-th transition. An MDP is solved finding the optimal policy π^*, which is the policy that maximizes the expected cumulative discounted reward. The optimal policy satisfies the optimal Bellman equation [83] $Q^*(s, a) \triangleq \int_{\mathcal{S}} \mathcal{P}(s'|s, a) \left[\mathcal{R}(s, a, s') + \gamma \max_{a'} Q^*(s', a')\right] ds'$, and is the fixed point of the optimal Bellman operator

$\mathcal{T}^* Q(s, a) \triangleq \int_{\mathcal{S}} \mathcal{P}(s'|s, a) \left[\mathcal{R}(s, a, s') + \gamma \max_{a'} Q(s', a')\right] ds'$.

We define the Bellman operator under the policy π as

$\mathcal{T}_\pi Q(s, a) \triangleq \int_{\mathcal{S}} \mathcal{P}(s'|s, a) \left[\mathcal{R}(s, a, s') + \gamma \int_{\mathcal{A}} \pi(a'|s') Q(s', a') da'\right] ds'$, the optimal value function $V^*(s) \triangleq \max_{a \in \mathcal{A}} Q^*(s, a)$, and the value function under the policy π as $V^\pi(s) \triangleq \max_{a \in \mathcal{A}} Q^\pi(s, a)$.

3.2.2. Monte-Carlo Tree Search and Upper Confidence bounds for Trees

Monte-Carlo Tree Search (MCTS) is a planning strategy based on a combination of Monte-Carlo sampling and tree search to solve MDPs. MCTS builds a tree where the nodes are the visited states of the MDP, and the edges are the actions executed in each state. MCTS converges to the optimal policy [12], [31], iterating over a loop composed of four steps:

1. **Selection:** starting from the root node, a *tree-policy* is executed to navigate the tree until a node with unvisited children, i.e. expandable node, is reached;

2. **Expansion:** the reached node is expanded according to the tree policy;
3. **Simulation:** run a rollout, e.g. Monte-Carlo simulation, from the visited child of the current node to the end of the episode;
4. **Backup:** use the collected reward to update the action-values $Q(\cdot)$ of the nodes visited in the trajectory from the root node to the expanded node.

The tree-policy used to select the action to execute in each node needs to balance the use of already known good actions, and the visitation of unknown states. The Upper Confidence bounds for Trees (UCT) sampling strategy [12] extends the use of the well-known UCB1 sampling strategy for multi-armed bandits [45], to MCTS. Considering each node corresponding to a state $s \in \mathcal{S}$ as a different bandit problem, UCT selects an action $a \in \mathcal{A}$ applying an upper bound to the action-value function

$$\text{UCT}(s, a) = Q(s, a) + \epsilon \sqrt{\frac{\log N(s)}{N(s, a)}}, \tag{3.1}$$

where $N(s, a)$ is the number of executions of action a in state s, $N(s) = \sum_a N(s, a)$, and ϵ is a constant parameter to tune exploration. UCT asymptotically converges to the optimal action-value function Q^*, for all states and actions, with the probability of executing a suboptimal action at the root node approaching 0 with a polynomial rate $O(\frac{1}{t})$, for a simulation budget t [12], [31].

3.3. Regularized Monte-Carlo Tree Search

The success of RL methods based on entropy regularization comes from their ability to achieve state-of-the-art performance in decision making and control problems, while enjoying theoretical guarantees and ease of implementation [72], [74], [82]. However, the use of entropy regularization in MCTS is still mostly unexplored, although its advantageous exploration and value function estimation would be desirable to reduce the detrimental effect of high-branching factor in AlphaGo and AlphaZero. To the best of our knowledge, the MENTS algorithm [31] is the first and only method to combine MCTS and entropy regularization. In particular, MENTS uses a maximum entropy regularizer in AlphaGo, proving an exponential convergence rate to the solution of the respective softmax objective function and achieving state-of-the-art performance in some Atari games [84]. In the following, motivated by the success in RL and the promising results of MENTS, we derive a unified theory of regularization in MCTS based on the Legendre-Fenchel transform [77],

that generalizes the use of maximum entropy of MENTS to an arbitrary convex regularizer. Notably, our theoretical framework enables to rigorously motivate the advantages of using maximum entropy and other entropy-based regularizers, such as relative entropy or Tsallis entropy, drawing connections with their RL counterparts TRPO [72] and Sparse DQN [82], as MENTS does with Soft Actor-Critic (SAC) [74].

3.3.1. Legendre-Fenchel transform

Consider an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$, as previously defined. Let $\Omega : \Pi \to \mathbb{R}$ be a strongly convex function. For a policy $\pi_s = \pi(\cdot|s)$ and $Q_s = Q(s, \cdot) \in \mathbb{R}^{\mathcal{A}}$, the Legendre-Fenchel transform (or convex conjugate) of Ω is $\Omega^* : \mathbb{R}^{\mathcal{A}} \to \mathbb{R}$, defined as

$$\Omega^*(Q_s) \triangleq \max_{\pi_s \in \Pi_s} \mathcal{T}_{\pi_s} Q_s - \tau \Omega(\pi_s), \tag{3.2}$$

where the temperature τ specifies the strength of regularization. Among the several properties of the Legendre-Fenchel transform, we use the following [76], [77].

Proposition 1. *Let Ω be strongly convex.*

- *Unique maximizing argument: $\nabla\Omega^*$ is Lipschitz and satisfies*

$$\nabla\Omega^*(Q_s) = \arg\max_{\pi_s \in \Pi_s} \mathcal{T}_{\pi_s} Q_s - \tau \Omega(\pi_s). \tag{3.3}$$

- *Boundedness: if there are constants L_Ω and U_Ω such that for all $\pi_s \in \Pi_s$, we have $L_\Omega \leq \Omega(\pi_s) \leq U_\Omega$, then*

$$\max_{a \in \mathcal{A}} Q_s(a) - \tau U_\Omega \leq \Omega^*(Q_s) \leq \max_{a \in \mathcal{A}} Q_s(a) - \tau L_\Omega. \tag{3.4}$$

- *Contraction: for any $Q_1, Q_2 \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$*

$$\| \Omega^*(Q_1) - \Omega^*(Q_2) \|_\infty \leq \gamma \| Q_1 - Q_2 \|_\infty . \tag{3.5}$$

Note that if $\Omega(\cdot)$ is strongly convex, $\tau\Omega(\cdot)$ is also strongly convex; thus all the properties shown in Proposition 1 still hold[1].
Solving equation (2) leads to the solution of the optimal primal policy function $\nabla\Omega^*(\cdot)$.

[1] Other works use the same formula, e.g. Equation (3.2) in [85].

Since $\Omega(\cdot)$ is strongly convex, the dual function $\Omega^*(\cdot)$ is also convex. One can solve the optimization problem (3.2) in the dual space [86] as

$$\Omega(\pi_s) = \max_{Q_s \in \mathbb{R}^{\mathcal{A}}} \mathcal{T}_{\pi_s} Q_s - \tau \Omega^*(Q_s) \tag{3.6}$$

and find the solution of the optimal dual value function as $\Omega^*(\cdot)$. Note that the Legendre-Fenchel transform of the value conjugate function is the convex function Ω, i.e. $\Omega^{**} = \Omega$. In the next section, we leverage on this primal-dual connection based on the Legendre-Fenchel transform as both conjugate value function and policy function, to derive the regularized MCTS backup and tree policy.

3.3.2. Regularized backup and tree policy

In MCTS, each node of the tree represents a state $s \in \mathcal{S}$ and contains a visitation count $N(s, a)$. Given a trajectory, we define $n(s_T)$ as the leaf node corresponding to the reached state s_T. Let $s_0, a_0, s_1, a_1 ..., s_T$ be the state action trajectory in a simulation, where $n(s_T)$ is a leaf node of $\mathcal{T}$. Whenever a node $n(s_T)$ is expanded, the respective action values (Equation 3.7) are initialized as $Q_\Omega(s_T, a) = 0$, and $N(s_T, a) = 0$ for all $a \in \mathcal{A}$. For all nodes in the trajectory, the visitation count is updated by $N(s_t, a_t) = N(s_t, a_t) + 1$, and the action-values by

$$Q_\Omega(s_t, a_t) = \begin{cases} r(s_t, a_t) + \gamma\rho & \text{if } t = T \\ r(s_t, a_t) + \gamma\Omega^*(Q_\Omega(s_{t+1})/\tau) & \text{if } t < T \end{cases} \tag{3.7}$$

where $Q_\Omega(s_{t+1}) \in \mathbb{R}^{\mathcal{A}}$ with $Q_\Omega(s_{t+1}, a), \forall a \in \mathcal{A}$, and ρ is an estimate returned from an evaluation function computed in s_T, e.g. a discounted cumulative reward averaged over multiple rollouts, or the value-function of node $n(s_{T+1})$ returned by a value-function approximator, e.g. a neural network pretrained with deep Q-learning [30], as done in [28], [31]. We revisit the E2W sampling strategy limited to maximum entropy regularization [31] and, through the use of the convex conjugate in Equation (3.7), we derive a novel sampling strategy that generalizes to any convex regularizer

$$\pi_t(a_t|s_t) = (1 - \lambda_{s_t})\nabla\Omega^*(Q_\Omega(s_t)/\tau)(a_t) + \frac{\lambda_{s_t}}{|\mathcal{A}|}, \tag{3.8}$$

where $\lambda_{s_t} = \epsilon|\mathcal{A}|/\log(\sum_a N(s_t, a)+1)$ with $\epsilon > 0$ as an exploration parameter, and $\nabla\Omega^*$ depends on the measure in use (see Table 3.1 for maximum, relative, and Tsallis entropy). We call this sampling strategy *Extended Empirical Exponential Weight* (E3W) to highlight the

extension of E2W from maximum entropy to a generic convex regularizer. E3W defines the connection to the duality representation using the Legendre-Fenchel transform, that is missing in E2W. Moreover, while the Legendre-Fenchel transform can be used to derive a theory of several state-of-the-art algorithms in RL, such as TRPO, SAC, A3C [87], our result is the first introducing the connection with MCTS.

3.3.3. Convergence rate to regularized objective

We show that the regularized value V_Ω can be effectively estimated at the root state $s \in \mathcal{S}$, with the assumption that each node in the tree has a σ^2-subgaussian distribution. This result extends the analysis provided in [31], which is limited to the use of maximum entropy.

Theorem 1. *At the root node s where $N(s)$ is the number of visitations, with $\epsilon > 0$, $V_\Omega(s)$ is the estimated value, with constant C and $\hat{C}$, we have*

$$\mathbf{Pr}(|V_\Omega(s) - V^*_\Omega(s)| > \epsilon) \leq C \exp\{\frac{-N(s)\epsilon}{\hat{C}\sigma \log^2(2 + N(s))}\}, \tag{3.9}$$

where $V_\Omega(s) = \Omega^*(Q_s)$ and $V^*_\Omega(s) = \Omega^*(Q^*_s)$.

From this theorem, we obtain that the convergence rate of choosing the best action a^* at the root node, when using the E3W strategy, is exponential.

Theorem 2. *Let a_t be the action returned by E3W at step t. For large enough t and constants $C, \hat{C}$*

$$\mathbf{Pr}(a_t \neq a^*) \leq Ct \exp\{-\frac{t}{\hat{C}\sigma(\log(t))^3}\}. \tag{3.10}$$

This result shows that, for every strongly convex regularizer, the convergence rate of choosing the best action at the root node is exponential, as already proven in the specific case of maximum entropy [31].

3.4. Entropy-regularization backup operators

From the introduction of a unified view of generic strongly convex regularizers as backup operators in MCTS, we narrow the analysis to entropy-based regularizers. For each entropy

function, Table 3.1 shows the Legendre-Fenchel transform and the maximizing argument, which can be respectively replaced in our backup operation (Equation 3.7) and sampling strategy E3W (Equation 3.8). Using maximum entropy retrieves the maximum entropy MCTS problem introduced in the MENTS algorithm [31]. This approach closely resembles the maximum entropy RL framework used to encourage exploration [73], [74]. We introduce two novel MCTS algorithms based on the minimization of relative entropy of the policy update, inspired by trust-region [72], [88] and proximal optimization methods [81] in RL, and on the maximization of Tsallis entropy, which has been more recently introduced in RL as an effective solution to enforce the learning of sparse policies [82]. We call these algorithms RENTS and TENTS. Contrary to maximum and relative entropy, the definition of the Legendre-Fenchel and maximizing argument of Tsallis entropy is non-trivial, being

$$\Omega^*(Q_t) = \tau \cdot \text{spmax}(Q_t(s,\cdot)/\tau), \tag{3.11}$$

$$\nabla\Omega^*(Q_t) = \max\{\frac{Q_t(s,a)}{\tau} - \frac{\sum_{a\in\mathcal{K}} Q_t(s,a)/\tau - 1}{|\mathcal{K}|}, 0\}, \tag{3.12}$$

where spmax is defined for any function $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as

$$\text{spmax}(f(s,\cdot)) \triangleq \sum_{a\in\mathcal{K}} \left(\frac{f(s,a)^2}{2} - \frac{(\sum_{a\in\mathcal{K}} f(s,a) - 1)^2}{2|\mathcal{K}|^2} \right) + \frac{1}{2}, \tag{3.13}$$

and $\mathcal{K}$ is the set of actions that satisfy $1 + if(s,a_i) > \sum_{j=1}^{i} f(s,a_j)$, with a_i indicating the action with the i-th largest value of $f(s,a)$ [82]. We point out that the Tsallis entropy is not significantly more difficult to implement. Although introducing additional computation, requiring $O(|\mathcal{A}| \log(|\mathcal{A}|))$ time in the worst case, the order of Q-values does not change between rollouts, reducing the computational complexity in practice.

3.4.1. Regret analysis

At the root node, let each children node i be assigned with a random variable X_i, with mean value V_i, while the quantities related to the optimal branch are denoted by $*$, e.g. mean value V^*. At each timestep n, the mean value of variable X_i is V_{i_n}. The pseudo-regret [89] at the root node, at timestep n, is defined as $R_n^{\text{UCT}} = nV^* - \sum_{t=1}^{n} V_{i_t}$. Similarly,

Table 3.1.: List of entropy regularizers with Legendre-Fenchel transforms and maximizing arguments (Max arg. : Max argument).

Entropy	**Regularizer** $\Omega(\pi_s)$	**Legendre-Fenchel** $\Omega^*(Q_s)$	**Max arg.** $\nabla\Omega^*(Q_s)$
Maximum	$\sum_a \pi(a\|s)\log\pi(a\|s)$	$\tau\log\sum_a e^{\frac{Q(s,a)}{\tau}}$	$\frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_b e^{\frac{Q(s,b)}{\tau}}}$
Relative	$\mathrm{D_{KL}}(\pi_t(a\|s)\|\|\pi_{t-1}(a\|s))$	$\tau\log\sum_a \pi_{t-1}(a\|s)e^{\frac{Q_t(s,a)}{\tau}}$	$\frac{\pi_{t-1}(a\|s)e^{\frac{Q_t(s,a)}{\tau}}}{\sum_b \pi_{t-1}(b\|s)e^{\frac{Q_t(s,b)}{\tau}}}$
Tsallis	$\frac{1}{2}(\parallel \pi(a\|s) \parallel_2^2 -1)$	Equation (3.11)	Equation (3.12)

we define the regret of E3W at the root node of the tree as

$$R_n = nV^* - \sum_{t=1}^{n} V_{i_t} = nV^* - \sum_{t=1}^{n} \mathbb{I}(i_t = i)V_{i_t} \tag{3.14}$$
$$= nV^* - \sum_i V_i \sum_{t=1}^{n} \hat{\pi}_t(a_i|s),$$

where $\hat{\pi}_t(\cdot)$ is the policy at time step t, and $\mathbb{I}(\cdot)$ is the indicator function.
The expected regret is defined as

$$\mathbb{E}[R_n] = nV^* - \sum_{t=1}^{n} \langle \hat{\pi}_t(\cdot), V(\cdot)\rangle . \tag{3.15}$$

Theorem 3. *Consider an E3W policy applied to the tree. Let define $\mathcal{D}_{\Omega^*}(x,y) = \Omega^*(x) - \Omega^*(y) - \nabla\Omega^*(y)(x-y)$ as the Bregman divergence between x and y, The expected pseudo regret R_n satisfies*

$$\mathbb{E}[R_n] \leq -\tau\Omega(\hat{\pi}) + \sum_{t=1}^{n} \mathcal{D}_{\Omega^*}(\hat{V}_t(\cdot) + V(\cdot), \hat{V}_t(\cdot)) \tag{3.16}$$
$$+ \mathcal{O}(\frac{n}{\log n}).$$

This theorem bounds the regret of E3W for a generic convex regularizer Ω; the regret bounds for each entropy regularizer can be easily derived from it. Let $m = \min_a \nabla\Omega^*(a|s)$.

Corollary 1. *Maximum entropy regret*
$\mathbb{E}[R_n] \leq \tau(\log|\mathcal{A}|) + \frac{n|\mathcal{A}|}{\tau} + \mathcal{O}(\frac{n}{\log n})$.

Corollary 2. *Relative entropy regret*
$\mathbb{E}[R_n] \leq \tau(\log|\mathcal{A}| - \frac{1}{m}) + \frac{n|\mathcal{A}|}{\tau} + \mathcal{O}(\frac{n}{\log n})$.

Corollary 3. *Tsallis entropy regret*
$\mathbb{E}[R_n] \leq \tau(\frac{|\mathcal{A}|-1}{|\mathcal{A}|}) + \frac{n|\mathcal{K}|}{2} + \mathcal{O}(\frac{n}{\log n})$.

Remarks. The regret bound of UCT and its variance have already been analyzed for non-regularized MCTS with binary tree [89]. On the contrary, our regret bound analysis in Theorem 3 applies to generic regularized MCTS. From the specialized bounds in the corollaries, we observe that the maximum and relative entropy share similar results, although the bounds for relative entropy are slightly smaller due to $\frac{1}{m}$. Remarkably, the bounds for Tsallis entropy become tighter for increasing number of actions, which translates in limited regret in problems with high branching factor. This result establishes the advantage of Tsallis entropy in complex problems w.r.t. to other entropy regularizers, as empirically confirmed in Section 3.5.

3.4.2. Error analysis

We analyse the error of the regularized value estimate at the root node $n(s)$ w.r.t. the optimal value: $\varepsilon_\Omega = V_\Omega(s) - V^*(s)$.

Theorem 4. *For any $\delta > 0$ and generic convex regularizer Ω, with some constant $C, \hat{C}$, with probability at least $1-\delta$, ε_Ω satisfies*

$$-\sqrt{\frac{\hat{C}\sigma^2 \log\frac{C}{\delta}}{2N(s)}} - \frac{\tau(U_\Omega - L_\Omega)}{1-\gamma} \leq \varepsilon_\Omega \leq \sqrt{\frac{\hat{C}\sigma^2 \log\frac{C}{\delta}}{2N(s)}}. \tag{3.17}$$

To the best of our knowledge, this theorem provides the first result on the error analysis of value estimation at the root node of convex regularization in MCTS. To give a better understanding of the effect of each entropy regularizer in Table 3.1, we specialize the bound in Equation 3.17 to each of them. From [82], we know that for maximum entropy $\Omega(\pi_t) = \sum_a \pi_t \log \pi_t$, we have $-\log|\mathcal{A}| \leq \Omega(\pi_t) \leq 0$; for relative entropy $\Omega(\pi_t) = \mathrm{KL}(\pi_t||\pi_{t-1})$, if we define $m = \min_a \pi_{t-1}(a|s)$, then we can derive $0 \leq \Omega(\pi_t) \leq -\log|\mathcal{A}| +$

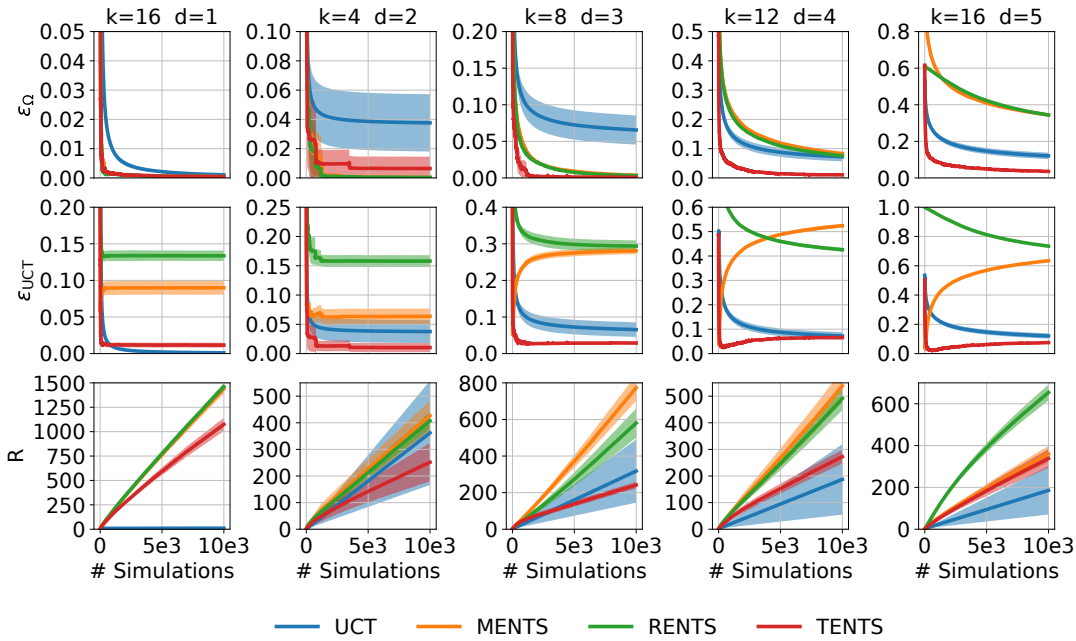


Figure 3.1.: For each algorithm, we show the convergence of the value estimate at the root node to the respective optimal value (top), to the UCT optimal value (middle), and the regret (bottom).

$\log \frac{1}{m}$; and for Tsallis entropy $\Omega(\pi_t) = \frac{1}{2}(\| \pi_t \|_2^2 - 1)$, we have $-\frac{|\mathcal{A}|-1}{2|\mathcal{A}|} \leq \Omega(\pi_t) \leq 0$. Then, defining $\Psi = \sqrt{\frac{\hat{C}\sigma^2 \log \frac{C}{\delta}}{2N(s)}}$,

Corollary 4. *Maximum entropy error*
$-\Psi - \dfrac{\tau \log |\mathcal{A}|}{1-\gamma} \leq \varepsilon_\Omega \leq \Psi.$

Corollary 5. *Relative entropy error*
$-\Psi - \dfrac{\tau(\log |\mathcal{A}| - \log \frac{1}{m})}{1-\gamma} \leq \varepsilon_\Omega \leq \Psi.$

Corollary 6. *Tsallis entropy error*
$-\Psi - \dfrac{|\mathcal{A}|-1}{2|\mathcal{A}|} \dfrac{\tau}{1-\gamma} \leq \varepsilon_\Omega \leq \Psi.$

These results show that when the number of actions $|\mathcal{A}|$ is large, TENTS enjoys the smallest error; moreover, we also see that lower bound of RENTS is always smaller than for MENTS.

3.5. Empirical evaluation

In this section, we empirically evaluate the benefit of the proposed entropy-based MCTS regularizers. First, we complement our theoretical analysis with an empirical study of the synthetic tree toy problem introduced in [31], which serves as a simple scenario to give an interpretable demonstration of the effects of our theoretical results in practice. Second, we compare to AlphaGo [28], recently introduced to enable MCTS to solve large scale problems with high branching factor. Our implementation is a simplified version of the original algorithm, where we remove various tricks in favor of better interpretability. For the same reason, we do not compare with the most recent and state-of-the-art MuZero [90], as this is a slightly different solution highly tuned to maximize performance, and a detailed description of its implementation is not available.
The learning time of AlphaZero can be slow in problems with high branching factor, due to the need of a large number of MCTS simulations for obtaining good estimates of the randomly initialized action-values. To overcome this problem, AlphaGo [28] initializes the action-values using the values retrieved from a pretrained network, which is kept fixed during the training.

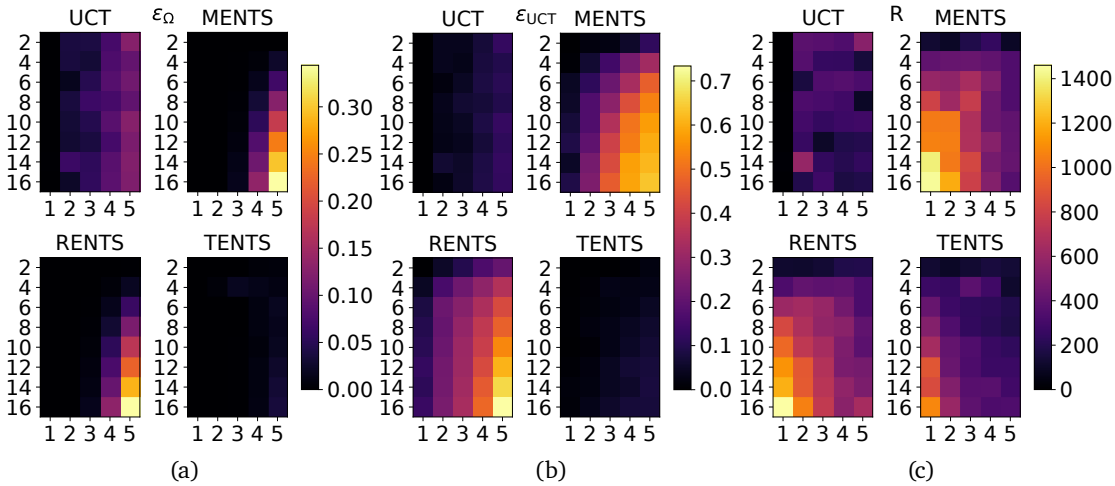
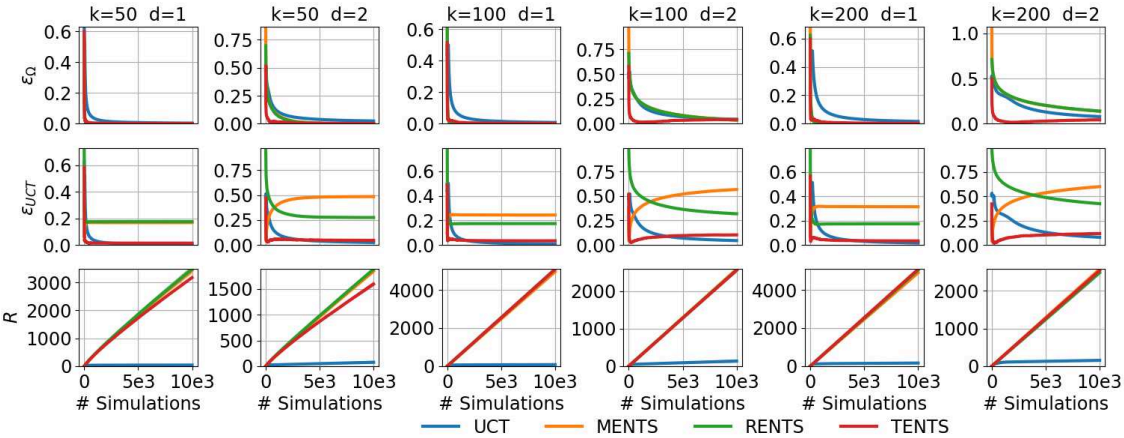


Figure 3.2.: For different branching factor k (rows) and depth d (columns), the heatmaps show: the absolute error of the value estimate at the root node after the last simulation of each algorithm w.r.t. the respective optimal value (a), and w.r.t. the optimal value of UCT (b); regret at the root node (c).

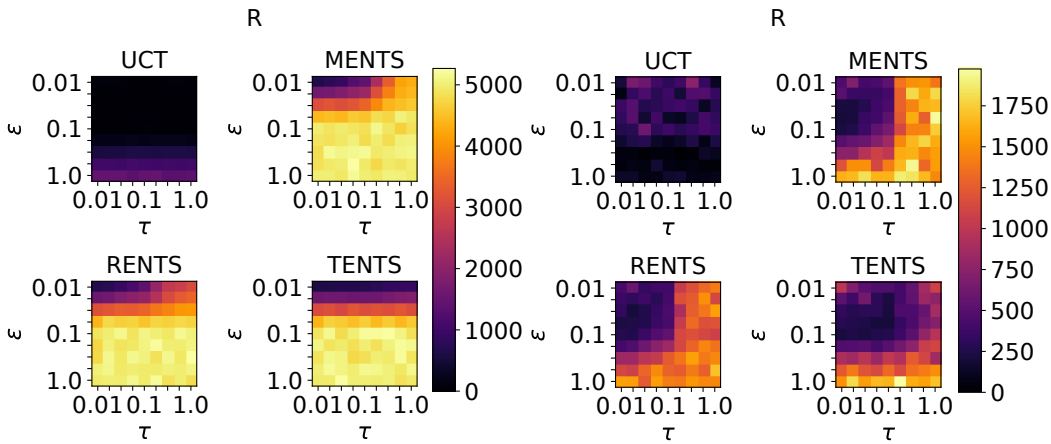
3.5.1. Synthetic tree

This toy problem is introduced in [31] to highlight the improvement of MENTS over UCT. It consists of a tree with branching factor k and depth d. Each edge of the tree is assigned a random value between 0 and 1. At each leaf, a Gaussian distribution is used as an evaluation function resembling the return of random rollouts. The mean of the Gaussian distribution is the sum of the values assigned to the edges connecting the root node to the leaf, while the standard deviation is $\sigma = 0.05$[2]. For stability, all the means are normalized between 0 and 1. As in [31], we create 5 trees on which we perform 5 different runs in each, resulting in 25 experiments, for all the combinations of branching factor $k = \{2, 4, 6, 8, 10, 12, 14, 16\}$ and depth $d = \{1, 2, 3, 4, 5\}$, computing: (i) the value estimation error at the root node w.r.t. the regularized optimal value: $\varepsilon_\Omega = V_\Omega - V^*_\Omega$; (ii) the value estimation error at the root node w.r.t. the unregularized optimal value $\varepsilon_{\text{UCT}} = V_\Omega - V^*_{\text{UCT}}$; (iii) the regret R as in Equation (3.14). For a fair comparison, we use fixed $\tau = 0.1$ and $\epsilon = 0.1$ across all algorithms. Figure 3.1 and 3.2 show how UCT and each regularizer behave for different configurations of the tree. We observe that, while

[2]The value of the standard deviation is not provided in [31]. After trying different values, we observed that our results match the one in [31] when using $\sigma = 0.05$.



(a) Results in trees with high branching factor.



(b) $k = 100$, $d = 1$.

(c) $k = 8$, $d = 3$.

Figure 3.3.: High branching factor trees (a), regret sensitivity study w.r.t. ε and τ (b, c).

RENTS and MENTS converge slower for increasing tree sizes, TENTS is robust w.r.t. the size of the tree and almost always converges faster than all other methods to the respective optimal value. Notably, the optimal value of TENTS seems to be very close to the one of UCT, i.e. the optimal value of the unregularized objective, and also converges faster than the one estimated by UCT, while MENTS and RENTS are considerably further from this value. In terms of regret, UCT explores less than the regularized methods and it is less prone to high regret, at the cost of slower convergence time. Nevertheless, the regret of TENTS is the smallest between the ones of the other regularizers, which seem to explore

too much. In Figure (a), we show further results evincing the advantages of TENTS over the baselines in problems with high branching factor, in terms of approximation error and regret. Finally, in Figures (b) and (c) we carry out a sensitivity analysis of each algorithm w.r.t. the values of the exploration coefficient ε and τ in two different trees. Note that ε is only used by E3W to choose whether to sample uniformly or from the regularized policy. We observe that the choice of τ does not significantly impact the regret of TENTS, as opposed to the other methods. These results show a general superiority of TENTS in this toy problem, also confirming our theoretical findings about the advantage of TENTS in terms of approximation error (Corollary 6) and regret (Corollary 3), in problems with many actions.

3.5.2. Entropy-regularized AlphaGo

Atari. Atari 2600 [84] is a popular benchmark for testing deep RL methodologies [30], [91], [92] but still relatively disregarded in MCTS. We use a Deep Q-Network, pretrained using the same experimental setting of [30], to initialize the action-value function of each node after expansion as $Q_{init}(s,a) = (Q(s,a) - V(s))/\tau$, for MENTS and TENTS, as done in [31]. For RENTS we init $Q_{init}(s,a) = \log P_{\text{prior}}(a|s)) + (Q(s,a) - V(s))/\tau$, where P_{prior} is the Boltzmann distribution induced by action-values $Q(s,.)$ computed from the network. Each experimental run consists of 512 MCTS simulations. The temperature τ is optimized for each algorithm and game via grid-search between 0.01 and 1. The discount factor is $\gamma = 0.99$, and for PUCT the exploration constant is $c = 0.1$. Table 3.2 shows the performance, in terms of cumulative reward, of standard AlphaGo with PUCT and our three regularized versions, on 22 Atari games. Moreover, we test also AlphaGo using the MaxMCTS backup [63] for further comparison with classic baselines. We observe that regularized MCTS dominates other baselines, in particular TENTS achieves the highest scores in all the 22 games, showing that sparse policies are more effective in Atari. In particular, TENTS significantly outperforms the other methods in the games with many actions, e.g. Asteroids, Phoenix, confirming the results obtained in the synthetic tree experiment, explained by corollaries 3 and 6 on the benefit of TENTS in problems with high-branching factor.

3.6. Related Work

Entropy regularization is a common tool for controlling exploration in Reinforcement Learning (RL) and has lead to several successful methods [72]–[74], [93]. Typically spe-

cific forms of entropy are utilized such as maximum entropy [74] or relative entropy [72]. This approach is an instance of the more generic duality framework, commonly used in convex optimization theory. Duality has been extensively studied in game theory [79], [80] and more recently in RL, for instance considering mirror descent optimization [94], [95], drawing the connection between MCTS and regularized policy optimization [96], or formalizing the RL objective via Legendre-Rockafellar duality [78]. Recently [77] introduced regularized Markov Decision Processes, formalizing the RL objective with a generalized form of convex regularization, based on the Legendre-Fenchel transform. In this chapter, we provide a novel study of convex regularization in MCTS, and derive relative entropy (KL-divergence) and Tsallis entropy regularized MCTS algorithms, i.e. RENTS and TENTS respectively. Note that the recent maximum entropy MCTS algorithm MENTS [31] is a special case of our generalized regularized MCTS. Unlike MENTS, RENTS can take advantage of any action distribution prior, in the experiments the prior is derived using Deep Q-learning [30]. On the other hand, TENTS allows for sparse action exploration and thus higher dimensional action spaces compared to MENTS. Several works focus on modifying classical MCTS to improve exploration. UCB1-tuned [45] modifies the upper confidence bound of UCB1 to account for variance in order to improve exploration. [51] proposes a Bayesian version of UCT, which obtains better estimates of node values and uncertainties given limited experience. Many heuristic approaches based on specific domain knowledge have been proposed, such as adding a bonus term to value estimates [53]–[57] or prior knowledge collected during policy search [58]–[62]. [63] formalizes and analyzes different on-policy and off-policy complex backup approaches for MCTS planning based on RL techniques. [64] proposes an approach called SARSA-UCT, which performs the dynamic programming backups using SARSA [65]. Both [63] and [64] directly borrow value backup ideas from RL to estimate the value at each tree node, but they do not provide any proof of convergence.

3.7. Conclusion

We introduced a theory of convex regularization in Monte-Carlo Tree Search (MCTS) based on the Legendre-Fenchel transform. We proved that a generic strongly convex regularizer has an exponential convergence rate for the selection of the optimal action at the root node. Our result gives theoretical motivations to previous results specific to maximum entropy regularization. Furthermore, we provided the first study of the regret of MCTS when using a generic strongly convex regularizer, and an analysis of the error between the regularized value estimate at the root node and the optimal regularized value. We use these results to motivate the use of entropy regularization in MCTS, considering

maximum, relative, and Tsallis entropy, and we specialized our regret and approximation error bounds to each entropy-regularizer. We tested our regularized MCTS algorithm in a simple toy problem, where we give an empirical evidence of the effect of our theoretical bounds for the regret and approximation error. Finally, we introduced the use of convex regularization in AlphaGo, and carried out experiments on several Atari games. Overall, our empirical results show the advantages of convex regularization, and in particular the superiority of Tsallis entropy w.r.t. other entropy-regularizers.
Future developments of this work can investigate the possibility of mixing UCT and the regularized policy in a complementary manner. In our empirical results, we observed that UCT enjoys a better cumulative regret than regularized policies, while regularized policies have a better one-step regret due to the exponential convergence rate. Considering that cumulative regret and one-step regret can be both significant objectives to minimize according to the problem at hand [97], studying theoretically sound ways of mixing UCT and regularized policy seems a promising idea.

Table 3.2.: Average score in Atari over 100 seeds per game. Bold denotes no statistically significant difference to the highest mean (t-test, $p < 0.05$). Bottom row shows # no difference to highest mean.

	UCT	MaxMCTS	MENTS	RENTS	TENTS
Alien	**1,486.80**	**1,461.10**	**1,508.60**	**1,547.80**	**1,568.60**
Amidar	115.62	**124.92**	**123.30**	**125.58**	**121.84**
Asterix	4,855.00	**5,484.50**	**5,576.00**	**5,743.50**	**5,647.00**
Asteroids	873.40	899.60	1,414.70	1,486.40	**1,642.10**
Atlantis	35,182.00	**35,720.00**	**36,277.00**	35,314.00	**35,756.00**
BankHeist	475.50	458.60	**622.30**	**636.70**	**631.40**
BeamRider	**2,616.72**	**2,661.30**	**2,822.18**	2,558.94	**2,804.88**
Breakout	**303.04**	296.14	**309.03**	300.35	**316.68**
Centipede	1,782.18	1,728.69	**2,012.86**	**2,253.42**	**2,258.89**
DemonAttack	579.90	640.80	**1,044.50**	**1,124.70**	**1,113.30**
Enduro	**129.28**	124.20	128.79	**134.88**	**132.05**
Frostbite	1,244.00	1,332.10	**2,388.20**	**2,369.80**	**2,260.60**
Gopher	3,348.40	3,303.00	**3,536.40**	**3,372.80**	**3,447.80**
Hero	3,009.95	3,010.55	**3,044.55**	**3,077.20**	**3,074.00**
MsPacman	1,940.20	1,907.10	2,018.30	**2,190.30**	**2,094.40**
Phoenix	2,747.30	2,626.60	3,098.30	2,582.30	**3,975.30**
Qbert	7,987.25	8,033.50	8,051.25	8,254.00	**8,437.75**
Robotank	**11.43**	11.00	**11.59**	**11.51**	**11.47**
Seaquest	**3,276.40**	**3,217.20**	**3,312.40**	**3,345.20**	**3,324.40**
Solaris	895.00	923.20	**1,118.20**	**1,115.00**	**1,127.60**
SpaceInvaders	778.45	**835.90**	**832.55**	**867.35**	**822.95**
WizardOf Wor	685.00	666.00	**1,211.00**	**1,241.00**	**1,231.00**
# Highest mean	6/22	7/22	17/22	16/22	**22/22**

4. A Unified Perspective on Value Backup and Exploration in Monte-Carlo Tree Search

> "I don't believe we're seeing the beginning of a divergence. We have seen a partial divergence on this case."
>
> — *Mario Monti*

Monte-Carlo Tree Search (MCTS) is a class of methods for solving complex decision-making problems through the synergy of Monte-Carlo planning and Reinforcement Learning (RL). The highly combinatorial nature of the problems commonly addressed by MCTS requires the use of efficient exploration strategies for navigating the planning tree and quickly convergent value backup methods. These crucial problems are particularly evident in recent advances that combine MCTS with deep neural networks for function approximation. In this work, we introduce a mathematical framework based on using the α-divergence for backup and exploration in MCTS. We show that this theoretical formulation unifies different approaches, including our newly introduced ones (Power-UCT and E3W), under the same mathematical framework, allowing us to obtain different methods by simply changing the value of α. In practice, our unified perspective offers a flexible way to balance exploration and exploitation by tuning the single α parameter according to the problem at hand. We validate our methods through a rigorous empirical study of a basic toy task Synthetic Tree problem.

4.1. Introduction

Monte-Carlo Tree Search (MCTS) is an effective method that combines a random sampling strategy with tree search to determine the optimal decision for on-the-fly planning tasks.

MCTS has yielded impressive results in Go [28] (AlphaGo), Chess [14] (AlphaZero), or video games [33], and it has been further exploited successfully in motion planning [98], [99], autonomous car driving [100], [101], and autonomous robotic assembly tasks [102]. Many of the MCTS successes [14], [28], [71] rely on coupling MCTS with neural networks trained using Reinforcement Learning (RL) [29] methods such as Deep Q-Learning [30], to speed up learning of large scale problems.

Despite AlphaGo and AlphaZero achieving state-of-the-art performance in games with high branching factors like Go [28] and Chess [14], both methods suffer from poor sample efficiency, mostly due to the inefficiency of the average mean backup operator, which is well-known for the issue of underestimating the optimum and leading to the polynomial convergence rate of PUCT [31]. This problem, combined with the need for effective exploration techniques, particularly in highly stochastic environments like Atari [103] poses an open research problem for the MCTS community: effective exploration methods and sufficient backup operators for the planning tree.

In this work, we provide a theory of the use of α-divergence in MCTS, respectively showing how the different range of α parameter solves the exploration-exploitation trade-off schema and prove that a class of our novel backup operators ensure the exponential convergence rate, showing the advantages over the polynomial convergence rate of UCT [12]. We further draw the connection between the two recent advanced MCTS methods, Power-UCT [43] and E3W [44], which have been proven to provide effective solutions for the exploration and backup operator problems in the tree, by providing a rigorous theoretical study of α-divergence in MCTS and analyze how α-divergence can help to derive power mean and entropic regularization in MCTS.

α-divergence has been first extensively studied in RL context by [88], and later on, has been proposed to use in [104] as a generalized Tsallis Entropy regularizer in MDP. However, the study of α-divergence in MCTS is still an open question. In this work, we first show that power mean (the new backup operator used in Power-UCT) can be derived as a closed-form solution of a mean of distribution by considering α-divergence as the probability distance, generalizing the eclipse distance that is used to derive average mean of a distribution. We further exploit the convex regularization framework in MCTS by analyzing the α-divergence function as the regularizer to introduce novel regularized backup operators for MCTS, relatively derive the maximum entropy, the relative entropy of the policy update, and, more importantly, derive the Tsallis entropy of the policy those has been proposed in E3W. Finally, we measure α-divergence in Synthetic Tree and show how α-divergence help to achieve competitive results in challenging problems.

4.2. Related Work

We want to improve the efficiency and performance of MCTS by addressing the two crucial problems of value backup and exploration. Our contribution follows on from a plethora of previous works that we briefly summarize in the following.

Backup operators. To improve upon the UCT algorithm in MCTS, [63] formalize and analyze different on-policy and off-policy complex backup approaches for MCTS planning based on techniques in the RL literature. [63] propose four complex backup strategies: MCTS(λ), MaxMCTS(λ), MCTS$_\gamma$, MaxMCTS$_\gamma$, and report that MaxMCTS(λ) and MaxMCTS$_\gamma$ perform better than UCT for certain parameter setups. [64] propose an approach called SARSA-UCT, which performs the dynamic programming backups using SARSA [65]. Both [63] and [64] directly borrow value backup ideas from RL in order to estimate the value at each tree node. However, they do not provide any proof of convergence. The recently introduced MENTS algorithm [31], uses softmax backup operator at each node in combination with an entropy-based exploration policy, and shows a better convergence rate w.r.t. UCT.

Exploration. Entropy regularization is a common tool for controlling exploration in RL and has led to several successful methods [72]–[74], [93]. Typically specific forms of entropy are utilized such as maximum entropy [74] or relative entropy [72]. This approach is an instance of the more generic duality framework, commonly used in convex optimization theory. Duality has been extensively studied in game theory [79], [80] and more recently in RL, for instance considering mirror descent optimization [94], [95], drawing the connection between MCTS and regularized policy optimization [96], or formalizing the RL objective via Legendre-Rockafellar duality [78]. Recently [77] introduced regularized Markov Decision Processes, formalizing the RL objective with a generalized form of convex regularization, based on the Legendre-Fenchel transform. Several works focus on modifying classical MCTS to improve exploration. For instance, [51] propose a Bayesian version of UCT to improve estimation of node values and uncertainties given limited experience.

α-divergence. α-divergence has been extensively studied in RL context by [88], that propose to use it as the divergence measurement policy search, generalizing the relative entropy policy search to constrain the policy update. [88] further study a particular class of f-divergence, called α-divergence, resulting in compatible policy update and value function improvement in the actor-critic methods. [104] on the other hand, analyze α-divergence as a generalized Tsallis Entropy regularizer in MDP. Controlling the generalized Tsallis Entropy regularizer by scaling the α parameter as an entropic index, [104] derive

the Shannon-Gibbs entropy and Tsallis Entropy as special cases.

4.3. Preliminaries

In this section, we will present background knowledge on Markov Decision Processes, Monte-Carlo Tree Search and α-divergence.

4.3.1. Markov Decision Processes

Please refer to Section 3.2.1

4.3.2. Monte-Carlo Tree Search

Please refer to Section 2.3.1

4.3.3. α-divergence

The f-divergence [105] generalizes the definition of the distance between two probabilistic distributions P and Q on a finite set $\mathcal{A}$ as

$$D_f\left(P\|Q\right) = \sum_{a\in\mathcal{A}} Q(a) f\left(\frac{P(a)}{Q(a)}\right), \tag{4.1}$$

where f is a convex function on $(0,\infty)$ such as $f(1)=0$. For example, the KL-divergence corresponds to $f_{KL} = x\log x - (x-1)$. The $\alpha-$divergence is a subclass of f-divergence generated by $\alpha-$function with $\alpha\in\mathbb{R}$. $\alpha-$function is defined as

$$f_\alpha(x) = \frac{(x^\alpha - 1) - \alpha(x-1)}{\alpha(\alpha-1)}. \tag{4.2}$$

The $\alpha-$divergence between two probabilistic distributions P and Q on a finite set $\mathcal{A}$ is defined as

$$D_\alpha\left(P\|Q\right) = \sum_{a\in\mathcal{A}} Q(a) f_\alpha\left(\frac{P(a)}{Q(a)}\right), \tag{4.3}$$

where $\sum_{a\in\mathcal{A}} Q(a) = \sum_{a\in\mathcal{A}} P(a) = 1$.
Furthermore, given the $\alpha-$function, we can derive the generalization of Tsallis entropy of a policy π as

$$\mathrm{H}_{\alpha}^{\pi}(s) = \frac{1}{\alpha(1-\alpha)}\left(1 - \sum_{a\in\mathcal{A}} \pi(s,a)^{\alpha}\right) \tag{4.4}$$

In addition, we have

$$\lim_{\alpha\to 1} \mathrm{H}_{1}^{\pi}(s) = -\sum_{a\in\mathcal{A}} \pi(s,a)\log\pi(s,a) \tag{4.5}$$

$$\mathrm{H}_{2}^{\pi}(s) = \frac{1}{2}\left(1 - \sum_{a\in\mathcal{A}} \pi(s,a)^{2}\right), \tag{4.6}$$

respectively, the Shannon entropy (4.5) and the Tsallis entropy (4.6) functions.

4.4. α-divergence in MCTS

In this section, we show how to use α-divergence as a convex regularization function to generalize the entropy regularization in MCTS and respectively derive MENTS, RENTS, and TENTS. Additionally, we show how to derive power mean (which is used as the backup operator in Power-UCT) using α-divergence as the distance function to replace the Euclidean distance in the definition of the empirical average mean value. Finally, we study the regret bound and error analysis of the α-divergence regularization in MCTS.

4.4.1. α-divergence Regularization in MCTS

We introduce α-divergence regularization to MCTS. Denote the Legendre-Fenchel transform (or convex conjugate) of α-divergence regularization with $\Omega^* : \mathbb{R}^{\mathcal{A}} \to \mathbb{R}$, defined as

$$\Omega^*(Q_s) \triangleq \max_{\pi_s\in\Pi_s} \mathcal{T}_{\pi_s} Q_s - \tau f_{\alpha}(\pi_s), \tag{4.7}$$

where the temperature τ specifies the strength of regularization, and f_{α} is the α function defined in (4.2). Note that α-divergence of the current policy π_s and the uniform policy has the same form as the α function $f_s(\pi_s)$.

It is known that:

- when $\alpha = 1$, we have the regularizer $f_1(\pi_s) = \pi_s \log \pi_s = -H(\pi_s)$, and derive Shannon entropy, getting MENTS. Note that if we apply the α-divergence with $\alpha = 1$, we get RENTS;
- when $\alpha = 2$, we have the regularizer $f_2(\pi_s) = \frac{1}{2}(\pi_s - 1)^2$, and derive Tsallis entropy, getting TENTS.

For $\alpha > 1, \alpha \neq 2$ we can derive [106]

$$\nabla\Omega^*(Q_t) = \left(\max\left\{\frac{Q^{\pi^*_\tau(s,a)}}{\tau} - \frac{c(s)}{\tau}, 0\right\}(\alpha - 1)\right)^{\frac{1}{\alpha-1}} \tag{4.8}$$

where

$$c(s) = \tau\frac{\sum_{a\in\mathcal{K}(s)} \frac{Q^{\pi^*_\tau(s,a)}}{\tau} - 1}{\|\mathcal{K}(s)\|} + \tau\left(1 - \frac{1}{\alpha - 1}\right), \tag{4.9}$$

with $\mathcal{K}(s)$ representing the set of actions with non-zero chance of exploration in state s, as determined below

$$\mathcal{K}(s) = \left\{a_i \middle| 1 + i\frac{Q^{\pi^*_\tau(s,a_i)}}{\tau} > \sum_{j=1}^{i} \frac{Q^{\pi^*_\tau(s,a_j)}}{\tau} + i(1 - \frac{1}{\alpha - 1})\right\}, \tag{4.10}$$

where a_i denotes the action with the $i-$th highest Q-value in state s. and the regularized value function

$$\Omega^*(Q_t) = \left\langle \nabla\Omega^*(Q_t), Q^{\pi^*_\tau(s,a)} \right\rangle. \tag{4.11}$$

4.4.2. Connecting Power Mean with α-divergence

In order to connect the Power-UCT approach that we introduced in Chapter 2 with α-divergence, we study here the entropic mean [107] which uses f-divergence, of which α-divergence is a special case, as the distance measure. Since power mean is a special case of the entropic mean, the entropic mean allows us to connect the geometric properties of the power mean used in Power-UCT with α-divergence.

In more detail, let $a = (a_1, a_2, ...a_n)$ be given strictly positive numbers and let $w = (w_1, w_2, ..., w_n)$ be given weights and $\sum_{i=1}^{n} w_i = 1, w_i > 0, i = 1...n$. Let's define $dist(\alpha, \beta)$ as the distance measure between $\alpha, \beta > 0$ that satisfies

$$dist(\alpha, \beta) = \begin{cases} 0 \text{ if } \alpha = \beta \\ > 0 \text{ if } \alpha \neq \beta \end{cases} \tag{4.12}$$

When we consider the distance as f-divergence between the two distributions, we get the entropic mean of $a = (a_1, a_2, ...a_n)$ with weights $w = (w_1, w_2, ..., w_n)$ as

$$\text{mean}_w(a) = \min_{x>0} \left\{ \sum_{i=1}^{n} w_i a_i f\left(\frac{x}{a_i}\right) \right\}. \tag{4.13}$$

When applying $f_\alpha(x) = \frac{x^{1-p}-p}{p(p-1)} + \frac{x}{p}$, with $p = 1 - \alpha$, we get

$$\text{mean}_w(a) = \left(\sum_{i=1}^{n} w_i a_i^p \right)^{\frac{1}{p}}, \tag{4.14}$$

which is equal to the power mean.

4.4.3. Regret and Error Analysis of α-divergence in Monte-Carlo Tree Search

We measure how different values of α in the α-divergence function affect the regret in MCTS.

Theorem 12. *When $\alpha \in (0, 1)$, the regret of E3W is*

$$\mathbb{E}[R_n] \leq \frac{\tau}{\alpha(1-\alpha)}(|\mathcal{A}|^{1-\alpha} - 1) + n(2\tau)^{-1}|\mathcal{A}|^{\alpha} + \mathcal{O}(\frac{n}{\log n}).$$

For $\alpha \in (1, \infty)$, we derive the following results

Theorem 13. *When $\alpha \in (1, \infty)$, the regret of E3W is*

$$\mathbb{E}[R_n] \leq \frac{\tau}{\alpha(1-\alpha)}(|\mathcal{A}|^{1-\alpha} - 1) + \frac{n|\mathcal{K}|}{2} + \mathcal{O}(\frac{n}{\log n}).$$

where $|\mathcal{K}|$ is the number of actions that are assigned non-zero probability in the policy at the root node. Note that when $\alpha = 1, 2$, please refer to Corollary 1, 2, 3 in Chapter 3.

We analyse the error of the regularized value estimate at the root node $n(s)$ w.r.t. the optimal value $\varepsilon_\Omega = V_\Omega(s) - V^*(s)$. where Ω is the α-divergence regularizer f_α.

Theorem 14. *For any $\delta > 0$ and α-divergence regularizer f_α ($\alpha \neq 1, 2$), with some constant $C, \hat{C}$, with probability at least $1 - \delta$, ε_Ω satisfies*

$$-\sqrt{\frac{\hat{C}\sigma^2 \log \frac{C}{\delta}}{2N(s)}} - \frac{\tau}{\alpha(1-\alpha)}(|\mathcal{A}|^{1-\alpha} - 1) \leq \varepsilon_\Omega \leq \sqrt{\frac{\hat{C}\sigma^2 \log \frac{C}{\delta}}{2N(s)}}. \tag{4.15}$$

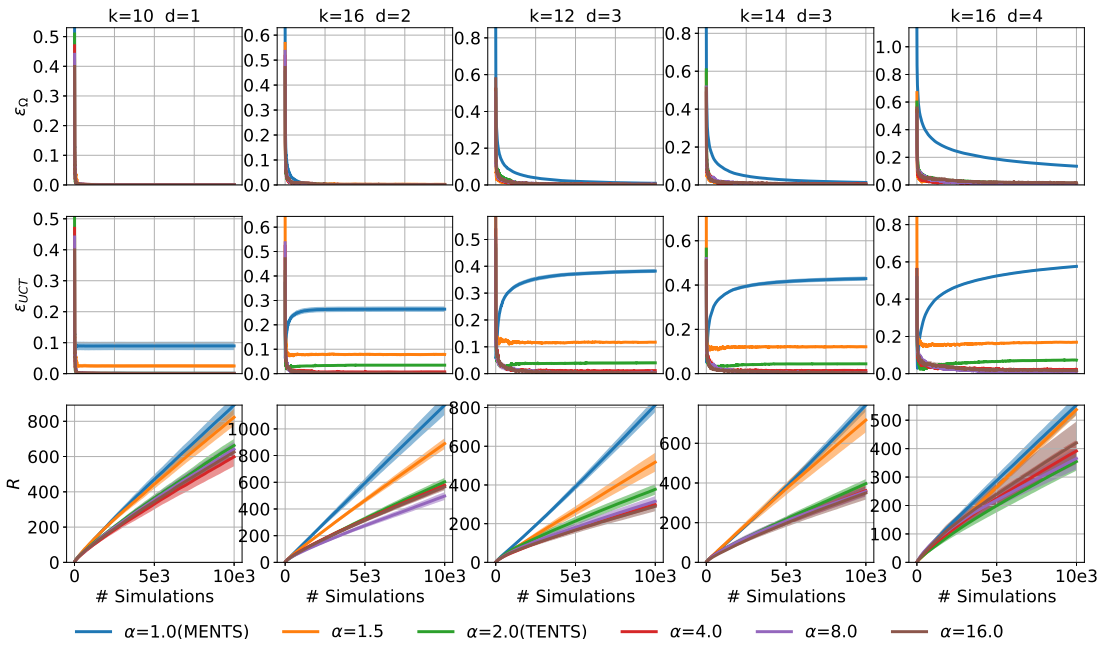


Figure 4.1.: We show the convergence of the value estimate at the root node to the respective optimal value (top), to the UCT optimal value (middle), and the regret (bottom) with different α parameter of α-divergence in Synthetic tree environment with $\alpha = 1.0$ (MENTS), 1.5, 2.0 (TENTS), 4.0, 8.0, 16.0.

For $\alpha = 1, 2$, please refer to Corollary 4, Corollary 5, Corollary 6 in Chapter 3. We can see that when α increases, the error bound decreases.

4.5. Empirical Evaluation

In this section, we plan to measure the performance of the advanced method Power-UCT and the entropy based regularizer MCTS with repect to a range of difference value of α hyperparameters.

4.5.1. Synthetic Tree

We further use the toy problem Synthetic Tree to measure how the α-divergence help to balance exploration and exploitation in MCTS. We use the same experimental settings

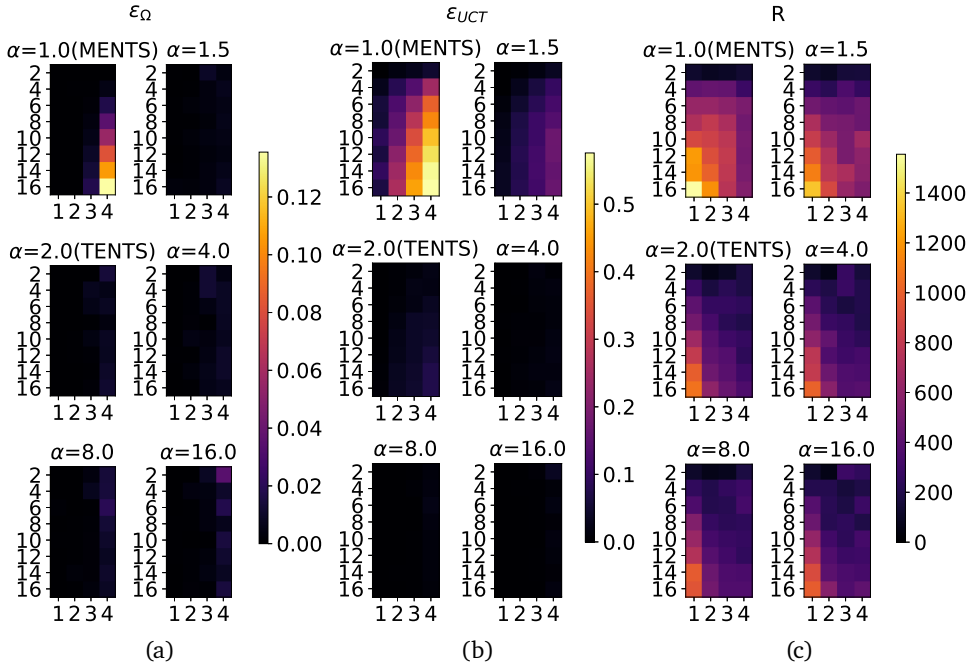


Figure 4.2.: We show the effectiveness of α-divergence in Synthetic Tree environment with different branching factor k (rows) and depth d (columns). The heatmaps show: the absolute error of the value estimate at the root node after the last simulation of each algorithm w.r.t. the respective optimal value (a), and w.r.t. the optimal value of UCT (b); regret at the root node (c).

as in the last section, with the variance of the distributions at each node of the Synthetic Tree set to $\sigma = 0.05$. The mean value of each distribution at each node of the toy problem is normalized between 0 and 1 for stabilizing. We set the temperature $\tau = 0.1$ and the exploration $\epsilon = 0.1$. Figure 4.2 illustrates the heatmap of the absolute error of the value estimate at the root node after the last simulation of each algorithm w.r.t. the respective optimal regularized value, the optimal value of UCT and regret at the root node with $\alpha = 1.0$ (Maximum entropy Monte-Carlo planning (MENTS)), 1.5, 2.0 (Tsallis entropy Monte-Carlo planning (TENTS)), 4.0, 8.0, 16.0. Figure 4.1 shows the convergence of the value estimate and regret at the root node of α-divergence in the Synthetic Tree environment. It shows that the error of the value estimate at the root node with respect to the optimal UCT value and the regularized value decrease when α increase, which

matches our theoretical results in Theorem 14. Regarding the regret, the performance is different depending on different branching factors k and depth d, which illustrates that the value of α helps trade-off between exploration and exploitation depending on each environment. For example, with $k = 16, d = 2$, the regret is smaller when we increase the value of α, and the regret is smallest with $\alpha = 8.0$. When $k = 14, d = 3$, the regret is smaller when we increase the value of α and the regret performance is the best with $\alpha = 16.0$, and when $k = 16, d = 4$, the regret enjoys the best performance with $\alpha = 2.0$ (TENTS)

4.6. Conclusion

We introduced a unified view of the use of α-divergence in Monte-Carlo Tree Search(MCTS). We show that Power-UCT and the convex regularization in MCTS can be connected using α-divergence. In detail, the Power Mean backup operator used in Power-UCT can be derived as the solution of using α function as the probabilistic distance to replace the Eclipse distance used to calculate the average mean, in which the closed-form solution is the generalized power mean. Furthermore, entropic regularization in MCTS can be derived using α-function regularization. We provided the analysis of the regret bound of Power-UCT and E3W with respect to the α parameter. We further analyzed the error bound between the regularized value estimate and the optimal regularized value at the root node. Empirical results in Synthetic Tree showed the effective balance between exploration and exploitation of α-divergence in MCTS with different values of α.

5. Monte-Carlo Robot Path Planning

> "No one saves us but ourselves. No one can and no one may. We ourselves must walk the path."
>
> — *Buddha*

Path planning is a crucial algorithmic component when optimizing robot behavior. Sampling-based approaches, like rapidly exploring random trees (RRTs) [37] or probabilistic roadmaps [108], are prominent algorithmic solutions for path planning problems. Despite its exponential convergence rate, RRT can only find suboptimal paths. On the other hand, RRT*, a widely-used extension to RRT, guarantees probabilistic completeness for finding optimal paths but suffers in practice from slow convergence in complex environments. Furthermore, real-world robotic environments are often partially observable or with poorly described dynamics, casting the application of RRT* in complex tasks suboptimal. This work studies a novel algorithmic formulation of the popular Monte-Carlo tree search (MCTS) algorithm for robot path planning. Notably, we study Monte-Carlo Path Planning (MCPP) by analyzing and proving, on the one part, its exponential convergence rate to the optimal path in fully observable Markov decision processes (MDPs), and on the other part, its probabilistic completeness for finding feasible paths in partially observable MDPs (POMDPs) assuming limited distance observability (proof sketch). Our algorithmic contribution allows us to employ recently proposed variants of MCTS [109] with different exploration strategies for robot path planning. Our experimental evaluations in simulated 2D and 3D environments with a 7 degrees of freedom (DOF) manipulator, as well as in a real-world robot path planning task, demonstrate the superiority of MCPP in POMDP tasks.

5.1. Introduction

Robot path planning refers to the process of finding a sequence of configurations that lead a robot system from a starting configuration to a goal configuration without violating

task constraints. Path planning is a crucial component in robotics [110], autonomous driving [111] and other domains such as surgical planning, computational biology, and molecules [112]. In robotics, path planning is an integral tool for manipulation tasks with robotic manipulator arms [113]–[115] and mobile robots [7], [116], [117].

Due to the redundancy of robotic arms and the complexity and constraints of real-world tasks, sampling-based approaches yielded significant results [35], [36]. Among the different algorithmic contributions [108], [110], [118], [119], RRT* [37] is a widely used method that ensures finding the optimal path with probabilistic completeness guarantees [38]. While RRT* is effective in solving path planning tasks in fully observable MDPs, real-world robotics applications are characterized by partial information, casting their settings into POMDP problems. In the real world, robots should make decisions based on information from laser sensors [39], camera images [40], and sensory feedback [41], which generally contains noise, and subsequently makes it hard for planners such as RRT*. Therefore, robot path planning under uncertainty [120]–[123] has become one of the critical topics in the robotics community and remains an open research challenge.

This work proposes an algorithmic formulation to path planning problems based on the popular MCTS algorithm. We argue that the exploration-exploitation properties of MCTS algorithms are essential for robotic path planning in POMDPs, and they can outperform sampling-based planners like RRT* that greedily explore the state-space. To this end, we formulate an MCPP algorithmic framework that we analyze theoretically and provide proofs of convergence for the MDP and POMDP settings. In particular, when applying the upper confidence bounds for Trees (UCT) algorithm [124], we can guarantee the exponential convergence of MCPP to optimal paths in MDP problems. Crucially, we extend our theoretical analysis to prove the probabilistic completeness of MCPP in POMDP problems assuming limited distance observability. To the best of our knowledge, this is the first work to provide the theoretical analysis for MCTS in both MDP and POMDP robot path plannings. We continue by proposing different exploration strategies in MCPP for robotic path planning. In particular, we build on top of our prior work on power-mean UCT (Power-UCT) [43] and convex regularization with Tsallis Entropy Monte-Carlo Planning (TENTS) [44], integrating them in MCPP. We provide various experimental evaluations of MCPP, initially in MDP environments for completeness and thereafter in challenging POMDP tasks in 2D and 3D while planning with a 7-DOF robot arm. Moreover, we evaluate the different variants of MCPP against RRT* in a real-world POMDP experiment (see Fig. 5.2), where the robot can only observe collisions in the box while planning to take out a bunny-toy. Our experimental results confirm that MCPP has a higher probability of solving POMDP path planning tasks with less planning time and requiring fewer samples than the baseline methods. We believe that our theoretical findings and empirical results

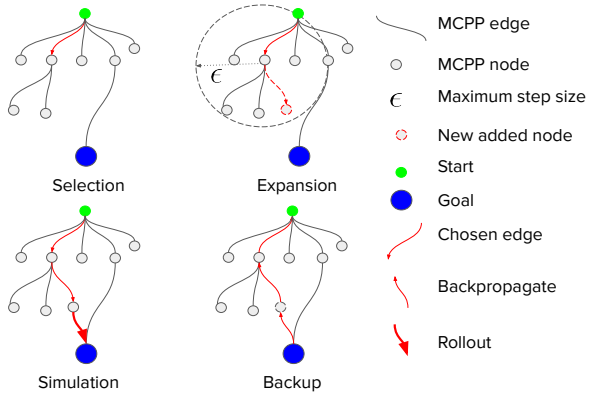


Figure 5.1.: Four stages of MCPP planner to traverse from the initial position (in green color) to the goal position (in blue color).

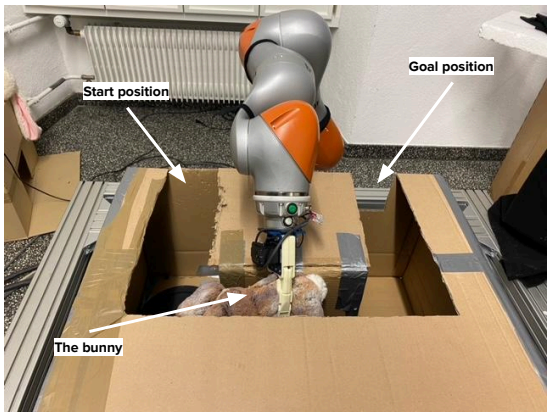


Figure 5.2.: Demonstration of path planning using MCPP in a robotic disentangling task. A 7-DOF robotic KUKA arm tries to extract an object from the cardboard box through the hole in the back of the box. The robot does not use any sensors except for proprioception, making the task partially observable. Therefore, the task requires advanced MCPP-based path planning that takes information gathering about the environment into account. We put a limit to prevent the robot arm to move the hand up, therefore, the robot arm has to find the path from the start position on the left side to the goal position on the right side inside the box.

will shed new light on robotic path planning in complex, partially observable tasks. To

summarize, our *contribution* is threefold:

- we prove that MCPP enjoys exponential convergence in choosing the optimal path in MDP problems and has convergence guarantees to find a feasible path in POMDP environments with limited distance observability (for the POMDP case, we provide the proof sketch);
- using our theoretical insights, we propose an MCTS-based path planning framework that can incorporate different exploration strategies, such as our state-of-the-art methods, Power-UCT, and TENTS, into POMDP path planning problems;
- we provide empirical evaluations in simulation and real-world experiments that confirm our theoretical findings for the MCPP algorithmic framework to be a promising solution for planning in POMDP environments.

5.2. Related Work

Probabilistic RoadMaps (PRMs) [108] and RRTs [37] are fundamental approaches for sampling-based motion planning. RRT* improves over RRT by applying the rewiring technique to shorten the unnecessary traversing path. Moreover, RRT* has proven to guarantee probabilistic completeness for choosing the optimal path in MDP problems, but no convergence rate of RRT* has been studied so far.

There are several heuristic improvements over the state-of-the-art RRT and RRT*. For example, A* is a sufficient heuristic path planning-based method for finding an optimal path given the graph representation of the environment. A*-RRT [125] integrates the benefit of the heuristic A* in RRT by sampling a new tree node using an A* path, and therefore improving the performance in terms of sample efficiency and cost compared to RRT. A*-RRT* [125] combines A* with RRT* to improve the sample efficiency over RRT*. Theta*-RRT [126] considers Theta*, an any-angle discrete search method combined with RRT. Palmieri et al. [126] prove that Theta*-RRT enjoys the probabilistic completeness of RRT and RRT*, while finding shorter trajectories and plans significantly faster than baseline planners (RRT, A*-RRT, RRT*, A*-RRT*). Informed-RRT* [127] focuses the search on the ellipsoidal informed subset of the state-space of the initial running solution found by RRT*.

Regarding applications of MCTS in path planning, Kim et al. [128] proposes the use of Voronoi diagrams to discretize the action space and provides a regret-bound analysis for the sample efficiency, but the authors do not provide a convergence rate for goal

reaching in the robot path planning setting. Sun et al. [129] propose POMCP++, an improvement over POMCP [68] to solve continuous observation problems. First, the authors propose using multiple particle samples from the current initial belief instead of a single particle sample of POMCP. Second, the authors handle the continuous observation space by proposing a new measurement sampling method. At each Q-node in the tree, POMCP++ either samples a new observation or chooses existing observations with some probability. Experiments show that POMCP++ yields a significantly higher success rate and total reward. However, there is no actual convergence rate analysis in the robot path planning settings. Sunberg et al. [130] integrated the progressive widening technique in MCTS to discretize the continuous action and observation cases in POMDP settings and derived POMCPOW and POMCP-DPW (with double progressive widening). The authors further combined a weighted particle filter with progressive widening and showed the benefits over the baseline algorithm Determinized Sparse Partially Observable Tree (DESPOT) [131].

Our work uses a simple uniform discretization of the action space for the MCTS algorithm in the context of robotic path planning. While our approach can apply the Voronoi diagram discretization of [128], in this work, we focus on the theoretical justification of our method and its comparison to sampling-based planners. We provide proofs of convergence for planning the optimal path to the goal in MDPs and a feasible path in POMDPs (with a proof sketch), which is not provided in [129], [130], but can also apply to them. Notably, we propose MCPP as a general MCTS-based framework for robotic path planning. MCPP can incorporate different exploration strategies [43], [44] to continuous actions, adapting, subsequently, the convergence rates for MCPP.

5.3. Background

In this section, we first show the most well-known state of the art methods in robot path planning RRT* showing both its pros and cons. Next, we will briefly give an overview of what is the robot path planning problem in MDP and POMDP setting, show the difficulty of planning in POMDP environment that RRT and RRT* cannot solve. finally, we will shortly summarize the four basic steps of MCTS and show the current advanced Power-UCT and TENTS methods.

5.3.1. RRT*

Rapidly exploring Random Trees (RRT) is the prominent approach to solve the robot path planning tasks. From the start position, RRTs randomly sample the new vertex in the robot configuration space to expand the tree until it gets to the goal position region. The new vertex is satisfied: 1. it lies inside the feasible region, and 2. it lies inside ϵ distance with the closest vertex of the current RRTs tree. Thus, RRT expands the whole space quickly and ensures exponential convergence to find the sub-optimal path to the goal region.

RRT* improves over RRT by rewiring the new sampling vertex with the nearby vertexes to save the unnecessary path. RRT* further ensures the Probabilistic Completeness of finding the optimal path to the goal position.

RRTs* can be applied to fully observable environments but these sampling based approaches can not be applied sufficiently in unstructured, partial observable settings.

Markov Decision Process. A finite-horizon MDP can be defined as a 5-tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$, where $\mathcal{S}$ is the state-space, $\mathcal{A}$ is the finite action-space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition kernel, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi \in \Pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a probability distribution of the event of executing an action a in a state s. Most sampling-based algorithms consider the environment as an MDP. Notably, in robot path planning problems, we know the obstacle space so that when we sample a new vertex, we can determine if the new sampled point lies in the free space or not and then calculate the cost function.

Partially Observable MDP. We consider a finite-horizon POMDP as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{R}, \mathcal{P}_s, \mathcal{P}_o, \gamma \rangle$, where $\mathcal{S}$ is the state-space, $\mathcal{O}$ is the observation space, $\mathcal{A}$ is the finite action-space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\mathcal{P}_s : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition kernel, $\mathcal{P}_o : \mathcal{O} \times \mathcal{A} \to \mathcal{S}$ is the observation dynamics, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi \in \Pi : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ is a probability distribution of the event of executing an action a in an observation o. In POMDP settings, the agent does not observe the full information of the state of the environment, and the decisions are based only on observations. In general, the decision process can be made based either on the history of all past actions and observations $h_t = \{a_0, o_0, a_1, o_1, ..., a_t, o_t\}$, or through the belief of the agent over the state-space [68].

Monte-Carlo Tree Search. MCTS [109] combines tree search with Monte-Carlo sampling in order to build a tree, where states and actions are modeled as nodes and edges, respectively, to compute optimal decisions. The MCTS algorithm consists of a loop of

four steps: *Selection*: start from the root node, interleave action selection and sample the next state (tree node) until a leaf node is reached; *Expansion*: expand the tree by adding a new edge (action) to the leaf node and sampling the next state (new leaf node); *Simulation*: rollout from the reached state to the end of the episode using random actions or a heuristic; *Backup*: update the nodes backward along the trajectory starting from the end of the episode until the root node according to the rewards collected.

UCT [12], [124] is an extension of the well-known UCB1 [45] multi-armed bandit algorithm. UCB1 chooses the arm (action a) using

$$a = \underset{i \in \{1 \ldots K\}}{\arg\max}\, \overline{Q}_{i,T_i(n-1)} + C\sqrt{\frac{\log n}{T_i(n-1)}}. \tag{5.1}$$

where $T_i(n) = \sum_{t=1}^{n} \mathbf{1}\{t = i\}$ is the number of times arm i is played up to time n, and $\overline{Q}_{i,T_i(n-1)}$ is the average reward of arm i up to time $n-1$ and $C = \sqrt{2}$ is an exploration constant. In UCT, the value of each node is backed up recursively from the leaf node to the root node as averaging over the child nodes. At each action selection step in MCTS, each arm in the tree is chosen as the maximum value of nodes in the current non-stationary multi-armed bandit setup, as in (5.1). UCT ensures the asymptotic convergence of choosing the optimal arm at the root node [12].

Power-UCT [43], an improvement over UCT, solves the problem of the underestimation of the average mean and the max-backup operators in MCTS by proposing the use of power mean as the backup operator. Power-UCT has a polynomial convergence rate for choosing the optimal action at the root node. TENTS [44] is derived as a result of Tsallis entropy regularization in MCTS. TENTS has an exponential convergence rate at the root node, which is faster than Power-UCT and UCT. TENTS has a lower value error and smaller regret bound at the root node compared to other regularization approaches.

5.4. Problem formulation

Let us define the robot path planning problem, both for MDPs and POMDPs. Let $\mathcal{X} = (a, b)^d$ be the configuration space of the robot, where $a, b \in \mathcal{R}$ are joint limits in configuration space, with $a < b$, and $d \in \mathcal{N}, d > 0$ denoted the robot's DOF. Let's define $\mathcal{X}_{\text{OBS}}$ as the obstacles region and $\mathcal{X} \backslash \mathcal{X}_{\text{OBS}}$ the open set, and the obstacle-free space as $\mathcal{X}_{\text{FREE}} = cl(\mathcal{X} \backslash \mathcal{X}_{\text{OBS}})$, where $cl(\cdot)$ denotes the closure of a set. The initial condition, or start region, $\mathbf{x}_{\text{INIT}}$ is an element of $\mathcal{X}_{\text{FREE}}$, and the goal region $\mathbf{x}_{\text{GOAL}}$ is an open subset of $\mathcal{X}_{\text{FREE}}$. A path planning problem is defined by the triplet $(\mathcal{X}_{\text{FREE}}, \mathbf{x}_{\text{INIT}}, \mathbf{x}_{\text{GOAL}})$. A trajectory is defined as the map $\tau : [0, T] \to \mathcal{X}_{\text{FREE}}$, where $\tau(0) = \mathbf{x}_{\text{INIT}}$, $\tau(T) = \mathbf{x}_{\text{GOAL}}$. Let's define a function

$\sigma : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as the cost function for moving the robot from the configuration point $\mathbf{x}_i$ to $\mathbf{x}_j$, where $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$. A solution to such a problem is a trajectory that moves the robot from the initial region to the goal region, while avoiding collisions with obstacles and having minimum cost.

Fully observable problem. Here, we assume that we know the state of the environment, i.e., we know the $\mathcal{X}_{\text{OBS}}$ space and $\mathcal{X}_{\text{FREE}}$ regions. Whenever a new point is sampled in the configuration space $\mathcal{X} = (a, b)^d$, we can measure the cost and determine if the point is inside the free space or not.

Partially observable problem. In this setting, we assume that the environment is partially observable, i.e., we only know the start position and the goal position. We do not observe the full state but only observations of the environment and progressively build a belief about the environment's state from observations.

5.5. Monte-Carlo path planning

We wish to transform MCTS into a sampling-based method for solving robot path planning problems when applicable. We build our proposed MCPP approach starting from the UCT algorithm. MCPP and UCT share similar ways of selecting nodes to traverse and back up the value of nodes in the tree. However, we need to make several algorithmic choices to do path planning with UCT. First, we draw an ϵ-ball to limit the maximum distance that the robot can move from the current configuration point. Second, we perform uniform sampling of the configuration points inside the ϵ-ball to discretize the continuous actions in the MDP. Third, we investigate different exploration strategies for MCPP, like in the Power-UCT and TENTS algorithms. We provide a proof of the exponential convergence rate of finding the optimal path in MDPs. Moreover, we connect this result to Power-UCT and TENTS and derive their respective convergence rates for path planning. In POMDPs, we provide a probabilistic completeness guarantee for finding the feasible path to the goal with limited distance observations.

5.5.1. Fully observable environment

In an MDP, the agent knows the full state of the environment. Let us define the start position as $\mathbf{x}_{\text{INIT}}$ and the goal position as $\mathbf{x}_{\text{GOAL}}$. The cost function is the Euclidean distance $d(\mathbf{x}, \mathbf{y})$ between two points $\mathbf{x}$ and $\mathbf{y}$. We want to minimize the total cost, that is, the total distance traveled from the start to the goal position using MCPP. As shown in Fig. 5.1, at

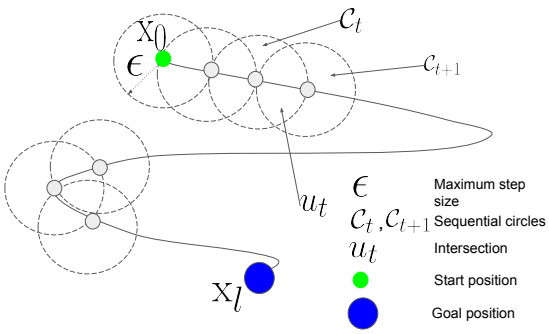


Figure 5.3.: 2-D sketch of the proof for exponential convergence of MCPP to the optimal path in MDPs. The MDP proof relies on showing that MCPP convergences exponentially to a path starting from $\mathbf{x}_0$ and ending at $\mathbf{x}_l$ while the agent stays inside a tube composed of a sequence of spheres with a radius of ϵ.

each node of the tree, starting from the root node, actions are generated by uniformly sampling random points in the ϵ-ball distance from the current node.

The Algorithm 5.1 provides the pseudocode of the MCPP method in the MDP case. The `MainLoop` procedure is the main loop of the algorithm. The algorithm stops when the $\mathbf{x}_{\text{GOAL}}$ position is reached. The algorithm follows the four basic steps of a regular MCTS method. First, at the *Selection* step, we determine the next node to traverse in the tree by selecting the action as in the `SelectAction` procedure. Here, an action is selected based on the UCB algorithm. Note that when we implement Power-UCT [43] we also use UCB, while TENTS [44] uses stochastic Tsallis entropy regularization for the action sampling. Second, at the *Expansion* step, $|A|$ number of actions are generated by uniformly sampling inside the circle $\mathcal{C}(s, \epsilon)$ as shown in the `Expand` procedure. When we reach the leaf node, a new node is created and added to the MCTS tree. Third, at the *Simulation* step, as shown in the `Rollout` procedure, the value function of the current node s is calculated as the distance from that node s to the goal position. Finally, at the *Backup* step, the return value is backpropagated in the two procedures `SimulateV, SimulateQ`.

5.5.2. Partially observable environment

Under partial observability, the agent does not observe the full state of the environment, but has only access to possibly noisy observations. The MCPP planner makes decisions based on the current belief of the agent over the state of the environment. Therefore, our approach in POMDP will be the same as in MDP, except for the fact that we do the

planning in the belief space. The other choice is that MCPP planner can make decisions over the history of actions and observation as if it has some sufficient statistic [132], [133].

```
|A|: number of actions.
N(s): number of simulations of V_Node.
  Default is 0.
n(s,a): number of simulations of Q_Node.
  Default is 0.
r, r(s,a): intermediate rewards defined as the
  distance between two nodes.
V(s): Value of V_Node at state s. Default is 0.
Q(s,a): Value of Q_Node. Default is 0.
γ: discount factor. Default is 1.

R = Rollout(s)
    R = Distance from the current node s to
      the goal position.
    return R

a = SelectAction(s)
    return arg max_a Q(s,a) + C sqrt(log N(s) / n(s,a))

a = Search(s)
    while Time remaining do
        SimulateV (s)
    return arg max_a Q(s,a)

R = Expand(s)
    Generate |A| actions for the current node s
      by randomly sampling |A| via-points
      inside the circle C(s,ϵ)
    discountedReward = Rollout(s)
    return discountedReward

SimulateV(s)
    a = SelectAction (s)
    SimulateQ (s,a)
    N(s) = N(s) + 1
    V(s) = ( Σ_a n(s,a)/N(s) Q(s,a) )

SimulateQ(s,a)
    (s', r) ~ τ(s,a)
    if V(s') not expanded then
        r = r + γ Expand(s')
    else
        SimulateV (s')
    r(s,a) = r(s,a) + r
    n(s,a) = n(s,a) + 1
    Q(s,a) = ( (Σ_a r_{s,a}) + γ Σ_{s'} N(s').V(s') ) / n(s,a)
    where V(s') is the value function of the
      next node by action a from the current
      Q(s,a) node

MainLoop
    while Xgoal is reached do
        a = Search(s)
        Execute(a) in real Robot
```

Algorithm 5.1: Pseudocode of MCPP.

5.5.3. Theoretical analysis

In this section, we prove that MCPP ensures an exponential convergence rate for finding the optimal path from the start position to the goal position in an MDP environment. In a

POMDP setting, we prove that there is a high probability that MCPP can find the path to the goal position.

MDP

First, we make the following assumption.

Assumption 1. *There exists an optimal path from the start position* $\mathbf{x}_{INIT}$ *to the goal position* $\mathbf{x}_{GOAL}$ *with* δ *clearance (minimum distance to an obstacle).*

Based on this assumption, we derive a theorem for the convergence rate of finding the optimal path using MCPP:

Theorem 1. *The probability that MCPP fails to find the optimal path from* $\mathbf{x}_{INIT}$ *to* $\mathbf{x}_{GOAL}$ *after* n *simulations is at most* $ae^{-bf(n)n}$*, for some constants* $a, b \in R_{>0}$.

Proof. Let us consider all feasible paths from the start position ($\mathbf{x}_{\text{INIT}}$) to the goal position ($\mathbf{x}_{\text{GOAL}}$). We will prove that MCPP ensures probabilistic completeness of finding the shortest path from ($\mathbf{x}_{\text{INIT}}$) to ($\mathbf{x}_{\text{GOAL}}$). We will further prove that the failure probability of finding the shortest path decays exponentially for an infinite number of samples.

We choose a ball with radius $\epsilon = \delta$, where δ is the clearance of the shortest path τ^*. Along the path τ^*, we define a set of $l + 1$ circles with the radius ϵ and the center $\mathrm{x}_t(t = 0...l) \in \mathcal{X}_{\text{FREE}}$. Here $\mathrm{x}_0 = \mathbf{x}_{\text{INIT}}$ and $\mathrm{x}_l = \mathbf{x}_{\text{GOAL}}$, as shown in Fig. 5.3. We define each circle $\mathcal{C}_t = (\mathrm{x}_t, \epsilon), t = 0, 1, ...l$. We define the intersection set $u_t = \mathcal{C}_t \cap \mathcal{C}_{t+1}$. Let p be the probability that MCPP can move from $\mathcal{C}_t$ to $\mathcal{C}_{t+1}$. Consider starting from planning node x_t, which is the center of the circle $\mathcal{C}_t$. If the next planning node x_{t+1} lies in the circle $\mathcal{C}_{t+1}$, it has to lie inside the intersection u_t, and we can see that $p < 1/2$. For the robot to travel from x_0 to x_l, it has to use at least l MCPP vertices. Let the probability that MCPP chooses the best action (action with smallest cost) be $f(n)$. Therefore, the probability of MCPP of taking an optimal action that also lies inside u_t is $f(n)p$. The failure probability that MCPP cannot find the shortest path τ^* from x_0 to x_l is $\mathbf{Pr}(X_n < l)$ where X_n is the number of circles $\mathcal{C}_t, t \in 1, 2, ...l$ which are connected by vertices, that is, for an optimal path, all circles need to be connected by vertices. To calculate X_n, the initial value of X_n is zero. We will incrementally increase X_n by one when a new circle along the optimal path is connected with a new vertex. When X_n is equal to or greater than l, we, then, have found the optimal path. Let us upper bound the failure probability $\mathbf{Pr}(X_n < l)$ by

first upper bounding $\mathbf{Pr}(X_n = h)$, as

$$\begin{aligned}\mathbf{Pr}(X_n = h) &\leq \binom{n}{h}(f_0 p)^h(1 - f_0 p)^{n-h} \\ &\leq \binom{n}{h}(f(n)p)^h(1 - f_0 p)^{n-h},\end{aligned} \tag{5.2}$$

where $(f(n)p)^h$ is the upper bound probability of having h circles connected by vertices and $\binom{n}{h}$ makes sure there is at least one consecutive sequence of connected circles. f_0 is the initial probability of choosing the optimal action. equation 5.2 can be explained as $H(x) = x^h(1-x)^{n-h}$ is a decreasing function. This yields an upper bound for $\mathbf{Pr}(X_n = h)$

$$\begin{aligned}\mathbf{Pr}(X_n = h) &\leq \binom{n}{h}(f(n)p)^h(1 - f_0 p)^{n-h} \\ &\leq \binom{n}{h}(f(n)p)^h(1 - \alpha f(n)p)^{n-h}\end{aligned}$$

where $f_0 = \alpha$ is a constant and $f(n) \leq 1$ so that $1 - f_0 p \leq 1 - \alpha f(n)p$. The probability of failing to find the optimal path is then

$$\begin{aligned}\mathbf{Pr}(X_n < l) &= \sum_{h=0}^{l-1} Pr(X_n = h) \\ &\leq \sum_{h=0}^{l-1} \binom{n}{h}(f(n)p)^h(1 - \alpha f(n)p)^{n-h} \\ &\leq \sum_{h=0}^{l-1} \binom{n}{l-1}(f(n)p)^h(1 - \alpha f(n)p)^{n-h} \text{ (as } l << n) \\ &\leq \binom{n}{l-1} \sum_{h=0}^{l-1}(1 - \alpha f(n)p)^n \\ &\text{(as } f(n)p < 1/2 \text{ so that } f(n)p < 1 - f(n)p < 1 - \alpha f(n)p) \\ &\leq \binom{n}{l-1} \sum_{h=0}^{l-1}(e^{-\alpha f(n)p})^n = \binom{n}{l-1} l e^{-\alpha f(n)pn} \\ &\text{(as } 1 - \alpha f(n)p <= e^{-\alpha f(n)p}) \\ &= \frac{\prod_{i=n-l}^{n} i}{(n-1)!} l e^{-\alpha f(n)pn} \leq \frac{l}{(l-1)!} n^l e^{-\alpha f(n)pn} \leq a e^{-bf(n)pn}\end{aligned}$$

The provided convergence rate proves that the MCPP algorithm is probabilistically complete and converges to the optimal path exponentially. □

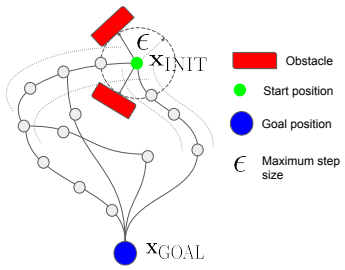


Figure 5.4.: Sketch of how to generate paths for MCPP algorithm from $\mathbf{x}_{\text{INIT}}$ to $\mathbf{x}_{\text{GOAL}}$ positions with minimum number of via-points in POMDP environments.

Let us define $g(t)$ as the failure probability of finding the optimal path from $\mathbf{x}_{\text{INIT}}$ to $\mathbf{x}_{\text{GOAL}}$ after t time steps. We derive the following corollaries:
With Power-UCT, $f_{\text{Power-UCT}} = 1 - (\frac{1}{t})^{\alpha})t$. The probability that the MCPP using Power-UCT fails to find the path from x_0 to x_l is as follows.

Corollary 7. *Power-UCT*
$g_{\textit{Power-UCT}}(t) = ae^{-b(1-(\frac{1}{t})^{\alpha})t}$, *where* $0 < \alpha < 1, a, b \in \mathbb{R}_{>0}$

With TENTS $f_{\text{TENTS}} = 1 - ct\exp\{-\frac{t}{\hat{c}(\log t^3))t}\}$. The probability that MCPP using TENTS fails to find the path from x_0 to x_l is

Corollary 8. *TENTS*
$g_{\textit{TENTS}}(t) = ae^{-b(1-ct\exp\{-\frac{t}{\hat{c}(\log t^3))t}\}}$, *where* $a, b, c, \hat{c} \in \mathbb{R}_{>0}$

The results show that MCPP-TENTS robot path planning converges faster compared to MCPP-Power-UCT.

POMDP

First, we make the following assumption.

Assumption 2. *The agent observes the environment only up to γ distance.*

This assumption is reasonable in many robotic settings, e.g., for mobile robotics. Based on this assumption and Assumption 1, we derive a theorem to show that with high probability, the MCPP algorithm can find the feasible path to the goal position in a POMDP environment:

Theorem 2. *In POMDP environments with limited distance observability, MCPP will find a path from the start position $\mathbf{x}_{\textit{INIT}}$ to the goal position $\mathbf{x}_{\textit{GOAL}}$ with high probability.*

Proof. We assume that there is a finite number of feasible paths $(\tau_1, \tau_2, ...\tau_K)$ to go from the start position $\mathbf{x}_{\text{INIT}}$ (or $\mathbf{x}_0$) to the goal position $\mathbf{x}_{\text{GOAL}}$. Each feasible path τ_i has at least δ_i clearance from obstacles. We choose $\epsilon = \min\{\delta_1, \delta_2, ...\delta_K, \gamma\}$. γ is the observation distance defined in Assumption 2.

Along each path τ_i, let us define a set of circles $\mathcal{C}_i = (\mathbf{x}_i, \epsilon), i = 0, 1, ...l_{\tau_i}$. as shown in Fig. 5.3. Let us define $\mathbf{C}$ as the set of all circles (along all the feasible paths that we define). We assume that if the agent collides, the agent moves back to the last planning point and will not go to the direction of the obstacle again with high probability. We define that the probability $p_{\text{collision}} \to 0$ when the time step $t \to \infty$. We prove that with high probability, the agent can find the path from the start position $\mathbf{x}_{\text{INIT}}$ to the goal position $\mathbf{x}_{\text{GOAL}}$.

The proof is derived by induction. From the start position $\mathbf{x}_0$, there is a finite number of circles $\mathcal{C}_i \in \mathbf{C}$ as the next feasible region that the MCPP planner can sample as the next node in the tree (MCPP samples the next planning point inside the ϵ-ball distance). Because the probability of colliding again is $p_{\text{collision}} \to 0$ when the time step $t \to \infty$, and the MCPP objective is to minimize the cost to go to the goal position. When we increase the number of samples, the next planning node will lie inside the circle that contains the optimal path. Therefore, with high probability $1 - p_{\text{collision}}$ the next MCPP node will be inside one of the circles $\mathcal{C}_i$. Assume now that the agent is inside the circle $\mathcal{C}_i$. Using the same induction, there is high probability $1 - p_{\text{collision}}$ that the next MCPP node will be inside one of the next circle $\mathcal{C}_{i+1}$. $\mathcal{C}_{i+1} \in \mathbf{C}$. Since the number of circles is finite, the agent will get to the goal region after a certain number of time steps with high probability, concluding the proof. □

5.6. Experiments

In this section, we evaluate the performance MCPP in challenging POMDP environments. In MCPP, we apply the two recent advanced improvement techniques in MCTS, Power-UCT and TENTS, along with the baseline MCTS method, UCT. We compare our new robot path planning methods against the baseline sampling-based method RRT*, and a state-of-the-art continuous action POMDP solver POMCP-DPW [130]. In simulation, we also compared against two different simple heuristic methods. The first method puts a ball around the agent to sample the next point. We use the same step size (the ball's diameter) and the same number of samples as MCPP and RRT*. In the second heuristic, we use an ϵ-greedy probability (1%) to sample the goal position or otherwise the next node, similarly to random node sampling in RRT*. We do not put any restrictions on the

step size to sample the next node.

The POMDP setting of our experiment is the same as in [134]. Similar to the path planning definition in Sec. 5.4, the state space $\mathcal{X}$ consists of configuration space coordinates such as robot joint angles. The action space $\mathcal{A}$ is identical to the state space, consisting of target configuration space coordinates defining where to move the robot. We use linear interpolation to move the robot from the current configuration to the next one. The observation is a configuration in collision with an obstacle in obstacle space $\mathcal{X}_{\text{OBS}}$, or the goal configuration $\mathbf{x}_{\text{GOAL}}$, when the robot reaches the goal. We assume static obstacles and deterministic transition and observation probabilities $\mathcal{P}_s$ and $\mathcal{P}_o$. We define the reward function as a success pseudo-probability along the path from one configuration (x_1) to another (x_2) $\mathcal{R}(\mathrm{x}_1, \mathrm{x}_2) = P_{\text{SUCCESS}}(\mathrm{x}_1, \mathrm{x}_2)$ where $P_{\text{SUCCESS}}(\mathrm{x}_1, \mathrm{x}_2)$ is defined in [134]. We set the discount factor $\gamma = 1$ and limit the planning horizon. As in [134], to approximate the belief over states, based on prior collisions, we compute a probabilistic map that assigns a probability of colliding to any given position in the environment. The belief distribution is able to represent multi-modal and asymmetric belief distributions (see Fig. 2 in [134]). Initially, we assume a non-zero probability of colliding at any location on the map. After each collision, we update the map by assigning a failure probability that takes into account the collision coordinates and the movement direction (see Fig. 2 in [134]).

Following the previous POMDP definition, we evaluate the methods in simulation in two 7-DOF configuration space POMDP tasks with 2D and 3D task spaces. Finally, we compare MCPP to RRT* in a real robot POMDP disentangling application, similar to the one described in [134].

5.6.1. Experimental evaluation in simulation

We provide three simulation settings in 2D and 3D state spaces to demonstrate that the MCPP planner is more explorative and can easily solve POMDP path planning problems compared to baselines. First, in a 2D U-Shape problem (Fig. 5.5), the start position is in green color while the goal position is in blue color. We compare UCT, Power-UCT, and TENTS compared to RRT* and POMCP-DPW. As shown in Table 5.1, over 100 random seeds with the same number of samples (500), UCT and Power-UCT obtain 93% and 95% success rate, respectively, with approximately the same number of collisions. TENTS is less explorative, with 31% success rate and 18.3 collisions. POMCP-DPW gets 46% success rate and 22.7 collisions while RRT* gets 76% success rate with 14.6 collisions. The benefits of MCPP over RRT* can be explained as, even using a similar representation with the

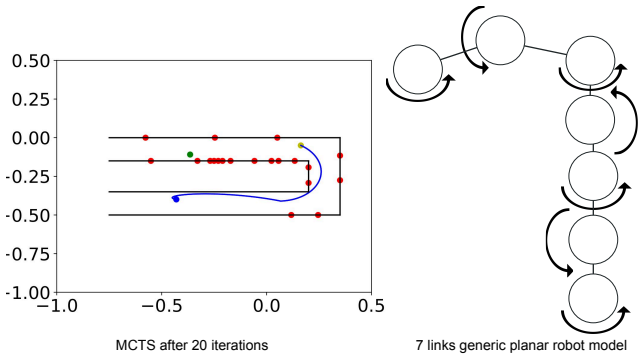


Figure 5.5.: U-Shape 2D POMDP. Green point is the start position. Blue point is the goal position. Red points are the collisions. The figure shows a success case using MCPP, where the blue line depicts the 2D trajectory of the end effector. Note that in all the 2D experiments we plan in the configuration space using a 7-DOF planar robot arm model illustrated on the right.

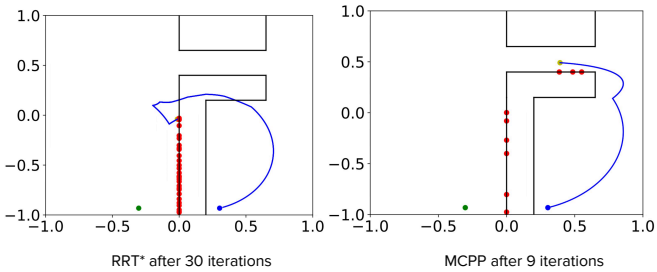


Figure 5.6.: L-Shape 2D POMDP. Green point is the start position. Blue point is the goal position. Red points are the collisions. The blue lines are the planning path. The figure shows a failure case of RRT* and a success case for MCPP, which shows that it is more explorative. Over 20 random seeds, RRT* failure to solve the problem with 0% success, while MCPP obtain 100% success with UCT and Power-UCT. TENTS gets 85%.

updating belief (probabilistic map), MCPP makes decisions based on the value function of the POMDP (by building a multistep look-ahead forward tree search), which is more explorative towards the goal. In contrast, each step decision of RRT* will be more greedy in choosing the smallest cost. Meanwhile, MCPP shows the benefit of uniformly sampling the actions inside the ϵ-ball compared to POMCP-DPW, which restricts the number of actions by using the progressive widening technique. Both of the two random heuristic

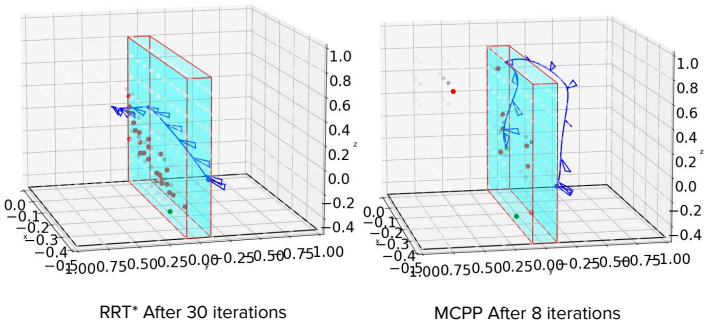


Figure 5.7.: High Wall Environment in 3D. Grey point is the start position. Blue point is the goal position. Red points are collisions. Over 20 random seeds, RRT* can only success with 35%, UCT obtains 55% success rate. Power-UCT gets 70% success rate while TENTS gets 45% success rate.

baselines fail to solve this task. We demonstrate one more 2D POMDP experiment with an L shape obstacle (Fig. 5.6. Over 20 random seeds with 500 samples, RRT* fails to solve the problem, while UCT and Power-UCT obtain 100% success rate. TENTS is less explorative with 85% success rate. POMCP-DPW obtains 85% success rate. The two heuristic methods fail to solve this task.

Third, we build a High-wall 3D POMDP (Fig. 5.7). In this problem, the start position (green color) and the goal position (blue color) are very close, while there is a high wall standing between. The agent is not aware of the existence of the wall. As we can see in Table 5.2, for 20 random seeds with the same number of samples (500), UCT obtains 55% success rate with 16.4 collisions on average. Power-UCT gets a higher success rate with 70% and 15.7 collisions on average. TENTS is less explorative with 45% success rate and 23.0 collisions on average. POMCP-DPW gets 70% success rate and 15.9 collisions. On the other hand, RRT* can only obtain 10% success rate and 26.0 collisions. Finally, the first baseline heuristic method fails to solve this task, while the second one achieves 15% success rate.

5.6.2. Real robot object disentangling task

We compare MCPP against RRT* in the real-robot disentangling POMDP problem, as in [134]. We use a 7-DOF KUKA LBR robot arm equipped with a SAKE gripper. Fig. 5.2 illustrates the intermediate scenario of the robot arm trying to reach the goal position in

Table 5.1.: Comparison for the U-Shape 2D POMDP

Methods	Time(second)	Collisions	Success Rate
RRT*	1555±229	14.6±1.5	76%
UCT	141.7±16.3	15.9.±1.7	93.0%
Power-UCT	146±17	15.8±1.7	95.0%
TENTS	179.5±16	18.3±1.7	31%
POMCP-DPW	322±30.8	22.7±1.6	46%

Table 5.2.: Comparison for the High-Wall 3D POMDP

Methods	Time(second)	Collisions	Success Rate
RRT*(bias=1)	2854.6±4.1	26.0±0.4	10%
RRT*(bias=100)	2080.3±2.6	19.4±1.7	35%
RRT*(bias=200)	2548.1±2.7	22.2±1.	55%
UCT	178.8±13.0	16.4±1.0	55%
Power-UCT	208±18.5	15.7±1.2	70.0%
TENTS	267.6±17.1	23.0±1.2	45.0%
POMCP-DPW	215.56±23.9	15.9±1.4	70.0%

Table 5.3.: Comparison for the real robot object disentangling

Methods	Time(second)	Collisions	Success Rate
RRT*	1099 ±356	10.25 ±3	40%
UCT	346.2 ±64	20.2 ±3	70%
Power-UCT	436.5 ±177	22.5 ±6	40.0%
TENTS	428.5 ±133	25 ±6	20.0%

the unknown box environment, while trying to disentangle the toy-bunny that was lying inside the box (the grasp part was pre-programmed). The robot does not know the shape of the box.

To evaluate the performance of our MCPP variants, we run both UCT, Power-UCT, and TENTS. We run 10 random seeds, each random seed with 500 number of samples, and perform 30 iterations to determine if the planners can reach the goal position or not. After each iteration, if the robot hits a collision, the robot moves back a bit from the last collision position and performs the planning again with the new start position. The detailed results are shown in Table 5.3. While RRT* can get 40% success rate over ten random seeds, UCT achieves 70% which shows the benefits of MCPP in a real-world POMDP. Power-UCT achieves 40% success rate, and TENTS can only succeed 20% of the times. In terms of

time, the average time in all success cases of UCT, Power-UCT, and TENTS are 364.2 seconds, 436.5 seconds, and 428.5, respectively, which are much faster compared to RRT* with an average of 1099 seconds. This can be explained as RRT* spends more time in performing the sorting to find the nearest node, as it is more biased to grow towards large unsearched areas.

5.7. Conclusions

This chapter addressed the major challenges of planning robot paths under partial observability from a theoretical perspective, deriving a new framework for applying MCTS planning in continuous action spaces for robot path planning. We theoretically analyzed our proposed Monte-Carlo Path Planning (MCPP) approach and proved an exponential convergence rate for MCPP for choosing the optimal path in fully observable MDPs and probabilistic completeness for finding a feasible path in POMDPs. Moreover, MCPP allows us to integrate different established exploration strategies in MCTS literature to improve exploration for path planning. We empirically analyzed our MCPP variants in benchmarks for POMDP path planning problems, showing superiority in terms of performance and computational time compared to RRT*, and POMCP-DPW. We further applied our new method to a real robot POMDP problem using a KUKA 7-DOF robot arm to disentangle objects from other objects, without any sensory information, except for the observation of collisions. Future development involves the application of MCPP to more complicated robotic tasks and studying heuristics to accelerate the planning process with MCPP.

6. Conclusion

"Let us go over, and sit in the shade of the trees."
— *Stonewall Jackson*

This thesis proposed and studied advanced MCTS methods both in terms of theoretical and experimental perspectives to tackle the following open MCTS problems: *sample efficiency*, *exploration-exploitation* trade-off and transform these advanced methodologies to solve robot path planning tasks, showing superior performances both in simulations and real robotics in POMDP settings. First, we investigated the benefits of using power mean as a novel value backup operator, called Power-UCT, compared to baselines. Second, we provided a comprehensive study of the use of convex regularization, and an in-depth study of entropy based regularization in MCTS. Third, we introduced the use of α-divergence in MCTS and theoretically connected the two works Power-UCT and entropy regularization using α-divergence. Last, we transformed our advanced MCTS methods to robot path planning tasks and showed the benefits in POMDPs. Detailed contributions are shown below.

6.1. Summary of Contributions

In Chapter 2, we proposed to use power mean as a novel backup operator in MCTS and derived a variant of UCT based on this operator, which we called Power-UCT. In detail, Power-UCT and UCT [12] share the same way of node selection strategy in the tree using UCB [42]. However, Power-UCT differentiates from UCT by replacing the average value backup operator in UCT using the power mean backup operator. Furthermore, we theoretically provided an asymptotic convergence guarantee of Power-UCT to the optimal value, given that the value computed by the power mean lies between the average and the maximum, alleviating the problem of underestimating of the average backup operator in UCT. We eventually showed how Power-UCT could help with *sample efficiency*

by empirically providing advantages of Power-UCT in stochastic MDPs and POMDPs w.r.t. other baselines.

In Chapter 3, we provided an in-depth study of the use of convex regularization in MCTS. We proposed a new algorithm called Extended Empirical Exponential Weight (E3W), derived the first regret analysis of the regularised MCTS and showed that it guarantees an exponential convergence rate to choose the optimum. We took a further step to study different kinds of entropy regularization in MCTS and respectively derived Maximum Entropy Monte-Carlo Planning (MENTS), Relative Entropy Monte-Carlo Planning (Relative entropy Monte-Carlo planning (RENTS)), and Tsallis Entropy Monte-Carlo Planning (TENTS). We compared the benefits of the three methods both in terms of theory and experiments, showing the superiority of TENTS over MENTS and RENTS. We shed a light on how different entropy regularization methods help to balance exploration and exploitation in the tree. In detail, MENTS benefits of more explorations due to the softmax form of the regularized policy, while TENTS has the advantages of the sparse policy and can effectively exploit, especially in environments with high branching factors.

In Chapter 4, we introduced the use of α-divergence in MCTS, showed how to establish the connection between the two works (Power-UCT and E3W) using α-divergence, and provided a rigorous theoretical study of α-divergence in MCTS. In detail, we used the α-divergence function as the regularizer to introduce novel regularized backup operators for MCTS, relatively derived the maximum entropy, the relative entropy of the policy update, and, more importantly, derived the Tsallis entropy of the policy those has been proposed in E3W. Furthermore, we generalized the definition of average mean by considering the α-divergence function as the probability distance and deriving power mean, which has been used as a novel backup operator in Power-UCT. Finally, we showed how the α parameter could help to balance between exploration and exploitation in Synthetic Tree environments.

Chapter 5 focused on solving major challenges of planning robot paths under partial observability, in which the current state-of-the-art method RRT* is only designed for full observability. We transformed the use of MCTS to solve the robot path planning tasks in continuous action spaces. We called our approach Monte-Carlo Path Planning (MCPP) and proved an exponential convergence rate for MCPP for choosing the optimal path in fully observable MDPs and probabilistic completeness for finding a feasible path in POMDPs (proof sketch). Furthermore, we showed how the current advanced MCTS methods in literature could be integrated into MCPP and subsequently demonstrated the superiority in terms of performance and computational time compared to RRT*, and other POMDP solvers Partially Observable Monte-Carlo Planning Double Progressive Widening

(POMCP-DPW) [130] in both simulations and real robotic tasks.

6.2. Open Problems

Despite having state-of-the-art methods that help to solve current open Monte-Carlo Tree Search (MCTS) problems and achieving promising results when transforming these advanced methods to apply to robot path planning tasks, there are still, however, many open challenges associated with our proposed works that will be addressed here.

6.2.1. Maximum Expected Value Estimation of Power Mean

Maximum expected value estimation plays an important role in many reinforcement learning (RL) algorithms [135], such as Q-learning [49], [136], [137] and MCTS [138]. However, despite promising results, many current approaches in the literature often introduce biases and/or variances [49], [138]. One can think of using the maximum estimator. However, the maximum is overestimated [32]. In MCTS, an effective value estimator should select the optimal branch for faster convergence in order to select optimal actions. The value returned by the power mean is between the average mean estimator and the maximum estimator and may increase/decrease based on the p coefficient; therefore, it may help to make a trade-off between bias/variance. As the number of samples in a given node increases, we can increase the p coefficient value to move our estimator towards the optimal value. Nevertheless, in our current approach, we consider the p-coefficient as a fixed constant, and therefore an open question arises: can we have an approach for adjusting the p-coefficient in Power-UCT?

6.2.2. Uncertainty Value Estimation

Value estimation is challenging in stochastic environments with partial observability. Current approaches, which consider the value function in the tree as a single value, do not consider the uncertainty of the reward function. Therefore, these methods cannot represent the uncertainty of the environment, which in return prevents the agent from making decisions, especially in an environment that needs to be explored. Our introduced convex regularization framework inspired by the Legendre-Fenchel transform [139] provides a guarantee for the exponential convergence rate to select the optimum and can help to find a tradeoff between exploration and exploitation. However, by representing each value node as a single value function, it cannot properly model the uncertainty from

the reward function and the stochasticity of the dynamic transition of environments [140]. This issue presents an open challenge for MCTS researchers in terms of modeling and backpropagating the uncertainty of value nodes in tree search.

6.2.3. Efficient Monte-Carlo Planning for Autonomous Car Driving

Autonomous driving has attracted significant attention from the scientific community in recent years. Especially with the rapid progress of artificial intelligence nowadays, the application of deep learning for vehicle planning in complex environments has become a trendy direction and has achieved promising results [141]–[144]. [144] developed a deep Monte Carlo tree search control method (deep-MCTS) for vision-based autonomous driving by maintaining two deep neural networks. One network is used to predict state from observed camera images of environments. The other network is trained to predict probabilistic action distributions and value functions of specific states. Based on the prior knowledge of these two neural networks, an MCTS tree is constructed to output the optimal trajectory for autonomous vehicle control. Although Deep-MCTS achieved significant performance results compared to Deep Deterministic Policy Gradient (DDPG) [145] and Deep Q-Network (DQN) [146], Deep-MCTS still suffers from the sample efficiency problem due to polynomial convergence rate of the UCT [124] method used. This problem, combined with the costly computations due to the slow processing time of camera image acquisition and deep neural network evaluation, prevents the applicability of the method in real vehicle scenarios. These problems open the need for an efficient planning method for vehicle planning.

6.3. Future Work

Possible future work includes proposals of a theoretically justified or heuristic approach to adapt the greediness of power mean, combining our advanced methods with AlphaGo algorithm [28] in autonomous car driving and introducing the use of α-function [147] in Wasserstein Barrycenter [148] to backpropagate uncertainty.

We are going to present these future directions as below.

6.3.1. p-Adaptation

To take advantage of Power-UCT, it is desirable to have a different p coefficient for each node and adapt its value during training according to the number of visits. In particular, we decrease p when the number of visits is small since the average operator performs better in this case; on the contrary, we increase it when the number of visits is large, which is the case where the maximum operator is more effective. For future work, we plan a heuristic adaptive procedure to tune p as p value increases a fixed step size length when the number of visits at that node increases. The intuitive motivation of this approach is supported by its theoretical guarantees of convergence for any value of p, which we prove in the Chapter 2. To the best of our knowledge, this is the first work to show an adaptive method of backup value in MCTS that still ensures convergence to the optimal value.

Even if the heuristic approach to adapting the greediness of power mean has an intuitive explanation, we plan to study a more rigorous and theoretically justified way of doing it. Future work could be analyse the bias and variance of the power mean estimator or provide theoretical analysis for the adaptation of p coefficient of Power-UCT in MCTS.

6.3.2. Wasserstein Monte-Carlo Tree Search

A single value function cannot fully represent the uncertainty of the environment. A promising approach is to consider each value function as a probabilistic distribution. There are several works in MCTS [51], [149]–[151] and in RL [92], [152], [153] following this idea.

Recently, [140] discussed how to model and propagate the uncertainty of value functions in temporal difference (TD) learning using L^2-Wasserstein barycenters, and applied it to Q-learning to demonstrate exploration benefits in different environments. The Wasserstein distance comes from the optimal transport community. Intuitively, it represents the "cost" of moving the probability mass from one distribution to another. [140] uses the L^2-Wasserstein barrycenter in combination with the Euclidean distance to derive the Wasserstein Q-learning (WQL) method, which outperforms Q-learning in some toy tasks and Atari games. To adopt this idea, one can think of applying it to MCTS to solve highly stochastic tasks. However, TD learning backpropagation is very sensitive to step-size learning and easily converges to the local optimum. Moreover, TD learning introduces errors at each learning step, which prevents the applicability of the approach compared to baselines.

Taking inspiration from this direction, we propose to learn value posterior functions using L^1-Wasserstein Barycenters and instead use the α-function to replace the Euclidean distance. Consequently, when considering each node in the tree as a Gaussian distribution, we can derive a close form solution to compute the mean, and standard deviation of each value node as the power mean backup operator of the mean and standard deviation of next Q value nodes. Action selection could be Optimistic Sampling [51] and Thompson Sampling [150].

6.3.3. AlphaGo for Autonomous Car Driving

Following the successes of our advanced MCTS methods, which have already shown their theoretical and experimental superiority in various environments, a possible future direction could be to apply Power-UCT and E3W (especially with the promising results of TENTS in AlphaGo) to autonomous driving tasks using pre-trained Deep Neural Networks. Specifically, the power mean backup operator can replace the inefficient average mean backup operator of the current UCT strategy in Deep-MCTS. The returned power mean solves the underestimation problem of the average mean backup operator and the overestimation problem of the maximum value backup operator, improving the Monte Carlo planning performance. Moreover, one can think of using entropy regularizations in MCTS for autonomous driving since the advantages of MENTS, RENTS, and especially TENTS have already been illustrated by using the AlphaGo algorithm in Atari games in Chapter 3. The results and advantages of it due to the guarantee of exponential convergence rate to the optimum of the convex regularization framework, such an approach is conceivable as shown in Chapter 3.

A. Appendix

A.1. Generalized Mean Estimation in Monte-Carlo Tree Search

We derive here the proof of convergence for Power-UCT. The proof is based on the proof of the UCT [12] method but differs in several key places. In this section, we show that Power-UCT can smoothly adapt to all theorems of UCT [12]. The following results can be seen as a generalization of the results for UCT, as we consider a generalized mean instead of a standard mean as the backup operator. Our main results are Theorem 6 and Theorem 7, which respectively prove the convergence of failure probability at the root node, and derive the bias of power mean estimated payoff. In order to prove them, we start with Theorem 1 to show the concentration of power mean with respect to i.i.d random variable X. Subsequently, Theorem 2 shows the upper bound of the expected number of times when a suboptimal arm is played. Theorem 3 bounds the expected error of the power mean estimation. Theorem 4 shows the lower bound of the number of times any arm is played. Theorem 5 shows the concentration of power mean backup around its mean value at each node in the tree.

We start with well-known lemmas and respective proofs: The following lemma shows that Power Mean can be upper bound by Average Mean plus with a constant

Lemma 2. *Let* $0 < l \leq X_i \leq U, C = \frac{U}{l}, \forall i \in (1, ..., n)$ *and* $p > q$. *We define*

$$Q(X, w, p, q) = \frac{M_n^{[p]}(X, w)}{M_n^{[q]}(X, w)}$$
$$D(X, w, p, q) = M_n^{[p]}(X, w) - M_n^{[q]}(X, w).$$

Then we have

$$
\begin{aligned}
&Q(X,w,p,q) \le L_{p,q} \\
&D(X,w,p,q) \le H_{p,q} \\
&L_{p,q} = \left(\frac{q(C^p - C^q)}{(p-q)(C^q-1)}\right)^{\frac{1}{p}} \left(\frac{p(C^q - C^p)}{(q-p)(C^p-1)}\right)^{-\frac{1}{q}} \\
&H_{p,q} = (\theta U^p + (1-\theta)l^p)^{\frac{1}{p}} - (\theta U^q + (1-\theta)l^q)^{1/q},
\end{aligned}
$$

where θ is defined in the following way. Let

$$h(x) = x^{\frac{1}{p}} - (ax+b)^{1/q}$$

where

$$
\begin{aligned}
a &= \frac{U^q - l^q}{U^p - l^p} \\
b &= \frac{U^p l^q - U^q l^p}{U^p - l^p} \\
x^{'} &= \arg\max\{h(x), x \in (l^p, U^p)\}
\end{aligned}
$$

then

$$\theta = \frac{x' - l^p}{U^p - l^p}.$$

Proof. Refer to [66]. □

Lemma 3. *Let X be an independent random variable with common mean μ and $a \le X \le b$. Then for any t*

$$\mathbb{E}[\exp(tX)] \le \exp\left(t\mu + t^2\frac{(b-a)^2}{8}\right) \tag{A.1}$$

Proof. Refer to [154] page 67. □

Lemma 4. *Chernoff's inequality $t > 0$,*

$$\mathbf{Pr}(X > \epsilon) \le \exp(-t\epsilon)\mathbb{E}[\exp(tX)] \tag{A.2}$$

Proof. This is a well-known result. □

The next result show the concentration Inequality of Power Mean estimation

Theorem 1. *If $X_1, X_2, ..., X_n$ are independent with* $\mathbf{Pr}(0 \leq X_i \leq 1) = 1$ *then for any $\epsilon > 0$, $p \geq 1, \exists C_p > 0$ that*

$$\mathbf{Pr}\left(\left|\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}} - \mathbb{E}\left[\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}}\right]\right| > \epsilon\right) \leq 2\exp\left(-C_p n\epsilon^2\right)$$

Proof. Apply the Theorem 6 [155], we have

$$\frac{\sqrt{\text{Var}(\| X \|_p)}}{\mathbb{E}(\| X \|_p)} \simeq \frac{1}{\sqrt{n}}\left(\frac{1}{p}\frac{\sigma_p}{\mu_p}\right) \tag{A.3}$$

where $\| X \|_p$ is p-norm. $\mu_p = \mathbb{E}[X^p], \sigma_p^2 = \text{Var}(X^p)$
Then

$$\text{Var}\left(\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}}\right) \simeq \frac{1}{n}\left(\frac{1}{p}\frac{\sigma_p}{\mu_p}\mathbb{E}\left[\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}}\right]\right)^2 \tag{A.4}$$

Let define $\frac{1}{C_p} = \left(\frac{1}{p}\frac{\sigma_p}{\mu_p}\mathbb{E}\left[\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}}\right]\right)^2$ We have

$$\text{Var}\left(\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}}\right) \simeq \frac{1}{nC_p} \tag{A.5}$$

With $\sigma^2 = \text{Var}(X)$, Applying Lemma 3 and Lemma 4 we have

$$\mathbf{Pr}(|X - \mathbb{E}[X]| > \epsilon) \leq 2\exp\left(-\frac{\epsilon^2}{\sigma^2}\right) \tag{A.6}$$

Apply (A.6) for the power mean of $X_1, X_2, ..., X_n$, and based on the result of A.5, we get the result of Theorem 1. □

With $\triangle_n = 9\sqrt{\frac{1}{C_p} n \log(\frac{2}{\delta})}$, ($\delta > 0$ are constant), and $\mu_p = \mathbb{E}\left[\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}}\right]$, apply Theorem 1, we get

$$\mathbf{Pr}\left(\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}} - \mu_p > \frac{\triangle_n}{9n}\right) \leq \exp\left(-C_p n\left(\frac{\triangle_n}{9n}\right)^2\right).$$

Therefore,

$$\mathbf{Pr}\left(\left(\frac{\sum_{i=1}^{n} X_i^p}{n}\right)^{\frac{1}{p}} - \mu_p > \frac{\triangle_n}{9n}\right) \leq \exp\left(-C_p \frac{1}{C_p} \log\left(\frac{2}{\delta}\right)\right) = \frac{\delta}{2}. \tag{A.7}$$

we have

$$\mathbf{Pr}\left(\left(\frac{\sum_{i=1}^{n} X_i^p}{n}\right)^{\frac{1}{p}} - \mu_p > \frac{\triangle_n}{9n}\right) \leq \frac{\delta}{2}. \tag{A.8}$$

Let's start with an assumption

Assumption 1. *Fix $1 \leq i \leq K$. Let $\{F_{it}\}_t$ be a filtration such that$\{X_{it}\}_t$ is $\{F_{it}\}$-adapted and $X_{i,t}$ is conditionally independent of $F_{i,t+1}, F_{i,t+2}, \ldots$ given $F_{i,t-1}$. Then $0 \leq X_{it} \leq 1$ and the limit of $\mu_{in} = \mathbb{E}[\overline{X_{in}}(p)]$ exists, Further, we assume that there exists a constant $C > 0$ and an integer N_c such that for $n > N_c$, for any $\delta > 0$, $\triangle_n(\delta) = C\sqrt{n \log(1/\delta)}$, the following bounds hold:*

$$\mathbf{Pr}(\overline{X}_{in}(p) \geq \mathbb{E}[\overline{X}_{in}(p)] + \triangle_n(\delta)/n) \leq \delta, \tag{A.9}$$

$$\mathbf{Pr}(\overline{X}_{in}(p) \leq \mathbb{E}[\overline{X}_{in}(p)] - \triangle_n(\delta)/n) \leq \delta. \tag{A.10}$$

Under Assumption 1, For any internal node arm k, at time step t, let define $\mu_{kt} = \mathbb{E}[\overline{X}_{kt}(p)]$, a suitable choice for bias sequence is that $c_{t,s} = 2C\sqrt{\frac{\log t}{s}}$ (C is an exploration constant) used in UCB1 (using power mean estimator), we get

$$\mathbf{Pr}\left(\left(\frac{\sum_{i=1}^{s} X_{ki}^p}{s}\right)^{\frac{1}{p}} - \mu_{kt} > 2C\sqrt{\frac{\log t}{s}}\right) \leq t^{-4} \tag{A.11}$$

$$\mathbf{Pr}\left(\left(\frac{\sum_{i=1}^{s} X_{ki}^p}{s}\right)^{\frac{1}{p}} - \mu_{kt} < -2C\sqrt{\frac{\log t}{s}}\right) \leq t^{-4}. \tag{A.12}$$

From Assumption 1, we derive the upper bound for the expectation of the number of plays a sub-optimal arm

Theorem 2. *Consider UCB1 (using power mean estimator) applied to a non-stationary problem where the pay-off sequence satisfies Assumption 1 and where the bias sequence, $c_{t,s} = 2C\sqrt{\log t/s}$ (C is an exploration constant). Fix $\epsilon \geq 0$. Let $T_k(n)$ denote the number of*

plays of arm k. Then if k is the index of a suboptimal arm then Each sub-optimal arm k is played in expectation at most

$$\mathbb{E}[T_k(n)] \leq \frac{16C^2 \ln n}{(1-\epsilon)^2 \triangle_k^2} + A(\epsilon) + N_c + \frac{\pi^2}{3} + 1. \tag{A.13}$$

Proof. When a sub-optimal arm k is pulled at time t we get

$$\left(\frac{\sum_{i=1}^{T_k(t-1)} X_{k,i}^p}{T_k(t-1)}\right)^{\frac{1}{p}} + 2C\sqrt{\frac{\ln t}{T_k(t-1)}} \geq \left(\frac{\sum_{i=1}^{T_{k^*}(t-1)} X_{k^*,i}^p}{T_{k^*}(t-1)}\right)^{\frac{1}{p}} + 2C\sqrt{\frac{\ln t}{T_{k^*}(t-1)}} \tag{A.14}$$

Now, consider the following two inequalities:

- The empirical mean of the optimal arm is not within its confidence interval

$$\left(\frac{\sum_{i=1}^{T_{k^*}(t-1)} X_{k^*,i}^p}{T_{k^*}(t-1)}\right)^{\frac{1}{p}} + 2C\sqrt{\frac{\ln t}{T_{k^*}(t-1)}} \leq \mu_t^* \tag{A.15}$$

- The empirical mean of the arm k is not within its confidence interval

$$\left(\frac{\sum_{i=1}^{T_k(t-1)} X_{k,i}^p}{T_k(t-1)}\right)^{\frac{1}{p}} \geq \mu_{kt} + 2C\sqrt{\frac{\ln t}{T_k(t-1)}} \tag{A.16}$$

If both previous inequalities (A.15), (A.16) do not hold, and if a sub-optimal arm k is pulled, then we deduce that

$$\mu_{kt} + 2C\sqrt{\frac{\ln t}{T_k(t-1)}} \geq \left(\frac{\sum_{i=1}^{T_k(t-1)} X_{k,i}^p}{T_k(t-1)}\right)^{\frac{1}{p}} \text{ see } (A.16) \tag{A.17}$$

and

$$\left(\frac{\sum_{i=1}^{T_k(t-1)} X_{k,i}^p}{T_k(t-1)}\right)^{\frac{1}{p}} \geq \left(\frac{\sum_{i=1}^{T_{k^*}(t-1)} X_{k^*,i}^p}{T_{k^*}(t-1)}\right)^{\frac{1}{p}} + 2C\sqrt{\frac{\ln t}{T_{k^*}(t-1)}} - 2C\sqrt{\frac{\ln t}{T_k(t-1)}} \text{ see } (A.14) \tag{A.18}$$

and

$$\left(\frac{\sum_{i=1}^{T_{k^*}(t-1)} X_{k^*,i}^p}{T_{k^*}(t-1)}\right)^{\frac{1}{p}} + 2C\sqrt{\frac{\ln t}{T_{k^*}(t-1)}} \geq \mu_t^* \text{ see } (A.15). \tag{A.19}$$

So that

$$\mu_{kt} + 4C\sqrt{\frac{\ln t}{T_k(t-1)}} \geq \mu_t^*. \tag{A.20}$$

$\mu_{kt} = \mu_k + \delta_{kt}$, $\mu_t^* = \mu^* + \delta_t^*$ and we have an assumption that $\lim_{t\to\infty} \mu_{kt} = \mu_k$ for any $k \in [1, 2, ...K]$ yields $\lim_{t\to\infty} \delta_{kt} = 0$ Therefore, for any $\epsilon > 0$, we can find an index $A(\epsilon)$ such that for any $t > A(\epsilon)$: $\delta_{kt} \leq \epsilon\triangle_k$ with $\triangle_k = \mu^* - \mu_k$. Which means that

$$4C\sqrt{\frac{\ln t}{T_k(t-1)}} \geq \triangle_k - \delta_{kt} + \delta_t^* \geq (1-\epsilon)\triangle_k \tag{A.21}$$

which implies $T_k(t-1) \leq \frac{16C^2 \ln t}{(1-\epsilon)^2 \triangle_k^2}$.
This says that whenever $T_k(t-1) \geq \frac{16C^2 \ln t}{(1-\epsilon)^2 \triangle_k^2} + A(\epsilon) + N_c$, either arm k is not pulled at time t or one of the two following events (A.15), (A.16) holds. Thus if we define $u = \frac{16C^2 \ln t}{(1-\epsilon)^2 \triangle_k^2} + A(\epsilon) + N_c$, we have

$$\begin{aligned} T_k(n) &\leq u + \sum_{t=u+1}^{n} \mathbf{1}\{I_t = k; T_k(t) \geq u\} \\ &\leq u + \sum_{t=u+1}^{n} \mathbf{1}\{(A.15), \text{ or } (A.16) \text{ holds }\} \end{aligned}$$

We have from (A.11),(A.12)

$$\mathbf{Pr}\left(\left(\frac{\sum_{i=1}^{T_{k^*}(t-1)} X_{k^*,i}^p}{T_{k^*}(t-1)}\right)^{\frac{1}{p}} + 2C\sqrt{\frac{\ln t}{T_{k^*}(t-1)}} \leq \mu_t^*\right) \leq \sum_{s=1}^{t} \frac{1}{t^4} = \frac{1}{t^3} \tag{A.22}$$

and

$$\mathbf{Pr}\left(\left(\frac{\sum_{i=1}^{T_k(t-1)} X_{k,i}^p}{T_k(t-1)}\right)^{\frac{1}{p}} \geq \mu_{kt} + 2C\sqrt{\frac{\ln t}{T_k(t-1)}}\right) \leq \sum_{s=1}^{t} \frac{1}{t^4} = \frac{1}{t^3} \tag{A.23}$$

so that from (A.22), we have

$$\begin{aligned}\mathbb{E}[T_k(n)] &\leq \frac{16C^2 \ln t}{(1-\epsilon)^2 \triangle_k^2} + A(\epsilon) + N_c + \sum_{t=u+1}^{n} \frac{2}{t^{8C^2-1}} = \frac{16C^2 \ln t}{(1-\epsilon)^2 \triangle_k^2} + A(\epsilon) + N_c \\ &+ \sum_{t=u+1}^{n} \frac{2}{t^3} \\ &\leq \frac{16C^2 \ln t}{(1-\epsilon)^2 \triangle_k^2} + A(\epsilon) + N_c + \frac{\pi^2}{3}\end{aligned}$$

□

Based on this result we derive an upper bound for the expectation of power mean in the next theorem as follows.

Theorem 3. *Under the assumptions of Theorem 2,*

$$\left|\mathbb{E}\left[\overline{X}_n(p)\right] - \mu^*\right| \leq |\delta_n^*| + \mathcal{O}\left(\frac{K(C^2 \log n + N_0)}{n}\right)^{\frac{1}{p}}.$$

Proof. In UCT, the value of each node is used for backup as $\overline{X}_n = \sum_{i=1}^{K} \left(\frac{T_i(n)}{n}\right) \overline{X}_{i,T_i(n)}$, and the authors show that

$$\begin{aligned}&\left|\mathbb{E}\left[\overline{X}_n\right] - \mu^*\right| \leq \left|\mathbb{E}\left[\overline{X}_n\right] - \mu_n^*\right| + \left|\mu_n^* - \mu^*\right| \\ &= \left|\delta_n^*\right| + \left|\mathbb{E}\left[\overline{X}_n\right] - \mu_n^*\right| \\ &\leq \left|\delta_n^*\right| + \mathcal{O}\left(\frac{K(C^2 \log n + N_0)}{n}\right)\end{aligned} \tag{A.24}$$

We derive the same results replacing the average with the power mean. First, we have

$$\mathbb{E}\left[\overline{X}_n(p)\right] - \mu_n^* = \mathbb{E}\left[\left(\sum_{i=1}^{K} \frac{T_i(n)}{n} \overline{X}_{i,T_i(n)}^p\right)^{\frac{1}{p}}\right] - \mu_n^*. \tag{A.25}$$

In the proof, we will make use of the following inequalities:

$$0 \leq X_i \leq 1, \tag{A.26}$$

$$x^{\frac{1}{p}} \leq y^{\frac{1}{p}} \text{ when } 0 \leq x \leq y, \tag{A.27}$$

$$(x+y)^m \leq x^m + y^m (0 \leq m \leq 1), \tag{A.28}$$

$$\mathbb{E}[f(X)] \leq f(\mathbb{E}[X]) \text{ (f(X) is concave)}. \tag{A.29}$$

With i^* being the index of the optimal arm, we can derive an upper bound on the difference between the value backup and the true average reward

$$
\begin{aligned}
&\mathbb{E}\left[\left(\sum_{i=1}^{K}\frac{T_i(n)}{n}\overline{X}^p_{i,T_i(n)}\right)^{\frac{1}{p}}\right]-\mu^*_n\\
&\leq \mathbb{E}\left[\left(\left(\sum_{i=1;i\neq i^*}^{K}\frac{T_i(n)}{n}\right)+\overline{X}^p_{i*,T_i*(n)}\right)^{\frac{1}{p}}\right]-\mu^*_n(\text{see }(A.26))\\
&\leq \mathbb{E}\left[\left(\sum_{i=1;i\neq i^*}^{K}\frac{T_i(n)}{n}\right)^{\frac{1}{p}}+\overline{X}_{i*,T_i*(n)}\right]-\mu^*_n(\text{see }(A.28))\\
&= \mathbb{E}\left[\left(\sum_{i=1;i\neq i^*}^{K}\frac{T_i(n)}{n}\right)^{\frac{1}{p}}\right]+\mathbb{E}\left[\overline{X}_{i*,T_i*(n)}\right]-\mu^*_n\\
&= \mathbb{E}\left[\left(\sum_{i=1;i\neq i^*}^{K}\frac{T_i(n)}{n}\right)^{\frac{1}{p}}\right]\\
&\leq \left(\sum_{i=1;i\neq i^*}^{K}\mathbb{E}\left[\frac{T_i(n)}{n}\right]\right)^{\frac{1}{p}}(\text{see }(A.29))\\
&\leq ((K-1)\mathcal{O}\left(\frac{K(C^2\log n+N_0)}{n}\right))^{\frac{1}{p}}(\text{Theorem 2 \& }(A.27))
\end{aligned}
\tag{A.30}
$$

According to Lemma 1, it holds that

$$\mathbb{E}\left[\overline{X}_n(p)\right]\geq\mathbb{E}\left[\overline{X}_n\right]$$

for $p\geq 1$. Because of this, we can reuse the lower bound given by (A.24)

$$-\mathcal{O}\left(\frac{K(C^2\log n+N_0)}{n}\right)\leq\mathbb{E}\left[\overline{X}_n\right]-\mu^*_n,$$

so that

$$
\begin{aligned}
&-\mathcal{O}\left(\frac{K(C^2\log n + N_0)}{n}\right) \leq \mathbb{E}\big[\overline{X}_n\big] - \mu_n^* \\
&\leq \mathbb{E}\big[\overline{X}_n(p)\big] - \mu_n^*. \qquad \text{(A.31)}
\end{aligned}
$$

Combining (A.30) and (A.31) concludes our prove

$$
\left|\mathbb{E}\big[\overline{X}_n(p)\big] - \mu^*\right| \leq |\delta_n^*| + O\left(\frac{K(C^2\log n + N_0)}{n}\right)^{\frac{1}{p}}.
$$

□

The following theorem shows lower bound of choosing all the arms

Theorem 4. ***(Lower Bound)*** *Under the assumptions of Theorem 2, there exists some positive constant ρ such that for all arms k and n, $T_k(n) \geq \lceil \rho \log(n) \rceil$*

Proof. There should exist a constant S that

$$
\left(\frac{\sum_{i=1}^{T_k(t-1)} X_{k,i}^p}{T_k(t-1)}\right)^{\frac{1}{p}} + 2C\sqrt{\frac{\ln t}{T_k(t-1)}} \leq S
$$

for all arm k so

$$
\mu_k + \delta_{kt} + 2C\sqrt{\frac{\log t}{T_k(t-1)}} \leq S
$$

because

$$
\lim_{t\to\infty} \delta_{kt} = 0
$$

so there exists a positive constant ρ that $T_k(t-1) \geq \lceil \rho \log(t) \rceil$ □

The next result shows the estimated optimal payoff concentration around its mean (Theorem 5). In order to prove that, we now reproduce here Lemma 5, 6 [12] that we use for our proof:

Lemma 5. ***Hoeffding-Azuma inequality for Stopped Martingales*** *(Lemma 10 in [12]). Assume that S_t is a centered martingale such that the corresponding martingale difference process is uniformly bounded by C. Then, for any fixed $\epsilon \geq 0$, integers $0 \leq a \leq b$, the following inequalities hold*

$$\mathbf{Pr}(S_N \geq \epsilon N) \leq (b-a+1)\exp\left(\frac{-2a^2\epsilon^2}{C^2}\right) + \mathbf{Pr}(N \notin [a,b]), \tag{A.32}$$

$$\mathbf{Pr}(S_N \leq \epsilon N) \leq (b-a+1)\exp\left(\frac{-2a^2\epsilon^2}{C^2}\right) + \mathbf{Pr}(N \notin [a,b]), \tag{A.33}$$

Lemma 6. *(Lemma 13 in [12]) Let (Z_i), i=1,...,n be a sequence of random variables such that Z_i is conditionally independent of $Z_{i+1}, ..., Z_n$ given $Z_1, ..., Z_{i-1}$. Let define $N_n = \sum_{i=1}^n Z_i$, and let a_n is an upper bound on $\mathbb{E}[N_n]$. Then for all $\triangle \geq 0$, if n is such that $a_n \leq \triangle/2$ then*

$$\mathbf{Pr}(N_n \geq \triangle) \leq \exp(-\triangle^2/(8n)). \tag{A.34}$$

The next lemma is our core result for propagating confidence bounds upward in the tree, and it is used for the prove of Theorem 5 about the concentration of power mean estimator.

Lemma 7. *let Z_i, a_i be as in Lemma 6. Let F_i denotes a filtration over some probability space. Y_i be an F_i-adapted real valued martingale-difference sequence. Let X_i be an i.i.d. sequence with mean μ. We assume that both X_i and Y_i lie in the [0,1] interval. Consider the partial sums*

$$S_n = \left(\frac{\sum_{i=1}^n (1-Z_i)X_i^p + Z_i Y_i^p}{n}\right)^{\frac{1}{p}}. \tag{A.35}$$

Fix an arbitrary $\delta > 0$, and fix $p \geq 1$. Let $\triangle_n = 9\sqrt{\frac{1}{C_p} n \log(2/\delta)}$, and $\triangle = (9/4)^{p-1}\triangle_n$ let

$$R_n = \mathbb{E}\left[\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}}\right] - \mathbb{E}[S_n]. \tag{A.36}$$

Then for n such that $a_n \leq (1/9)\triangle_n$ and $|R_n| \leq (4/9)(\triangle/n)^{\frac{1}{p}}$

$$\mathbf{Pr}(S_n \geq \mathbb{E}[S_n] + (\triangle/n)^{\frac{1}{p}}) \leq \delta \tag{A.37}$$

$$\mathbf{Pr}(S_n \leq \mathbb{E}[S_n] - (\triangle/n)^{\frac{1}{p}}) \leq \delta \tag{A.38}$$

Proof. We have a very fundamental probability inequality:
Consider two events: A, B. If $A \in B$, then $\mathbf{Pr}(A) \leq \mathbf{Pr}(B)$.
Therefore, if we have three random variables X, Y, Z and if we are sure that

$$Y \geq Z, \text{ then } \mathbf{Pr}(X \geq Y) \leq \mathbf{Pr}(X \geq Z) \tag{A.39}$$

We have

$$\begin{aligned}
&\left(\frac{\sum_{i=1}^{n}(1-Z_i)X_i^p + Z_iY_i^p}{n}\right)^{\frac{1}{p}} \\
&= \left(\frac{\sum_{i=1}^{n} X_i^p}{n} + \frac{Z_i(Y_i^p - X_i^p)}{n}\right)^{\frac{1}{p}} \leq \left(\frac{\sum_{i=1}^{n} X_i^p}{n} + \frac{2\sum_{i=1}^{n} Z_i}{n}\right)^{\frac{1}{p}} (X_i, Y_i \in [0,1]) \\
&\leq \left(\frac{\sum_{i=1}^{n} X_i^p}{n}\right)^{\frac{1}{p}} + \left(\frac{2\sum_{i=1}^{n} Z_i}{n}\right)^{\frac{1}{p}} (\text{see } (A.27))
\end{aligned} \tag{A.40}$$

Therefore,

$$T = \mathbf{Pr}\left(S_n \geq \mathbb{E}[S_n] + (\triangle/n)^{\frac{1}{p}}\right) \tag{A.41}$$

$$= \mathbf{Pr}\left(\left(\frac{\sum_{i=1}^{n}(1-Z_i)X_i^p + Z_iY_i^p}{n}\right)^{\frac{1}{p}} \geq \mathbb{E}[\frac{\sum_{i=1}^{n} X_i^p}{n}\Big)^{\frac{1}{p}}] - R_n + (\triangle/n)^{\frac{1}{p}}\right) (\text{see } (A.36)) \tag{A.42}$$

$$\leq \mathbf{Pr}\left(\left(\frac{\sum_{i=1}^{n} X_i^p}{n}\right)^{\frac{1}{p}} + \left(\frac{2\sum_{i=1}^{n} Z_i}{n}\right)^{\frac{1}{p}} \geq \mathbb{E}\left[\left(\frac{\sum_{i=1}^{n} X_i^p}{n}\right)^{\frac{1}{p}}\right] - R_n + (\triangle/n)^{\frac{1}{p}}\right)$$
$$(\text{see } (A.39), (A.40))) \tag{A.43}$$

Using the elementary inequality $I(A + B \geq \triangle/n) \leq I(A \geq \alpha\triangle/n) + I(B \geq (1-\alpha)\triangle/n)$ that holds for any $A, B \geq 0; 0 \leq \alpha \leq 1$, we get

$$\begin{aligned}
T \leq\ & \mathbf{Pr}\left(\left(\frac{\sum_{i=1}^{n} X_i^p}{n}\right)^{\frac{1}{p}} \geq \mathbb{E}\left[\left(\frac{\sum_{i=1}^{n} X_i^p}{n}\right)^{\frac{1}{p}}\right] + 1/9(\triangle/n)^{\frac{1}{p}}\right) \\
&+ \mathbf{Pr}\left(\left(\frac{2\sum_{i=1}^{n} Z_i}{n}\right)^{\frac{1}{p}} \geq 8/9(\triangle/n)^{\frac{1}{p}} - R_n\right)
\end{aligned} \tag{A.44}$$

Define $\mu_p = \mathbb{E}\left[\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}}\right]$, we have

$$\leq \mathbf{Pr}\left(\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}} \geq \mu_p + 1/9(\triangle/n)^{\frac{1}{p}}\right) + \mathbf{Pr}\left(\left(\frac{2\sum_{i=1}^n Z_i}{n}\right)^{\frac{1}{p}} \geq 4/9(\triangle/n)^{\frac{1}{p}}\right)$$

$$(\text{ see } R_n \leq (4/9)(\triangle/n)^{\frac{1}{p}}) \tag{A.45}$$

$$= \mathbf{Pr}\left(\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}} \geq \mu_p + \frac{1}{9}\frac{9}{4}(\frac{4}{9}\triangle_n/n)^{\frac{1}{p}}\right) + \mathbf{Pr}\left(\left(\frac{2\sum_{i=1}^n Z_i}{n}\right)^{\frac{1}{p}} \geq (\frac{(4/9)^p\triangle}{n})^{\frac{1}{p}}\right)$$

$$(\text{ definition of } \triangle) \tag{A.46}$$

$$\leq \mathbf{Pr}\left(\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}} \geq \mu_p + \triangle_n/9n\right) + \mathbf{Pr}\left(\left(\frac{\sum_{i=1}^n Z_i}{n}\right) \geq 2\triangle_n/9n\right)$$

$$(\text{see } (A.27) \text{ and } f(x) = a^x \text{ is decrease when } a < 1) \tag{A.47}$$

The first term is bounded by $\delta/2$ according to (A.8) and the second term is bounded by $\delta/2$ according to Lemma 6 (the condition of Lemma 6 is satisfied because $a_n \leq (1/9)\triangle_n$). This finishes the proof of the first part (A.37). The second part (A.38) can be proved in an analogous manner. □

Theorem 5. *Fix an arbitrary $\delta \leq 0$ and fix $p \geq 1$, let $\triangle_n = (\frac{9}{4})^{p-1}(9\sqrt{\frac{1}{C_p} n \log(2/\delta)})$. Let n_0 be such that*

$$\sqrt{n_0} \leq \mathcal{O}(K(C^2 \log n_0 + N_0(1/2))). \tag{A.48}$$

Then for any $n \geq n_0$, under the assumptions of Theorem 2, the following bounds hold true

$$\mathbf{Pr}(\overline{X}_n(p) \geq \mathbb{E}[\overline{X}_n(p)] + (\triangle_n/n)^{\frac{1}{p}}) \leq \delta \tag{A.49}$$

$$\mathbf{Pr}(\overline{X}_n(p) \leq \mathbb{E}[\overline{X}_n(p)] - (\triangle_n/n)^{\frac{1}{p}}) \leq \delta \tag{A.50}$$

Proof. Let X_t is the payoff sequence of the best arm. Y_t is the payoff at time t. Both X_t, Y_t lies in [0,1] interval, and
$\overline{X}_n(p) = \left(\frac{\sum_{i=1}^n (1-Z_i)X_i^p + Z_i Y_i^p}{n}\right)^{\frac{1}{p}}$ Apply Lemma 6 and remember that $X^{\frac{1}{p}} - Y^{\frac{1}{p}} \leq (X-Y)^{\frac{1}{p}}$

we have

$$
\begin{aligned}
R_n &= \mathbb{E}\left[\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}}\right] - \mathbb{E}\left[\left(\frac{\sum_{i=1}^n (1-Z_i)X_i^p + Z_i Y_i^p}{n}\right)^{\frac{1}{p}}\right]. \\
&= \mathbb{E}\left[\left(\frac{\sum_{i=1}^n X_i^p}{n}\right)^{\frac{1}{p}} - \left(\frac{\sum_{i=1}^n (1-Z_i)X_i^p + Z_i Y_i^p}{n}\right)^{\frac{1}{p}}\right]. \\
&\leq \mathbb{E}\left[\left(\frac{\sum_{i=1}^n X_i^p - \sum_{i=1}^n (1-Z_i)X_i^p - Z_i Y_i^p}{n}\right)^{\frac{1}{p}}\right]. \\
&= \mathbb{E}\left[\left(\frac{\sum_{i=1}^n Z_i(X_i^p - Y_i^p)}{n}\right)^{\frac{1}{p}}\right]. \\
&\leq \mathbb{E}\left[\left(\sum_{i=1}^K \frac{T_i(n)}{n}\right)^{\frac{1}{p}}\right]. \\
&\leq \left(\frac{\sum_{i=1}^K \mathbb{E}[T_i(n)]}{n}\right)^{\frac{1}{p}}. \text{ see Jensen inequality} \\
&= \left((K-1)O\left(\frac{K(C^2 \log n + N_0(1/2))}{n}\right)\right)^{\frac{1}{p}}.
\end{aligned}
$$

So that let n_0 be an index such that if $n \geq n_0$ then $a_n \leq \triangle_n/9$ and $R_n \leq 4/9(\triangle_n/n)^{\frac{1}{p}}$. Such an index exists since $\triangle_n = \mathcal{O}(\sqrt{n})$ and $a_n, R_n = \mathcal{O}((\log n/n)^{\frac{1}{p}})$. Hence, for $n \geq n_0$, the conditions of lemma 6 are satisfied and the desired tail-inequalities hold for $\overline{X_n}(p)$. □

In the next theorem, we show that Power-UCT can ensure the convergence of choosing the best arm at the root node.

Theorem 6. ***(Convergence of Failure Probability)*** *Under the assumptions of Theorem 2, it holds that*

$$\lim_{t\to\infty} \mathbf{Pr}(I_t \neq i^*) = 0 \tag{A.51}$$

Proof. We show that Power-UCT can smoothly adapt to UCT's prove. Let i be the index of a suboptimal arm and let $p_{it} = \mathbf{Pr}(\overline{X}_{i,T_i(t)}(p) \geq \overline{X}^*_{T^*(t)}(p))$ from above. Clearly, $\mathbf{Pr}(I_t \neq i*) \leq \sum_{i\neq i*} p_{it}$. Hence, it suffices to show that $p_{it} \leq \epsilon/K$ holds for all suboptimal

arms for t sufficiently large.
Clearly, if $\overline{X}_{i,T_i(t)}(p) \leq \mu_i + \triangle_i/2$ and $\overline{X}^*_{T^*(t)}(p) \geq \mu^* - \triangle_i/2$ then $\overline{X}_{i,T_i(t)}(p) < \overline{X}^*_{T^*(t)}(p)$. Hence,

$$p_t \leq \mathbf{Pr}(\overline{X}_{i,T_i(t)}(p) \leq \mu_i + \triangle_i/2) + \mathbf{Pr}(\overline{X}^*_{T^*(t)}(p) \geq \mu^* - \triangle_i/2)$$

The first probability can be expected to be converging much slower since $T_i(t)$ converges slowly. Hence, we bound it first.
In fact,

$$\mathbf{Pr}(\overline{X}_{i,T_i(t)}(p) \leq \mu_i + \triangle_i/2) \leq \mathbf{Pr}(\overline{X}_{i,T_i(t)}(p) \leq \overline{\mu}_{i,T_i(t)} - |\delta_{i,T_i(t)}| + \triangle_i/2).$$

Without the loss of generality, we may assume that $|\delta_{i,T_i(t)}| \leq \triangle_i/4$. Therefore

$$\mathbf{Pr}(\overline{X}_{i,T_i(t)}(p) \leq \mu_i + \triangle_i/2) \leq \mathbf{Pr}(\overline{X}_{i,T_i(t)}(p) \leq \overline{\mu}_{i,T_i(t)} + \triangle_i/4).$$

Now let a be an index such that if $t \geq a$ then $(t+1)\mathbf{Pr}(\overline{X}_{i,T_i(t)}(p) \leq \overline{\mu}_{i,T_i(t)} + \triangle_i/4) \leq \epsilon/(2K)$. Such an index exist by our assumptions on the concentration properties of the average payoffs. Then, for $t \geq a$

$$\begin{aligned}\mathbf{Pr}(\overline{X}_{i,T_i(t)}(p) \leq \overline{\mu}_{i,T_i(t)} + \triangle_i/4) \leq \mathbf{Pr}(\overline{X}_{i,T_i(t)}(p) \leq \overline{\mu}_{i,T_i(t)} + \triangle_i/4, T_i(t) \geq a) \\ + \mathbf{Pr}(T_i(t) \leq a)\end{aligned}$$

Since the lower-bound on $T_i(t)$ grows to infinity as $t \to \infty$, the second term becomes zero when t is sufficiently large. The first term is bounded using the method of Lemma 5. By choosing $b = 2a$, we get

$$\begin{aligned}\mathbf{Pr}(\overline{X}_{i,T_i(t)}(p) \leq \overline{\mu}_{i,T_i(t)} + \triangle_i/4, T_i(t) \geq a) \leq (a+1)\mathbf{Pr}(\overline{X}_{i,a}(p) \leq \overline{\mu}_{i,a} + \triangle_i/4, T_i(t) \geq a) \\ + \mathbf{Pr}(T_i(t) \geq 2b) \leq \epsilon/(2K),\end{aligned}$$

where we have assumed that t is large enough so that $P(T_i(t) \geq 2b) = 0$.
Bounding $\mathbf{Pr}(\overline{X}^*_{T^*(t)}(p) \geq \mu^* - \triangle_i/2)$ by $\epsilon/(2K)$ can be done in an analogous manner. Collecting the bound yields that $p_{it} \leq \epsilon/K$ for t sufficiently large which complete the prove. □

Now is our result to show the bias of expected payoff $\overline{X_n}(p)$

Theorem 7. *Consider algorithm Power-UCT running on a game tree of depth D, branching factor K with stochastic payoff at the leaves. Assume that the payoffs lie in the interval [0,1]. Then the bias of the estimated expected payoff, $\overline{X_n}$, is $\mathcal{O}(KD(\log(n)/n)^{\frac{1}{p}} + K^D(1/n)^{\frac{1}{p}})$. Further, the failure probability at the root convergences to zero as the number of samples grows to infinity.*

Proof. The proof is done by induction on D. When $D = 1$, Power-UCT becomes UCB1 problem and as the result of Hoeffding's inequality, the convergence is guaranteed directly from Theorem 1, Theorem 3 and Theorem 6.

Now we assume that the result holds up to depth $D - 1$ and consider the tree of Depth D. Running Power-UCT on root node is equivalence as UCB1 on non-stationary bandit settings. The error bound of running Power-UCT for the whole tree is the sum of payoff at root node with payoff starting from any node i after the first action chosen from root node until the end. This payoff by induction at depth $(D - 1)$ is

$$\mathcal{O}(K(D-1)(\log(n)/n)^{\frac{1}{p}} + K^{D-1}(1/n)^{\frac{1}{p}}).$$

According to the Theorem 3, the payoff at the root node is

$$|\delta_n^*| + \mathcal{O}\left(\frac{K(\log n + N_0)}{n}\right)^{\frac{1}{p}}.$$

The payoff of the whole tree with depth D

$$\begin{aligned}
&|\delta_n^*| + \mathcal{O}\left(\frac{K(\log n + N_0)}{n}\right)^{\frac{1}{p}} \\
&= \mathcal{O}(K(D-1)(\log(n)/n)^{\frac{1}{p}} + K^{D-1}(1/n)^{\frac{1}{p}}) \\
&+ \mathcal{O}\left(\frac{K(\log n + N_0)}{n}\right)^{\frac{1}{p}} \\
&\leq \mathcal{O}(K(D-1)(\log(n)/n)^{\frac{1}{p}} + K^{D-1}(1/n)^{\frac{1}{p}}) \\
&+ \mathcal{O}\left(K\left(\frac{\log n}{n}\right)^{\frac{1}{p}} + KN_0\left(\frac{1}{n}\right)^{\frac{1}{p}}\right) \\
&= \mathcal{O}(KD(\log(n)/n)^{\frac{1}{p}} + K^D(1/n)^{\frac{1}{p}})
\end{aligned}$$

with $N_0 = \mathcal{O}((K-1)K^{D-1})$, which completes our proof of the convergence of Power-UCT. Since by our induction hypothesis this holds for all nodes at a distance of one node from the root, the proof is finished by observing that Theorem 3 and Theorem 5 do indeed ensure that the drift conditions are satisfied. Interestingly, the proof guarantees the convergence for any finite value of p. □

A.2. Convex Regularization in Monte-Carlo Tree Search

In this section, we describe how to derive the theoretical results presented for the Convex Regularization in Monte-Carlo Tree Search.

First, the exponential convergence rate of the estimated value function to the conjugate regularized value function at the root node (Theorem 1) is derived based on induction with respect to the depth D of the tree. When $D = 1$, we derive the concentration of the average reward at the leaf node with respect to the ∞-norm (as shown in Lemma 1) based on the result from Theorem 2.19 in [156], and the induction is done over the tree by additionally exploiting the contraction property of the convex regularized value function. Second, based on Theorem 1, we prove the exponential convergence rate of choosing the best action at the root node (Theorem 2). Third, the pseudo-regret analysis of E3W is derived based on the Bregman divergence properties and the contraction properties of the Legendre-Fenchel transform (Proposition 1). Finally, the bias error of estimated value at the root node is derived based on results of Theorem 1, and the boundedness property of the Legendre-Fenchel transform (Proposition 1).

Let $\hat{r}$ and r be respectively the average and the the expected reward at the leaf node, and the reward distribution at the leaf node be σ^2-sub-Gaussian.

Lemma 1. *For the stochastic bandit problem E3W guarantees that, for $t \geq 4$,*

$$\mathbf{Pr}\Big(\parallel r - \hat{r}_t \parallel_\infty \geq \frac{2\sigma}{\log(2+t)} \Big) \leq 4|\mathcal{A}| \exp\Big(- \frac{t}{(\log(2+t))^3} \Big).$$

Proof. Let us define $N_t(a)$ as the number of times action a have been chosen until time t, and $\hat{N}_t(a) = \sum_{s=1}^{t} \pi_s(a)$, where $\pi_s(a)$ is the E3W policy at time step s. By choosing

$\lambda_s = \frac{|\mathcal{A}|}{\log(1+s)}$, it follows that for all a and $t \geq 4$,

$$\hat{N}_t(a) = \sum_{s=1}^{t} \pi_s(a) \geq \sum_{s=1}^{t} \frac{1}{\log(1+s)} \geq \sum_{s=1}^{t} \frac{1}{\log(1+s)} - \frac{s/(s+1)}{(\log(1+s))^2}$$
$$\geq \int_1^{1+t} \frac{1}{\log(1+s)} - \frac{s/(s+1)}{(\log(1+s))^2} ds = \frac{1+t}{\log(2+t)} - \frac{1}{\log 2} \geq \frac{t}{2\log(2+t)}.$$

From Theorem 2.19 in [156], we have the following concentration inequality

$$\mathbf{Pr}(|N_t(a) - \hat{N}_t(a)| > \epsilon) \leq 2\exp\{-\frac{\epsilon^2}{2\sum_{s=1}^{t}\sigma_s^2}\} \leq 2\exp\{-\frac{2\epsilon^2}{t}\},$$

where $\sigma_s^2 \leq 1/4$ is the variance of a Bernoulli distribution with $p = \pi_s(k)$ at time step s. We define the event

$$E_\epsilon = \{\forall a \in \mathcal{A}, |\hat{N}_t(a) - N_t(a)| \leq \epsilon\},$$

and consequently

$$\mathbf{Pr}(|\hat{N}_t(a) - N_t(a)| \geq \epsilon) \leq 2|\mathcal{A}|\exp(-\frac{2\epsilon^2}{t}). \tag{A.52}$$

Conditioned on the event E_ϵ, for $\epsilon = \frac{t}{4\log(2+t)}$, we have $N_t(a) \geq \frac{t}{4\log(2+t)}$. For any action a by the definition of sub-gaussian,

$$\mathbf{Pr}\left(|r(a) - \hat{r}_t(a)| > \sqrt{\frac{8\sigma^2\log(\frac{2}{\delta})\log(2+t)}{t}}\right) \leq \mathbf{Pr}\left(|r(a) - \hat{r}_t(a)| > \sqrt{\frac{2\sigma^2\log(\frac{2}{\delta})}{N_t(a)}}\right) \leq \delta$$

by choosing a δ satisfying $\log(\frac{2}{\delta}) = \frac{1}{(\log(2+t))^3}$, we have

$$\mathbf{Pr}\left(|r(a) - \hat{r}_t(a)| > \sqrt{\frac{2\sigma^2\log(\frac{2}{\delta})}{N_t(a)}}\right) \leq 2\exp\left(-\frac{1}{(\log(2+t))^3}\right).$$

Therefore, for $t \geq 2$

$$\mathbf{Pr}\left(\parallel r - \hat{r}_t \parallel_\infty > \frac{2\sigma}{\log(2+t)}\right) \leq \mathbf{Pr}\left(\parallel r - \hat{r}_t \parallel_\infty > \frac{2\sigma}{\log(2+t)} \middle| E_\epsilon\right) + \mathbf{Pr}(E_\epsilon^C)$$
$$\leq \sum_k \left(\mathbf{Pr}\left(|r(a) - \hat{r}_t(a)| > \frac{2\sigma}{\log(2+t)}\right) + \mathbf{Pr}(E_\epsilon^C) \leq 2|\mathcal{A}|\exp\left(-\frac{1}{(\log(2+t))^3}\right)\right)$$
$$+ 2|\mathcal{A}|\exp\left(-\frac{t}{(\log(2+t))^3}\right) = 4|\mathcal{A}|\exp\left(-\frac{t}{(\log(2+t))^3}\right).$$

□

Lemma 2. *Given two policies $\pi^{(1)} = \nabla\Omega^*(r^{(1)})$ and $\pi^{(2)} = \nabla\Omega^*(r^{(2)}), \exists L$, such that*

$$\| \pi^{(1)} - \pi^{(2)} \|_p \leq L \| r^{(1)} - r^{(2)} \|_p .$$

Proof. This comes directly from the fact that $\pi = \nabla\Omega^*(r)$ is Lipschitz continuous with ℓ^p-norm. Note that p has different values according to the choice of regularizer. Refer to [85] for a discussion of each norm using maximum entropy and Tsallis entropy regularizer. Relative entropy shares the same properties with maximum Entropy. □

Lemma 3. *Consider the E3W policy applied to a tree. At any node s of the tree with depth d, Let us define $N_t^*(s,a) = \pi^*(a|s).t$, and $\hat{N}_t(s,a) = \sum_{s=1}^{t} \pi_s(a|s)$, where $\pi_k(a|s)$ is the policy at time step k. There exists some C and $\hat{C}$ such that*

$$\mathbf{Pr}\big(|\hat{N}_t(s,a) - N_t^*(s,a)| > \frac{Ct}{\log t}\big) \leq \hat{C}|\mathcal{A}|t \exp\{-\frac{t}{(\log t)^3}\}.$$

Proof. We denote the following event,

$$E_{r_k} = \{\| r(s',\cdot) - \hat{r}_k(s',\cdot) \|_\infty < \frac{2\sigma}{\log(2+k)}\}.$$

Thus, conditioned on the event $\bigcap_{i=1}^{t} E_{r_t}$ and for $t \geq 4$, we bound $|\hat{N}_t(s,a) - N_t^*(s,a)|$ as

$$
\begin{aligned}
|\hat{N}_t(s,a) - N_t^*(s,a)| &\leq \sum_{k=1}^{t} |\hat{\pi}_k(a|s) - \pi^*(a|s)| + \sum_{k=1}^{t} \lambda_k \\
&\leq \sum_{k=1}^{t} \parallel \hat{\pi}_k(\cdot|s) - \pi^*(\cdot|s) \parallel_\infty + \sum_{k=1}^{t} \lambda_k \\
&\leq \sum_{k=1}^{t} \parallel \hat{\pi}_k(\cdot|s) - \pi^*(\cdot|s) \parallel_p + \sum_{k=1}^{t} \lambda_k \\
&\leq L \sum_{k=1}^{t} \parallel \hat{Q}_k(s',\cdot) - Q(s',\cdot) \parallel_p + \sum_{k=1}^{t} \lambda_k (\text{Lemma 2}) \\
&\leq L|\mathcal{A}|^{\frac{1}{p}} \sum_{k=1}^{t} \parallel \hat{Q}_k(s',\cdot) - Q(s',\cdot) \parallel_\infty + \sum_{k=1}^{t} \lambda_k (\text{ Property of } p\text{-norm}) \\
&\leq L|\mathcal{A}|^{\frac{1}{p}} \gamma^d \sum_{k=1}^{t} \parallel \hat{r}_k(s'',\cdot) - r(s'',\cdot) \parallel_\infty + \sum_{k=1}^{t} \lambda_k (\text{Contraction 3.3.1}) \\
&\leq L|\mathcal{A}|^{\frac{1}{p}} \gamma^d \sum_{k=1}^{t} \frac{2\sigma}{\log(2+k)} + \sum_{k=1}^{t} \lambda_k \\
&\leq L|\mathcal{A}|^{\frac{1}{p}} \gamma^d \int_{k=0}^{t} \frac{2\sigma}{\log(2+k)} dk + \int_{k=0}^{t} \frac{|\mathcal{A}|}{\log(1+k)} dk \\
&\leq \frac{Ct}{\log t}.
\end{aligned}
$$

for some constant C depending on $|\mathcal{A}|, p, d, \sigma, L$, and γ . Finally,

$$
\begin{aligned}
\mathbf{Pr}(|\hat{N}_t(s,a) - N_t^*(s,a)| \geq \frac{Ct}{\log t}) &\leq \sum_{i=1}^{t} \mathbf{Pr}(E_{r_t}^c) = \sum_{i=1}^{t} 4|\mathcal{A}| \exp(-\frac{t}{(\log(2+t))^3}) \\
&\leq 4|\mathcal{A}| t \exp(-\frac{t}{(\log(2+t))^3}) \\
&= O(t \exp(-\frac{t}{(\log(t))^3})).
\end{aligned}
$$

□

Lemma 4. *Consider the E3W policy applied to a tree. At any node s of the tree, Let us define $N_t^*(s,a) = \pi^*(a|s).t$, and $N_t(s,a)$ as the number of times action a have been chosen until*

time step t. There exists some C and $\hat{C}$ such that

$$\mathbf{Pr}\big(|N_t(s,a) - N_t^*(s,a)| > \frac{Ct}{\log t}\big) \leq \hat{C}t \exp\{-\frac{t}{(\log t)^3}\}.$$

Proof. Based on the result from Lemma 3, we have

$$\begin{aligned}
&\mathbf{Pr}\big(|N_t(s,a) - N_t^*(s,a)| > (1+C)\frac{t}{\log t}\big) \leq Ct \exp\{-\frac{t}{(\log t)^3}\} \\
&\leq \mathbf{Pr}\big(|\hat{N}_t(s,a) - N_t^*(s,a)| > \frac{Ct}{\log t}\big) + \mathbf{Pr}\big(|N_t(s,a) - \hat{N}_t(s,a)| > \frac{t}{\log t}\big) \\
&\leq 4|\mathcal{A}|t \exp\{-\frac{t}{(\log(2+t))^3}\} + 2|\mathcal{A}| \exp\{-\frac{t}{(\log(2+t))^2}\} (\text{Lemma 3 and } (A.52)) \\
&\leq O(t \exp(-\frac{t}{(\log t)^3})).
\end{aligned}$$

□

Theorem 8. *At the root node s of the tree, defining $N(s)$ as the number of visitations and $V_{\Omega^*}(s)$ as the estimated value at node s, for $\epsilon > 0$, we have*

$$\mathbf{Pr}(|V_\Omega(s) - V_\Omega^*(s)| > \epsilon) \leq C \exp\{-\frac{N(s)\epsilon}{\hat{C}(\log(2+N(s)))^2}\}.$$

Proof. We prove this concentration inequality by induction. When the depth of the tree is $D = 1$, from Proposition 1, we get

$$|V_\Omega(s) - V_\Omega^*(s)| = \| \Omega^*(Q_\Omega(s,\cdot)) - \Omega^*(Q_\Omega^*(s,\cdot)) \|_\infty \leq \gamma \| \hat{r} - r^* \|_\infty \text{ (Contraction)}$$

where $\hat{r}$ is the average rewards and r^* is the mean reward. So that

$$\mathbf{Pr}(|V_\Omega(s) - V_\Omega^*(s)| > \epsilon) \leq \mathbf{Pr}(\gamma \| \hat{r} - r^* \|_\infty > \epsilon).$$

From Lemma 1, with $\epsilon = \frac{2\sigma\gamma}{\log(2+N(s))}$, we have

$$\begin{aligned}
\mathbf{Pr}(|V_\Omega(s) - V_\Omega^*(s)| > \epsilon) &\leq \mathbf{Pr}(\gamma \| \hat{r} - r^* \|_\infty > \epsilon) \leq 4|\mathcal{A}| \exp\{-\frac{N(s)\epsilon}{2\sigma\gamma(\log(2+N(s)))^2}\} \\
&= C \exp\{-\frac{N(s)\epsilon}{\hat{C}(\log(2+N(s)))^2}\}.
\end{aligned}$$

Let assume we have the concentration bound at the depth $D-1$, Let us define $V_\Omega(s_a) = Q_\Omega(s,a)$, where s_a is the state reached taking action a from state s. then at depth $D-1$

$$\mathbf{Pr}(|V_\Omega(s_a) - V^*_\Omega(s_a)| > \epsilon) \leq C \exp\{-\frac{N(s_a)\epsilon}{\hat{C}(\log(2+N(s_a)))^2}\}. \tag{A.53}$$

Now at the depth D, because of the Contraction Property, we have

$$\begin{aligned}|V_\Omega(s) - V^*_\Omega(s)| &\leq \gamma \parallel Q_\Omega(s,\cdot) - Q^*_\Omega(s,\cdot) \parallel_\infty \\ &= \gamma |Q_\Omega(s,a) - Q^*_\Omega(s,a)|.\end{aligned}$$

So that

$$\begin{aligned}\mathbf{Pr}(|V_\Omega(s) - V^*_\Omega(s)| > \epsilon) &\leq \mathbf{Pr}(\gamma \parallel Q_\Omega(s,a) - Q^*_\Omega(s,a) \parallel > \epsilon) \\ &\leq C_a \exp\{-\frac{N(s_a)\epsilon}{\hat{C}_a(\log(2+N(s_a)))^2}\} \\ &\leq C_a \exp\{-\frac{N(s_a)\epsilon}{\hat{C}_a(\log(2+N(s)))^2}\}.\end{aligned}$$

From (A.53), we can have $\lim_{t\to\infty} N(s_a) = \infty$ because if $\exists L, N(s_a) < L$, we can find $\epsilon > 0$ for which (A.53) is not satisfied. From Lemma 4, when $N(s)$ is large enough, we have $N(s_a) \to \pi^*(a|s)N(s)$ (for example $N(s_a) > \frac{1}{2}\pi^*(a|s)N(s)$), that means we can find C and $\hat{C}$ that satisfy

$$\mathbf{Pr}(|V_\Omega(s) - V^*_\Omega(s)| > \epsilon) \leq C \exp\{-\frac{N(s)\epsilon}{\hat{C}(\log(2+N(s)))^2}\}.$$

□

Lemma 5. *At any node s of the tree, $N(s)$ is the number of visitations. We define the event*

$$E_s = \{\forall a \in \mathcal{A}, |N(s,a) - N^*(s,a)| < \frac{N^*(s,a)}{2}\} \text{ where } N^*(s,a) = \pi^*(a|s)N(s),$$

where $\epsilon > 0$ and $V_{\Omega^*}(s)$ is the estimated value at node s. We have

$$\mathbf{Pr}(|V_\Omega(s) - V^*_\Omega(s)| > \epsilon | E_s) \leq C \exp\{-\frac{N(s)\epsilon}{\hat{C}(\log(2+N(s)))^2}\}.$$

Proof. The proof is the same as in Theorem 2. We prove the concentration inequality by induction. When the depth of the tree is $D = 1$, from Proposition 1, we get

$$|V_\Omega(s) - V^*_\Omega(s)| = \| \Omega^*(Q_\Omega(s,\cdot)) - \Omega^*(Q^*_\Omega(s,\cdot)) \| \leq \gamma \| \hat{r} - r^* \|_\infty \text{ (Contraction Property)}$$

where $\hat{r}$ is the average rewards and r^* is the mean rewards. So that

$$\mathbf{Pr}(|V_\Omega(s) - V^*_\Omega(s)| > \epsilon) \leq \mathbf{Pr}(\gamma \| \hat{r} - r^* \|_\infty > \epsilon).$$

From Lemma 1, with $\epsilon = \frac{2\sigma\gamma}{\log(2+N(s))}$ and given E_s, we have

$$\begin{aligned}\mathbf{Pr}(|V_\Omega(s) - V^*_\Omega(s)| > \epsilon) &\leq \mathbf{Pr}(\gamma \| \hat{r} - r^* \|_\infty > \epsilon) \leq 4|\mathcal{A}| \exp\{-\frac{N(s)\epsilon}{2\sigma\gamma(\log(2+N(s)))^2}\} \\ &= C \exp\{-\frac{N(s)\epsilon}{\hat{C}(\log(2+N(s)))^2}\}.\end{aligned}$$

Let assume we have the concentration bound at the depth $D-1$, Let us define $V_\Omega(s_a) = Q_\Omega(s,a)$, where s_a is the state reached taking action a from state s, then at depth $D-1$

$$\mathbf{Pr}(|V_\Omega(s_a) - V^*_\Omega(s_a)| > \epsilon) \leq C \exp\{-\frac{N(s_a)\epsilon}{\hat{C}(\log(2+N(s_a)))^2}\}.$$

Now at depth D, because of the Contraction Property and given E_s, we have

$$\begin{aligned}|V_\Omega(s) - V^*_\Omega(s)| &\leq \gamma \| Q_\Omega(s,\cdot) - Q^*_\Omega(s,\cdot) \|_\infty \\ &= \gamma |Q_\Omega(s,a) - Q^*_\Omega(s,a)| (\exists a, \text{ satisfied}).\end{aligned}$$

So that

$$\begin{aligned}\mathbf{Pr}(|V_\Omega(s) - V^*_\Omega(s)| > \epsilon) &\leq \mathbf{Pr}(\gamma \| Q_\Omega(s,a) - Q^*_\Omega(s,a) \| > \epsilon) \\ &\leq C_a \exp\{-\frac{N(s_a)\epsilon}{\hat{C}_a(\log(2+N(s_a)))^2}\} \\ &\leq C_a \exp\{-\frac{N(s_a)\epsilon}{\hat{C}_a(\log(2+N(s)))^2}\} \\ &\leq C \exp\{-\frac{N(s)\epsilon}{\hat{C}(\log(2+N(s)))^2}\} (\text{because of } E_s)\end{aligned}$$

. □

Theorem 9. *Let a_t be the action returned by algorithm E3W at iteration t. Then for t large enough, with some constants $C, \hat{C}$,*

$$\mathbf{Pr}(a_t \neq a^*) \leq Ct \exp\{-\frac{t}{\hat{C}\sigma(\log(t))^3}\}.$$

Proof. Let us define event E_s as in Lemma 5. Let a^* be the action with largest value estimate at the root node state s. The probability that E3W selects a sub-optimal arm at s is

$$\mathbf{Pr}(a_t \neq a^*) \leq \sum_a \mathbf{Pr}(V_\Omega(s_a)) > V_\Omega(s_{a^*})|E_s) + \mathbf{Pr}(E_s^c)$$
$$= \sum_a \mathbf{Pr}((V_\Omega(s_a) - V_\Omega^*(s_a)) - (V_\Omega(s_{a^*}) - V_\Omega^*(s_{a^*})) \geq V_\Omega^*(s_{a^*}) - V_\Omega^*(s_a)|E_s) + \mathbf{Pr}(E_s^c).$$

Let us define $\Delta = V_\Omega^*(s_{a^*}) - V_\Omega^*(s_a)$, therefore for $\Delta > 0$, we have

$$\mathbf{Pr}(a_t \neq a^*) \leq \sum_a \mathbf{Pr}((V_\Omega(s_a) - V_\Omega^*(s_a)) - (V_\Omega(s_{a^*}) - V_\Omega^*(s_{a^*})) \geq \Delta|E_s) + +\mathbf{Pr}(E_s^c)$$
$$\leq \sum_a \mathbf{Pr}(|V_\Omega(s_a) - V_\Omega^*(s_a)| \geq \alpha\Delta|E_s) + \mathbf{Pr}(|V_\Omega(s_{a^*}) - V_\Omega^*(s_{a^*})| \geq \beta\Delta|E_s) + \mathbf{Pr}(E_s^c)$$
$$\leq \sum_a C_a \exp\{-\frac{N(s)(\alpha\Delta)}{\hat{C}_a(\log(2+N(s)))^2}\} + C_{a^*} \exp\{-\frac{N(s)(\beta\Delta)}{\hat{C}_{a^*}(\log(2+N(s)))^2}\} + \mathbf{Pr}(E_s^c),$$

where $\alpha + \beta = 1$, $\alpha > 0$, $\beta > 0$, and $N(s)$ is the number of visitations the root node s. Let us define $\frac{1}{\hat{C}} = \min\{\frac{(\alpha\Delta)}{C_a}, \frac{(\beta\Delta)}{C_{a^*}}\}$, and $C = \frac{1}{|\mathcal{A}|}\max\{C_a, C_{a^*}\}$ we have

$$\mathbf{Pr}(a \neq a^*) \leq C \exp\{-\frac{t}{\hat{C}\sigma(\log(2+t))^2}\} + \mathbf{Pr}(E_s^c).$$

From Lemma 4, $\exists C^{'}, \hat{C^{'}}$ for which

$$\mathbf{Pr}(E_s^c) \leq C^{'} t \exp\{-\frac{t}{\hat{C^{'}}(\log(t))^3}\},$$

so that

$$\mathbf{Pr}(a \neq a^*) \leq O(t \exp\{-\frac{t}{(\log(t))^3}\}).$$

□

Theorem 10. *Consider an E3W policy applied to the tree. Let define $\mathcal{D}_{\Omega^*}(x, y) = \Omega^*(x) - \Omega^*(y) - \nabla\Omega^*(y)(x - y)$ as the Bregman divergence between x and y, The expected pseudo regret R_n satisfies*

$$\mathbb{E}[R_n] \leq -\tau\Omega(\hat{\pi}) + \sum_{t=1}^{n} \mathcal{D}_{\Omega^*}(\hat{V}_t(\cdot) + V(\cdot), \hat{V}_t(\cdot)) + \mathcal{O}(\frac{n}{\log n}).$$

Proof. Without loss of generality, we can assume that $V_i \in [-1, 0], \forall i \in [1, |A|]$. as the definition of regret, we have

$$\mathbb{E}[R_n] = nV^* - \sum_{t=1}^{n} \langle \hat{\pi}_t(\cdot), V(\cdot) \rangle \leq \hat{V}_1(0) - \sum_{t=1}^{n} \langle \hat{\pi}_t(\cdot), V(\cdot) \rangle \leq -\tau\Omega(\hat{\pi}) - \sum_{t=1}^{n} \langle \hat{\pi}_t(\cdot), V(\cdot) \rangle .$$

By the definition of the tree policy, we can obtain

$$\begin{aligned}
-\sum_{t=1}^{n} \langle \hat{\pi}_t(\cdot), V(\cdot) \rangle &= -\sum_{t=1}^{n} \left\langle (1 - \lambda_t)\nabla\Omega^*(\hat{V}_t(\cdot)), V(\cdot) \right\rangle - \sum_{t=1}^{n} \left\langle \frac{\lambda_t(\cdot)}{|A|}, V(\cdot) \right\rangle \\
&= -\sum_{t=1}^{n} \left\langle (1 - \lambda_t)\nabla\Omega^*(\hat{V}_t(\cdot)), V(\cdot) \right\rangle - \sum_{t=1}^{n} \left\langle \frac{\lambda_t(\cdot)}{|A|}, V(\cdot) \right\rangle \\
&\leq -\sum_{t=1}^{n} \left\langle \nabla\Omega^*(\hat{V}_t(\cdot)), V(\cdot) \right\rangle - \sum_{t=1}^{n} \left\langle \frac{\lambda_t(\cdot)}{|A|}, V(\cdot) \right\rangle .
\end{aligned}$$

with

$$
\begin{aligned}
-\sum_{t=1}^{n}\left\langle \nabla\Omega^*(\hat{V}_t(\cdot)), V(\cdot)\right\rangle &= \sum_{t=1}^{n}\Omega^*(\hat{V}_t(\cdot)+V(\cdot)) - \sum_{t=1}^{n}\Omega^*(\hat{V}_t(\cdot)) - \sum_{t=1}^{n}\left\langle \nabla\Omega^*(\hat{V}_t(\cdot)), V(\cdot)\right\rangle \\
&- (\sum_{t=1}^{n}\Omega^*(\hat{V}_t(\cdot)+V(\cdot)) - \sum_{t=1}^{n}\Omega^*(\hat{V}_t(\cdot))) \\
&= \sum_{t=1}^{n}\mathcal{D}_{\Omega^*}(\hat{V}_t(\cdot)+V(\cdot), \hat{V}_t(\cdot)) \\
&- (\sum_{t=1}^{n}\Omega^*(\hat{V}_t(\cdot)+V(\cdot)) - \sum_{t=1}^{n}\Omega^*(\hat{V}_t(\cdot))) \\
&\leq \sum_{t=1}^{n}\mathcal{D}_{\Omega^*}(\hat{V}_t(\cdot)+V(\cdot), \hat{V}_t(\cdot)) + n \parallel V(\cdot) \parallel_\infty \\
&(\text{Contraction property, Proposition 1}) \\
&\leq \sum_{t=1}^{n}\mathcal{D}_{\Omega^*}(\hat{V}_t(\cdot)+V(\cdot), \hat{V}_t(\cdot)).(\text{ because } V_i \leq 0)
\end{aligned}
$$

And

$$
-\sum_{t=1}^{n}\left\langle \frac{\lambda_t(\cdot)}{|A|}, V(\cdot)\right\rangle \leq \mathcal{O}(\frac{n}{\log n}), (\text{Because} \sum_{k=1}^{n}\frac{1}{\log(k+1)} \to \mathcal{O}(\frac{n}{\log n}))
$$

So that

$$
\mathbb{E}[R_n] \leq -\tau\Omega(\hat{\pi}) + \sum_{t=1}^{n}\mathcal{D}_{\Omega^*}(\hat{V}_t(\cdot)+V(\cdot), \hat{V}_t(\cdot)) + \mathcal{O}(\frac{n}{\log n}).
$$

We consider the generalized Tsallis Entropy $\Omega(\pi) = \mathcal{S}_\alpha(\pi) = \frac{1}{1-\alpha}(1 - \sum_i \pi^\alpha(a_i|s))$. According to [157], when $\alpha \in (0,1)$

$$
\begin{aligned}
\mathcal{D}_{\Omega^*}(\hat{V}_t(\cdot)+V(\cdot), \hat{V}_t(\cdot)) &\leq (\tau\alpha)^{-1}|\mathcal{A}|^\alpha \\
-\Omega(\hat{\pi}_n) &\leq \frac{1}{1-\alpha}(|\mathcal{A}|^{1-\alpha} - 1).
\end{aligned}
$$

Then, for the generalized Tsallis Entropy, when $\alpha \in (0,1)$, the regret is

$$
\mathbb{E}[R_n] \leq \frac{\tau}{1-\alpha}(|\mathcal{A}|^{1-\alpha} - 1) + n(\tau\alpha)^{-1}|\mathcal{A}|^\alpha + \mathcal{O}(\frac{n}{\log n}),
$$

when $\alpha = 2$, which is the Tsallis entropy case we consider, according to [158], By Taylor's theorem $\exists z \in \text{conv}(\hat{V}_t, \hat{V}_t + V)$, we have

$$\mathcal{D}_{\Omega^*}(\hat{V}_t(\cdot) + V(\cdot), \hat{V}_t(\cdot)) \leq \frac{1}{2}\left\langle V(\cdot), \nabla^2\Omega^*(z)V(\cdot)\right\rangle \leq \frac{|\mathcal{K}|}{2}.$$

So that when $\alpha = 2$, we have

$$\mathbb{E}[R_n] \leq \tau(\frac{|\mathcal{A}| - 1}{|\mathcal{A}|}) + \frac{n|\mathcal{K}|}{2} + \mathcal{O}(\frac{n}{\log n}).$$

when $\alpha = 1$, which is the maximum entropy case in our work, we derive.

$$\mathbb{E}[R_n] \leq \tau(\log|\mathcal{A}|) + \frac{n|\mathcal{A}|}{\tau} + \mathcal{O}(\frac{n}{\log n})$$

Finally, when the convex regularizer is relative entropy, One can simply write $KL(\pi_t||\pi_{t-1}) = -H(\pi_t) - \mathbb{E}_{\pi_t}\log\pi_{t-1}$, let $m = \min_a \pi_{t-1}(a|s)$, we have

$$\mathbb{E}[R_n] \leq \tau(\log|\mathcal{A}| - \frac{1}{m}) + \frac{n|\mathcal{A}|}{\tau} + \mathcal{O}(\frac{n}{\log n}).$$

□

Before derive the next theorem, we state the Theorem 2 in [77]

- Boundedness: for two constants L_Ω and U_Ω such that for all $\pi \in \Pi$, we have $L_\Omega \leq \Omega(\pi) \leq U_\Omega$, then

$$V^*(s) - \frac{\tau(U_\Omega - L_\Omega)}{1-\gamma} \leq V^*_\Omega(s) \leq V^*(s). \tag{A.54}$$

Where τ is the temperature and γ is the discount constant.

Theorem 11. *For any $\delta > 0$, with probability at least $1 - \delta$, the ε_Ω satisfies*

$$-\sqrt{\frac{\hat{C}\sigma^2\log\frac{C}{\delta}}{2N(s)}} - \frac{\tau(U_\Omega - L_\Omega)}{1-\gamma} \leq \varepsilon_\Omega \leq \sqrt{\frac{\hat{C}\sigma^2\log\frac{C}{\delta}}{2N(s)}}.$$

Proof. From Theorem 2, let us define $\delta = C\exp\{-\frac{2N(s)\epsilon^2}{\hat{C}\sigma^2}\}$, so that $\epsilon = \sqrt{\frac{\hat{C}\sigma^2\log\frac{C}{\delta}}{2N(s)}}$ then for any $\delta > 0$, we have

$$\mathbf{Pr}(|V_\Omega(s) - V^*_\Omega(s)| \leq \sqrt{\frac{\hat{C}\sigma^2\log\frac{C}{\delta}}{2N(s)}}) \geq 1 - \delta.$$

Then, for any $\delta > 0$, with probability at least $1 - \delta$, we have

$$
\begin{aligned}
&|V_\Omega(s) - V^*_\Omega(s)| \leq \sqrt{\frac{\hat{C}\sigma^2 \log \frac{C}{\delta}}{2N(s)}} \\
&-\sqrt{\frac{\hat{C}\sigma^2 \log \frac{C}{\delta}}{2N(s)}} \leq V_\Omega(s) - V^*_\Omega(s) \leq \sqrt{\frac{\hat{C}\sigma^2 \log \frac{C}{\delta}}{2N(s)}} \\
&-\sqrt{\frac{\hat{C}\sigma^2 \log \frac{C}{\delta}}{2N(s)}} + V^*_\Omega(s) \leq V_\Omega(s) \leq \sqrt{\frac{\hat{C}\sigma^2 \log \frac{C}{\delta}}{2N(s)}} + V^*_\Omega(s).
\end{aligned}
$$

From Proposition 1, we have

$$
-\sqrt{\frac{\hat{C}\sigma^2 \log \frac{C}{\delta}}{2N(s)}} + V^*(s) - \frac{\tau(U_\Omega - L_\Omega)}{1-\gamma} \leq V_\Omega(s) \leq \sqrt{\frac{\hat{C}\sigma^2 \log \frac{C}{\delta}}{2N(s)}} + V^*(s).
$$

□

A.3. A Unified Perspective on Value Backup and Exploration in Monte-Carlo Tree Search

Theorem 12. *When $\alpha \in (0, 1)$, the regret of E3W [44] with the regularizer f_α is*

$$
\mathbb{E}[R_n] \leq \frac{\tau}{\alpha(1-\alpha)}(|\mathcal{A}|^{1-\alpha} - 1) + n(2\tau)^{-1}|\mathcal{A}|^\alpha + \mathcal{O}(\frac{n}{\log n}).
$$

Proof. Please refer to equation (A.54) in Theorem 10 □

Theorem 13. *When $\alpha \in (1, \infty)$, $\alpha \neq 2$, the regret of E3W [44] with the regularizer f_α is*

$$
\mathbb{E}[R_n] \leq \frac{\tau}{\alpha(1-\alpha)}(|\mathcal{A}|^{1-\alpha} - 1) + \frac{n|\mathcal{K}|}{2} + \mathcal{O}(\frac{n}{\log n}).
$$

where $|\mathcal{K}|$ is the number of actions that are assigned non-zero probability in the policy at the root node.

Proof. From Theorem 10, we have

$$\mathbb{E}[R_n] \leq -\tau\Omega(\hat{\pi}) + \sum_{t=1}^{n} \mathcal{D}_{\Omega^*}(\hat{V}_t(\cdot) + V(\cdot), \hat{V}_t(\cdot)) + \mathcal{O}(\frac{n}{\log n}).$$

Here, $\Omega(\hat{\pi}) = f_\alpha(\hat{\pi}) = \frac{1}{\alpha(1-\alpha)}(1 - \sum_i \hat{\pi}^\alpha(a_i|s))$. So as the result from Theorem 10, we have

$$-\Omega(\hat{\pi}_n) \leq \frac{1}{\alpha(1-\alpha)}(|\mathcal{A}|^{1-\alpha} - 1).$$

By Taylor's theorem $\exists z \in \text{conv}(\hat{V}_t, \hat{V}_t + V)$, we have

$$\mathcal{D}_{\Omega^*}(\hat{V}_t(\cdot) + V(\cdot), \hat{V}_t(\cdot)) \leq \frac{1}{2}\left\langle V(\cdot), \nabla^2\Omega^*(z)V(\cdot)\right\rangle.$$

So that according to Equations (4.8), (4.9), (4.10), (4.11), we have

$$\mathcal{D}_{\Omega^*}(\hat{V}_t(\cdot) + V(\cdot), \hat{V}_t(\cdot)) \leq \frac{1}{2}\left\langle V(\cdot), \nabla^2\Omega^*(z)V(\cdot)\right\rangle \leq \frac{|\mathcal{K}|}{2}.$$

so that

$$\mathbb{E}[R_n] \leq \frac{\tau}{\alpha(1-\alpha)}(|\mathcal{A}|^{1-\alpha} - 1) + \frac{n|\mathcal{K}|}{2} + \mathcal{O}(\frac{n}{\log n}).$$

□

We analyse the error of the regularized value estimate at the root node $n(s)$ w.r.t. the optimal value: $\varepsilon_\Omega = V_\Omega(s) - V^*(s)$. where Ω is the α-divergence regularizer f_α.

Theorem 14. *For any $\delta > 0$ and α-divergence regularizer f_α ($\alpha \neq 1, 2$), with some constant $C, \hat{C}$, with probability at least $1 - \delta$, ε_Ω satisfies*

$$-\sqrt{\frac{\hat{C}\sigma^2 \log\frac{C}{\delta}}{2N(s)}} - \frac{\tau}{\alpha(1-\alpha)}(|\mathcal{A}|^{1-\alpha} - 1) \leq \varepsilon_\Omega \leq \sqrt{\frac{\hat{C}\sigma^2 \log\frac{C}{\delta}}{2N(s)}}. \tag{A.55}$$

Proof. We have

$$0 \leq -\Omega(\hat{\pi}_n) \leq \frac{1}{\alpha(1-\alpha)}(|\mathcal{A}|^{1-\alpha} - 1).$$

combine with Theorem 11 we will have

$$-\sqrt{\frac{\hat{C}\sigma^2 \log \frac{C}{\delta}}{2N(s)}} - \frac{\tau}{\alpha(1-\alpha)}(|\mathcal{A}|^{1-\alpha} - 1) \leq \varepsilon_\Omega \leq \sqrt{\frac{\hat{C}\sigma^2 \log \frac{C}{\delta}}{2N(s)}}. \tag{A.56}$$

□

Curriculum Vitae

Tuan Dam

Technische Universität Darmstadt
Hochschulstr. 10
64289 Darmstadt, Germany
Computer Science
Email: tuanquangdam@gmail.com

https://www.ias.informatik.tu-darmstadt.de/Team/TuanDam
TU Darmstadt, Hessen, Germany
cell: available upon request

Research Interests: Reinforcement Learning under uncertainty, Monte Carlo Tree Search, Multi Armed Bandit, POMDPs, MDPs, Information Theory, Robotics

EDUCATION

TU Darmstadt, Hessen, Germany 2018 - now
PhD Candidate at Intelligent Autonomous Systems Group
Research: Monte-Carlo Tree Search, MDP, POMDP, Robot Path Planning

Hanyang University, Seoul, Korea 2014 - 2016
M.S. in Electrical and Computer Engineering

Posts & Telecommunications Institute of Technology (PTIT), Vietnam 2003 - 2007
B.S. in Computer Science

AWARDS

2004: **Silver prize** National Mathematical Medalist

2006: Encouragement prize of the Student Scientific Research at University

2007: Golden prize of the Student Scientific Research at University

2015: **The best researcher student** in OSLab

PUBLICATIONS Google Scholar

Conferences

Dam, T, Klink, P.; D'Eramo, C.; Peters, J.; Pajarinen, J. (2020). Generalized Mean Estimation in Monte-Carlo Tree Search, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).[pdf]

Dam, T, D'Eramo, C.; Peters, J.; Pajarinen J. (2021). Convex Regularization in Monte-Carlo Tree Search, Proceedings of the International Conference on Machine Learning (ICML).[pdf]

Dam, T; Chalvatzaki, G; Peters, J.; Pajarinen J. (2022). Monte-Carlo Robot Path Planning, IEEE/RSJ International Conference on Intelligent Robots and Systems 2022.

Journals

Dam, T, D'Eramo, C.; Peters, J.; Pajarinen J. (2022). A Unified Perspective on Value Backup and Exploration in Monte-Carlo Tree Search, Submitted to the Journal of Machine Learning Research (JMLR).[pdf]

Dam, T; Chalvatzaki, G; Peters, J.; Pajarinen J. (2022). Monte-Carlo Robot Path Planning, IEEE Robotics and Automation Letters 2022.

Technical reports

Dam, T. Regularize Sparse Markov Decision Processes for Reinforcement Learning.[pdf]

Dam, T. SAPD: Nonlinear Function Approximation Convergence in Reinforcement Learning. [pdf]

WORK EXPERIENCE Github

Auburn University 2017

Research Assistant, USA

DFKI: German Research Center for Artificial Intelligence 2017

Research Assistant, Berlin, Germany

HMI lab, VNU 2016 - 2017

Research Assistant, Hanoi, Vietnam

OSLab 2014 - 2016

Research Assistant, Master Student, Seoul, Korea

Vietinbank 2011 - 2014

Senior Software Developer, Hanoi, Vietnam

Nomovok 2008 - 2011

Software Developer, Hanoi, Vietnam

TALKS

A Unified Perspective on Value Backup and Exploration in Monte-Carlo Tree Search [video] [pdf] Scool, Inria

TEACHING

Robot Learning Summer 2022

Robot Learning Winter 2021/2022

Robot Learning: Integrated Project, Part 1 (Literature Review and Simulation Studies) Winter 2021/2022

Robot Learning: Integrated Project, Part 2 (Evaluation and Submission to a Conference) Winter 2021/2022

SUPERVISION

MS Thesis 2022, Pascal Stenger, Above Average Decision Making Under Uncertainty

MS Thesis 2022, (co-supervised with Junning Huang), Ruidi He, Path Consistency Learning for Autonomous Car Driving

MS Thesis 2021, (co-supervised with Joni Pajarinen and Georgia Chalvatzaki), Cedric Derstroff, Memory Representations for Partially Observable Reinforcement Learning

Robot Learning: Integrated Project Winter 2021, (co-supervised with Carlo D'Eramo), Lukas Schneider, Benchmarking advances in MCTS in Go and Chess

Robot Learning: Integrated Project Winter 2021, (co-supervised with Georgia Chalvatzaki, and Carlo D'Eramo), Daniel Mansfeld, Alex Ruffini, Learning Laplacian Representations for continuous MCTS

Robot Learning: Integrated Project Winter 2019, (co-supervised with Boris Belousov), Maximilian Hensel, Accelerated Mirror Descent Policy Search

REVIEWING

ICML, NeurIPS, AAAI, IROS, CoRL

PATENTS

SQLite SHADOW Journaling Mode: Techniques for ensuring data integrity by maintaining the original and replica of the database for journaling of SQLite databases on Android-based mobile systems. *Patent approved by Korean Government Research*.

SKILLS

Programming: Python (e.g., PyTorch, Tensorflow), LaTeX, Linux, C/C++, Git.

Languages: Vietnamese (Native speaker), English (Fluent)

List of Figures

1.1. This figure illustrates the four basic steps of Monte-Carlo Tree Search. . . 2
1.2. This figure illustrates the outline of the thesis. 7

2.1. Evaluating Power-UCT w.r.t. different p-values: The mean discounted total reward at 65536 simulations (shaded area denotes standard error) over 100 evaluation runs. 20
2.2. Performance of Power-UCT compared to UCT in *rocksample*. The mean of total discounted reward over 1000 evaluation runs is shown by thick lines while the shaded area shows standard error. 21
2.3. Performance of Power-UCT compared to UCT and MENTS in *rocksample* 11x11. The mean of discounted total reward over 1000 evaluation runs is shown by thick lines while the shaded area shows standard error. 23

3.1. For each algorithm, we show the convergence of the value estimate at the root node to the respective optimal value (top), to the UCT optimal value (middle), and the regret (bottom). 35
3.2. For different branching factor k (rows) and depth d (columns), the heatmaps show: the absolute error of the value estimate at the root node after the last simulation of each algorithm w.r.t. the respective optimal value (a), and w.r.t. the optimal value of UCT (b); regret at the root node (c). 37
3.3. High branching factor trees (a), regret sensitivity study w.r.t. ε and τ (b, c). 38

4.1. We show the convergence of the value estimate at the root node to the respective optimal value (top), to the UCT optimal value (middle), and the regret (bottom) with different α parameter of α-divergence in Synthetic tree environment with $\alpha = 1.0$ (MENTS), 1.5, 2.0 (TENTS), 4.0, 8.0, 16.0. 50

4.2. We show the effectiveness of α-divergence in Synthetic Tree environment with different branching factor k (rows) and depth d (columns). The heatmaps show: the absolute error of the value estimate at the root node after the last simulation of each algorithm w.r.t. the respective optimal value (a), and w.r.t. the optimal value of UCT (b); regret at the root node (c). 51

5.1. Four stages of MCPP planner to traverse from the initial position (in green color) to the goal position (in blue color). 55

5.2. Demonstration of path planning using MCPP in a robotic disentangling task. A 7-DOF robotic KUKA arm tries to extract an object from the cardboard box through the hole in the back of the box. The robot does not use any sensors except for proprioception, making the task partially observable. Therefore, the task requires advanced MCPP-based path planning that takes information gathering about the environment into account. We put a limit to prevent the robot arm to move the hand up, therefore, the robot arm has to find the path from the start position on the left side to the goal position on the right side inside the box. 55

5.3. 2-D sketch of the proof for exponential convergence of MCPP to the optimal path in MDPs. The MDP proof relies on showing that MCPP convergences exponentially to a path starting from $\mathbf{x}_0$ and ending at $\mathbf{x}_l$ while the agent stays inside a tube composed of a sequence of spheres with a radius of ϵ. . 61

5.4. Sketch of how to generate paths for MCPP algorithm from $\mathbf{x}_{\text{INIT}}$ to $\mathbf{x}_{\text{GOAL}}$ positions with minimum number of via-points in POMDP environments. . 65

5.5. U-Shape 2D POMDP. Green point is the start position. Blue point is the goal position. Red points are the collisions. The figure shows a success case using MCPP, where the blue line depicts the 2D trajectory of the end effector. Note that in all the 2D experiments we plan in the configuration space using a 7-DOF planar robot arm model illustrated on the right. . . 68

5.6. L-Shape 2D POMDP. Green point is the start position. Blue point is the goal position. Red points are the collisions. The blue lines are the planning path. The figure shows a failure case of RRT* and a success case for MCPP, which shows that it is more explorative. Over 20 random seeds, RRT* failure to solve the problem with 0% success, while MCPP obtain 100% success with UCT and Power-UCT. TENTS gets 85%. 68

5.7. High Wall Environment in 3D. Grey point is the start position. Blue point is the goal position. Red points are collisions. Over 20 random seeds, RRT* can only success with 35%, UCT obtains 55% success rate. Power-UCT gets 70% success rate while TENTS gets 45% success rate. 69

List of Tables

2.1. Mean and two times standard deviation of the success rate, over 500 evaluation runs, of UCT, Power-UCT and MENTS in *FrozenLake* from OpenAI Gym. The top row of each table shows the number of simulations used for tree-search at each time step. 20
2.2. Mean and two times standard deviation of discounted total reward, over 100 evaluation runs, of UCT, Power-UCT and MENTS in the copy environment with 144 actions (top) and 200 actions (bottom). Top row: number of simulations at each time step. 22
2.3. Discounted total reward in *pocman* for the comparison methods. Mean $\pm$ standard error are computed from 1000 simulations except in MENTS where we ran 100 simulations. 24

3.1. List of entropy regularizers with Legendre-Fenchel transforms and maximizing arguments (Max arg. : Max argument). 33
3.2. Average score in Atari over 100 seeds per game. Bold denotes no statistically significant difference to the highest mean (t-test, $p < 0.05$). Bottom row shows # no difference to highest mean. 42

5.1. Comparison for the U-Shape 2D POMDP 70
5.2. Comparison for the High-Wall 3D POMDP 70
5.3. Comparison for the real robot object disentangling 70

List of Algorithms

2.1. Pseudocode of Power-UCT. 15

5.1. Pseudocode of MCPP. 62

List of Acronyms

Notation	Description
AI	Artificial Intelligence
A3C	Asynchronous Advantage Actor Critic
DL	Deep Learning
DESPOT	Determinized Sparse Partially Observable Tree
DOF	Degrees Of Freedom
DQN	Deep Q Learning
Deep-MCTS	Deep Monte-Carlo Tree Search
E3W	Extended Empirical Exponential Weight
E2W	Empirical Exponential Weight
MCPP	Monte-Carlo Path Planning
MCTS	Monte-Carlo Tree Search
MDP	Markov Decision Process
MDPs	Markov Decision Processes
MENTS	Maximum Entropy Monte-Carlo Planning
ML	Machine Learning
POMCP	Partially Observable Monte Carlo Planning
POMCP-DPW	Partially Observable Monte Carlo Planning Double Progressive Widening
POMDP	Partially observable Markov Decision Process

POMDPs	Partially observable Markov Decision Processes
Power-UCT	Power Mean Upper Confidence bound Tree
PUCT	Polynomial Upper Confidence bound Tree
SAC	Soft Actor-Critic
SARSA	State-Action–Reward–State–Action
RL	Reinforcement learning
RENTS	Relative Entropy Monte-Carlo Planning
RRT	Rapidly exploring Random Tree
RRTs	Rapidly exploring Random Trees
TENTS	Tsallis Entropy Monte-Carlo Planning
TRPO	Trust Region Policy Optimization
UCB	Upper Confidence Bound
UCT	Upper Confidence bound Tree

List of Symbols

Notation	Description
Pr	Probability
$\mathbb{E}$	Expectation
Var	Variance
exp	Exponential power
log	logarithm
lim	limit
$\sum$	sum
$\mathbb{I}(.)$	indicator function
$\|\cdot\|_p$	p-norm
$\|\cdot\|_\infty$	maximum-norm
$\|.\|$	absolute value
$\Omega(.)$	regularizer
$\Omega^*(.)$	Legendre-Fenchel transform
D_{KL}	Kullback–Leibler divergence
$\mathrm{D}_f(P \parallel Q)$	f-divergence between two distributions P and Q
$\mathrm{D}_\alpha(P \parallel Q)$	α-divergence between two distributions P and Q
$\mathrm{H}_\alpha^\pi(s)$	generalization of Tsallis entropy of policy π at state s with parameter α
f_α	α function

$\mathcal{D}_{\Omega^*}(x, y)$	Bregman divergence between x and y
R_n	pseudo regret
τ	regularization temperature
$\Delta\Omega^*(.)$	max argument
L_Ω	lower bound of Ω
U_Ω	upper bound of Ω
$\lvert\mathcal{A}\rvert$	number of actions in set $\mathcal{A}$
$\lvert\mathcal{K}\rvert$	number of actions that are assigned non-zero probability
$\int$	integral
$\triangleq$	is defined as
s	state s
a	action a
r	reward
$\mathcal{S}$	state space
$\mathcal{A}$	action space
$\mathcal{P}$	transition kernel
$\mathcal{R}$	reward function
$\mathcal{O}$	Complexity
Π	policy set
γ	discount factor
ϵ	positive constant epsilon
$N(s)$	number of visitations at node V state s
$n(s, a)$	number of visitations at node Q state, action s, a
$T_i(n)$	the number of times arm i is played up to time n
$\mathcal{T}^*Q(s, a)$	optimal Bellman operator

$\mathcal{T}_\pi Q(s,a)$	Bellman operator under policy π
$\tau(s,a)$	transition function at state s, taking action a
$\bar{X}_{i,T_i(n)}$	average reward of arm i up to time n
$V(s)$	V value function at state s
$V^*(s)$	optimal V value function at state s
$V_\Omega(s)$	regularized V value function at state s
$V^*_\Omega(s)$	optimal regularized V value function at state s
$Q(s,a)$	Q value function at state s action a
$Q^*(s,a)$	optimal Q value function at state s action a
$Q_\Omega(s)$	regularized Q value function at state s
$Q^*_\Omega(s)$	optimal regularized Q value function at state s
π	policy
μ	mean
$\mu*$	optimal mean value

Bibliography

[1] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[2] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[3] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 712–733, 2020.

[4] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on robotics and automation*, vol. 16, no. 5, pp. 615–620, 2000.

[5] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[6] N. Sariff and N. Buniyamin, "An overview of autonomous mobile robot path planning algorithms," in *2006 4th student conference on research and development*, pp. 183–188, IEEE, 2006.

[7] H.-y. Zhang, W.-m. Lin, and A.-x. Chen, "Path planning for the mobile robot: A review," *Symmetry*, 2018.

[8] J. Mainprice and D. Berenson, "Human-robot collaborative manipulation planning using early prediction of human motion," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 299–306, IEEE, 2013.

[9] M. Chen, S. Nikolaidis, H. Soh, D. Hsu, and S. Srinivasa, "Planning with trust for

human-robot collaboration," in *Proceedings of the 2018 ACM/IEEE international conference on human-robot interaction*, pp. 307–315, 2018.

[10] M. Cutler, T. J. Walsh, and J. P. How, "Real-world reinforcement learning via multifidelity simulators," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 655–671, 2015.

[11] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[12] L. Kocsis, C. Szepesvári, and J. Willemson, "Improved monte-carlo search," *Univ. Tartu, Estonia, Tech. Rep*, vol. 1, 2006.

[13] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[14] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[15] S. Mo, X. Pei, and C. Wu, "Safe reinforcement learning for autonomous vehicle using monte carlo tree search," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[16] N. Funk, G. Chalvatzaki, B. Belousov, and J. Peters, "Learn2assemble with structured representations and search for robotic architectural construction," in *Proceedings of the 5th Conference on Robot Learning* (A. Faust, D. Hsu, and G. Neumann, eds.), vol. 164 of *Proceedings of Machine Learning Research*, pp. 1401–1411, PMLR, 08–11 Nov 2022.

[17] S. Eiffert, H. Kong, N. Pirmarzdashti, and S. Sukkarieh, "Path planning in dynamic environments using generative rnns and monte carlo tree search," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10263–10269, IEEE, 2020.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep

convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[21] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[23] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL*, 2018.

[24] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[25] H. A. Pierson and M. S. Gashler, "Deep learning in robotics: a review of recent research," *Advanced Robotics*, vol. 31, no. 16, pp. 821–835, 2017.

[26] L. Tai, J. Zhang, M. Liu, J. Boedecker, and W. Burgard, "A survey of deep network solutions for learning control in robotics: From reinforcement to imitation," *arXiv preprint arXiv:1612.07139*, 2016.

[27] L. Tai and M. Liu, "Deep-learning in mobile robotics-from perception to control systems: A survey on why and why not," *arXiv preprint arXiv:1612.07139*, vol. 1, 2016.

[28] 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, p. 484, 2016.

[29] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*, vol. 135. MIT press Cambridge, 1998.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[31] C. Xiao, R. Huang, J. Mei, D. Schuurmans, and M. Müller, "Maximum entropy monte-carlo planning," in *Advances in Neural Information Processing Systems*, pp. 9516–9524, 2019.

[32] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *International conference on computers and games*, Springer, 2006.

[33] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," *Advances in neural information processing systems*, vol. 29, pp. 4026–4034, 2016.

[34] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang, "Deep learning for real-time atari game play using offline monte-carlo tree search planning," *Advances in neural information processing systems*, vol. 27, 2014.

[35] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, 2014.

[36] I. Noreen, A. Khan, Z. Habib, *et al.*, "Optimal path planning using rrt* based approaches: a survey and future directions," *Int. J. Adv. Comput. Sci. Appl*, 2016.

[37] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2, pp. 995–1001, IEEE, 2000.

[38] M. Kleinbort, K. Solovey, Z. Littlefield, K. E. Bekris, and D. Halperin, "Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation," *IEEE RA-L*, 2018.

[39] M. Ivanov, L. Lindner, O. Sergiyenko, J. C. Rodríguez-Quiñonez, W. Flores-Fuentes, and M. Rivas-Lopez, "Mobile robot path planning using continuous laser scanning," in *Optoelectronics in machine vision-based theories and applications*, pp. 338–372, IGI Global, 2019.

[40] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE Trans. on robotics and automation*, 2002.

[41] V. J. Lumelsky, "Dynamic path planning for a planar articulated robot arm moving amidst unknown obstacles," *Automatica*, 1987.

[42] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 397–422, 2002.

[43] T. Dam, P. Klink, C. D'Eramo, J. Peters, and J. Pajarinen, "Generalized mean estimation in monte-carlo tree search," 2019.

[44] T. Q. Dam, C. D'Eramo, J. Peters, and J. Pajarinen, "Convex regularization in monte-carlo tree search," in *International Conference on Machine Learning*, pp. 2365–2375, PMLR, 2021.

[45] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[46] C. Watkins, *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, England, 1989.

[47] J. E. Smith and R. L. Winkler, "The optimizer's curse: Skepticism and postdecision surprise in decision analysis," *Management Science*, vol. 52, no. 3, 2006.

[48] H. V. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, 2010.

[49] C. D'Eramo, M. Restelli, and A. Nuara, "Estimating maximum expected value through gaussian approximation," in *International Conference on Machine Learning*, 2016.

[50] P. S. Bullen, *Handbook of means and their inequalities*. Springer Science & Business Media, 2013.

[51] G. Tesauro, V. Rajan, and R. Segal, "Bayesian inference in monte-carlo tree search," *arXiv preprint arXiv:1203.3519*, 2012.

[52] C. Mansley, A. Weinstein, and M. Littman, "Sample-based planning for continuous action markov decision processesgeneralizedfmean," in *Twenty-First International Conference on Automated Planning and Scheduling*, 2011.

[53] S. Gelly and Y. Wang, "Exploration exploitation in go: Uct for monte-carlo go," in *NIPS: Neural Information Processing Systems Conference On-line trading of Exploration and Exploitation Workshop*, 2006.

[54] F. Teytaud and O. Teytaud, "On the huge benefit of decisive moves in monte-carlo tree search algorithms," in *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, IEEE, 2010.

[55] B. E. Childs, J. H. Brodeur, and L. Kocsis, "Transpositions and move groups in monte carlo tree search," in *2008 IEEE Symposium On Computational Intelligence and Games*, IEEE, 2008.

[56] T. Kozelek, "Methods of mcts and the game arimaa," *Univerzita Karlova, Matematicko-fyzikální fakulta*, 2009.

[57] G. Chaslot, M. Winands, J. V. D. Herik, J. Uiterwijk, and B. Bouzy, "Progressive strategies for monte-carlo tree search," *New Mathematics and Natural Computation*, vol. 4, no. 03, pp. 343–357, 2008.

[58] S. Gelly and D. Silver, "Combining online and offline knowledge in uct," in *Proceedings of the 24th international conference on Machine learning*, pp. 273–280, ACM, 2007.

[59] D. P. Helmbold and A. Parker-Wood, "All-moves-as-first heuristics in monte-carlo go.," in *IC-AI*, pp. 605–610, 2009.

[60] R. J. Lorentz, "Improving monte–carlo tree search in havannah," in *International Conference on Computers and Games*, pp. 105–115, Springer, 2010.

[61] D. Tom, "Investigating uct and rave: Steps towards a more robust method," *Master thesis, University of Alberta*, 2010.

[62] J.-B. Hoock, C.-S. Lee, A. Rimmel, F. Teytaud, M.-H. Wang, and O. Teytaud, "Intelligent agents for the game of go," *IEEE Computational Intelligence Magazine*, 2010.

[63] P. Khandelwal, E. Liebman, S. Niekum, and P. Stone, "On the analysis of complex backup strategies in monte carlo tree search," in *International Conference on Machine Learning*, 2016.

[64] T. Vodopivec, S. Samothrakis, and B. Ster, "On monte carlo tree search and reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 60, pp. 881–936, 2017.

[65] G. A. Rummery, *Problem solving with reinforcement learning*. PhD thesis, University of Cambridge Ph. D. dissertation, 1995.

[66] D. S. Mitrinovic and P. M. Vasic, *Analytic inequalities*. Springer, 1970.

[67] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[68] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," in *Advances in neural information processing systems*, 2010.

[69] T. Smith and R. Simmons, "Heuristic search value iteration for pomdps," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pp. 520–527, AUAI Press, 2004.

[70] T. Yee, V. Lisỳ, M. H. Bowling, and S. Kambhampati, "Monte carlo tree search in continuous action spaces with execution uncertainty.," in *IJCAI*, pp. 690–697, 2016.

[71] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[72] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, pp. 1889–1897, 2015.

[73] J. Schulman, X. Chen, and P. Abbeel, "Equivalence between policy gradients and soft q-learning," *arXiv preprint arXiv:1704.06440*, 2017.

[74] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*, pp. 1861–1870, 2018.

[75] L. Buesing, N. Heess, and T. Weber, "Approximate inference in discrete distributions with monte carlo tree search and value functions," in *International Conference on Artificial Intelligence and Statistics*, pp. 624–634, PMLR, 2020.

[76] A. Mensch and M. Blondel, "Differentiable dynamic programming for structured prediction and attention," in *International Conference on Machine Learning*, pp. 3462–3471, 2018.

[77] M. Geist, B. Scherrer, and O. Pietquin, "A theory of regularized markov decision processes," in *International Conference on Machine Learning*, pp. 2160–2169, 2019.

[78] O. Nachum and B. Dai, "Reinforcement learning via fenchel-rockafellar duality," *CoRR*, vol. abs/2001.01866, 2020.

[79] S. Shalev-Shwartz and Y. Singer, "Convex repeated games and fenchel duality," *Advances in neural information processing systems*, vol. 19, pp. 1265–1272, 2006.

[80] L. Pavel, "An extension of duality to a game-theoretic framework," *Automatica*, vol. 43, no. 2, pp. 226 – 237, 2007.

[81] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[82] K. Lee, S. Choi, and S. Oh, "Sparse markov decision processes with causal sparse tsallis entropy regularization for reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1466–1473, 2018.

[83] R. Bellman, "The theory of dynamic programming," tech. rep., Rand corp santa monica ca, 1954.

[84] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[85] V. Niculae and M. Blondel, "A regularized framework for sparse and structured neural attention," *arXiv preprint arXiv:1705.07704*, 2017.

[86] O. Nachum and B. Dai, "Reinforcement learning via fenchel-rockafellar duality," *arXiv preprint arXiv:2001.01866*, 2020.

[87] M. Geist and B. Scherrer, "L1-penalized projected bellman residual," in *Proceedings of the European Workshop on Reinforcement Learning (EWRL 2011)*, Lecture Notes in Computer Science (LNCS), Springer Verlag - Heidelberg Berlin, september 2011.

[88] B. Belousov and J. Peters, "Entropic regularization of markov decision processes," *Entropy*, vol. 21, no. 7, p. 674, 2019.

[89] P.-A. Coquelin and R. Munos, "Bandit algorithms for tree search," *arXiv preprint cs/0703062*, 2007.

[90] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering atari, go, chess and shogi by planning with a learned model," 2019.

[91] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[92] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 449–458, JMLR. org, 2017.

[93] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, 2016.

[94] W. H. Montgomery and S. Levine, "Guided policy search via approximate mirror descent," in *Advances in Neural Information Processing Systems*, pp. 4008–4016, 2016.

[95] J. Mei, C. Xiao, R. Huang, D. Schuurmans, and M. Müller, "On principled entropy exploration in policy optimization," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 3130–3136, AAAI Press, 2019.

[96] J.-B. Grill, F. Altché, Y. Tang, T. Hubert, M. Valko, I. Antonoglou, and R. Munos, "Monte-carlo tree search as regularized policy optimization," *arXiv preprint arXiv:2007.12509*, 2020.

[97] S. Bubeck, R. Munos, and G. Stoltz, "Pure exploration in finitely-armed and continuous-armed bandits," *Theoretical Computer Science*, vol. 412, pp. 1832–1852, 04 2011.

[98] Q. V. Nguyen, F. Colas, E. Vincent, and F. Charpillet, "Long-term robot motion planning for active sound source localization with monte carlo tree search," in *2017 Hands-free Speech Communications and Microphone Arrays (HSCMA)*, pp. 61–65, IEEE, 2017.

[99] F. Sukkar, G. Best, C. Yoo, and R. Fitch, "Multi-robot region-of-interest reconstruction with dec-mcts," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9101–9107, IEEE, 2019.

[100] N. C. Volpi, Y. Wu, and D. Ognibene, "Towards event-based mcts for autonomous cars," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 420–427, IEEE, 2017.

[101] J. Chen, C. Zhang, J. Luo, J. Xie, and Y. Wan, "Driving maneuvers prediction based autonomous driving control by deep monte carlo tree search," *IEEE transactions on vehicular technology*, vol. 69, no. 7, pp. 7146–7158, 2020.

[102] N. Funk, G. Chalvatzaki, B. Belousov, and J. Peters, "Learn2assemble with structured representations and search for robotic architectural construction," in *5th Annual Conference on Robot Learning*, 2021.

[103] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and

M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[104] K. Lee, S. Kim, S. Lim, and S. Choi, "A unified framework for maximum entropy reinforcement learning," *arXiv preprint arXiv:1902.00137*, 2019.

[105] I. Csiszár, "Eine informationstheoretische ungleichung und ihre anwendung auf beweis der ergodizitaet von markoffschen ketten," *Magyer Tud. Akad. Mat. Kutato Int. Koezl.*, vol. 8, pp. 85–108, 1964.

[106] G. Chen, Y. Peng, and M. Zhang, "Effective exploration for deep reinforcement learning via bootstrapped q-ensembles under tsallis entropy regularization," *arXiv preprint arXiv:1809.00403*, 2018.

[107] A. Ben-Tal, A. Charnes, and M. Teboulle, "Entropic means," *Journal of Mathematical Analysis and Applications*, vol. 139, no. 2, pp. 537–551, 1989.

[108] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, S. Sorkin, *et al.*, "On finding narrow passages with probabilistic roadmap planners," in *Robotics: the algorithmic perspective: 1998 workshop on the algorithmic foundations of robotics*, pp. 141–154, 1998.

[109] H. Baier and P. D. Drake, "The power of forgetting: Improving the last-good-reply policy in monte carlo go," *IEEE Trans. on Computational Intelligence and AI in Games*, 2010.

[110] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," *Ames, IA, USA*, 1998.

[111] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Trans. on Vehicular Techn.*, 2016.

[112] J.-C. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts," *IJRR*, 1999.

[113] G. Sahar and J. M. Hollerbach, "Planning of minimum-time trajectories for robot arms," *IJRR*, 1986.

[114] B. K. Kim and K. G. Shin, "Minimum-time path planning for robot arms and their dynamics," *Trans. on SMC*, 1985.

[115] T. Kunz, U. Reiser, M. Stilman, and A. Verl, "Real-time path planning for a robot arm in changing environments," in *IROS*, 2010.

[116] A. Zelinsky, R. A. Jarvis, J. Byrne, S. Yuta, *et al.*, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Int'l Conf. on Advanced Robotics*, 1993.

[117] C. Alexopoulos and P. M. Griffin, "Path planning for a mobile robot," *IEEE Trans. on SMC*, vol. 22, no. 2, pp. 318–322, 1992.

[118] S. M. Persson and I. Sharf, "Sampling-based a* algorithm for robot path-planning," *IJRR*, 2014.

[119] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[120] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, 2013.

[121] N. Dadkhah and B. Mettler, "Survey of motion planning literature in the presence of uncertainty: Considerations for uav guidance," *Journal of Intelligent & Robotic Systems*, 2012.

[122] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart, "Motion-and uncertainty-aware path planning for micro aerial vehicles," *Journal of Field Robotics*, 2014.

[123] M. Kazemi, K. Gupta, and M. Mehrandezh, "Path-planning for visual servoing: A review and issues," *Visual Servoing via Advanced Numerical Methods*, 2010.

[124] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*, pp. 282–293, Springer, 2006.

[125] M. Brunner, B. Brüggemann, and D. Schulz, "Hierarchical rough terrain motion planning using an optimal sampling-based method," in *2013 IEEE International Conference on Robotics and Automation*, pp. 5539–5544, IEEE, 2013.

[126] L. Palmieri, S. Koenig, and K. O. Arras, "Rrt-based nonholonomic motion planning using any-angle path biasing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2775–2781, IEEE, 2016.

[127] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2997–3004, IEEE, 2014.

[128] B. Kim, K. Lee, S. Lim, L. Kaelbling, and T. Lozano-Pérez, "Monte carlo tree search in continuous spaces using voronoi optimistic optimization with regret bounds," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 9916–9924, 2020.

[129] K. Sun, B. Schlotfeldt, G. J. Pappas, and V. Kumar, "Stochastic motion planning under partial observability for mobile robots with continuous range measurements," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 979–995, 2020.

[130] Z. N. Sunberg and M. J. Kochenderfer, "Online algorithms for pomdps with continuous state, action, and observation spaces," in *Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018.

[131] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," *Advances in neural information processing systems*, vol. 26, pp. 1772–1780, 2013.

[132] T. Smith and R. Simmons, "Point-based pomdp algorithms: Improved analysis and implementation," *arXiv:1207.1412*, 2012.

[133] D. Braziunas, "Pomdp solution methods," *Uof Toronto*, 2003.

[134] J. Pajarinen, O. Arenz, J. Peters, and G. Neumann, "Probabilistic approach to physical object disentangling," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5510–5517, 2020.

[135] C. D'Eramo, A. Nuara, M. Pirotta, and M. Restelli, "Estimating the maximum expected value in continuous reinforcement learning problems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[136] Z. Zhang, Z. Pan, and M. J. Kochenderfer, "Weighted double q-learning.," in *IJCAI*, pp. 3455–3461, 2017.

[137] P. Lv, X. Wang, Y. Cheng, and Z. Duan, "Stochastic double deep q-network," *IEEE Access*, vol. 7, pp. 79446–79454, 2019.

[138] T. Imagaw and T. Kaneko, "Estimating the maximum expected value through upper confidence bound of likelihood," in *2017 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pp. 202–207, IEEE, 2017.

[139] H. Attouch and R. J.-B. Wets, "Isometries for the legendre-fenchel transform," *Transactions of the American Mathematical Society*, vol. 296, no. 1, pp. 33–60, 1986.

[140] A. M. Metelli, A. Likmeta, and M. Restelli, "Propagating uncertainty in reinforcement learning via wasserstein barycenters," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[141] Q. Rao and J. Frtunikj, "Deep learning for self-driving cars: Chances and challenges," in *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, pp. 35–38, 2018.

[142] J. Ni, Y. Chen, Y. Chen, J. Zhu, D. Ali, and W. Cao, "A survey on theories and applications for self-driving cars based on deep learning methods," *Applied Sciences*, vol. 10, no. 8, p. 2749, 2020.

[143] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[144] C. Li, T. Trinh, L. Wang, C. Liu, M. Tomizuka, and W. Zhan, "Efficient game-theoretic planning with prediction heuristic for socially-compliant autonomous driving," *arXiv preprint arXiv:2207.03673*, 2022.

[145] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[146] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[147] S.-I. Amari, "$alpha$-divergence is unique, belonging to both f-divergence and bregman divergence classes," *IEEE Transactions on Information Theory*, vol. 55, no. 11, pp. 4925–4931, 2009.

[148] M. Cuturi and A. Doucet, "Fast computation of wasserstein barycenters," in *International conference on machine learning*, pp. 685–693, PMLR, 2014.

[149] A. Bai, F. Wu, and X. Chen, "Bayesian mixture modelling and inference based thompson sampling in monte-carlo tree search," *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pp. 1646–1654, 2013.

[150] A. Bai, F. Wu, Z. Zhang, and X. Chen, "Thompson sampling based monte-carlo

planning in pomdps," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 24, 2014.

[151] C. F. Hayes, M. Reymond, D. M. Roijers, E. Howley, and P. Mannion, "Risk aware and multi-objective decision making with distributional monte carlo tree search," *arXiv preprint arXiv:2102.00966*, 2021.

[152] B. Mavrin, H. Yao, L. Kong, K. Wu, and Y. Yu, "Distributional reinforcement learning for efficient exploration," in *International conference on machine learning*, pp. 4424–4434, PMLR, 2019.

[153] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[154] L. Wasserman, "All of statistics: a concise course in statistical inference. 2004," 2004.

[155] D. François, V. Wertz, and M. Verleysen, "The concentration of fractional distances," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 7, pp. 873–886, 2007.

[156] M. J. Wainwright, *High-dimensional statistics: A non-asymptotic viewpoint*, vol. 48. Cambridge University Press, 2019.

[157] J. Abernethy, C. Lee, and A. Tewari, "Fighting bandits with a new kind of smoothness," *arXiv preprint arXiv:1512.04152*, 2015.

[158] J. Zimmert and Y. Seldin, "An optimal algorithm for stochastic and adversarial bandits," in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 467–475, PMLR, 2019.