Investigation of possible improvements to increase the efficiency of the AlphaZero algorithm.

Colin Clausen

3.9.2020

Inhalt

Inhalt

Einleitung

Einleitung

- Spiele als Lernumgebung für AI

Einleitung

- Spiele als Lernumgebung für AI
  - 1951: Nimrod

Einleitung

- Spiele als Lernumgebung für AI
  - 1951: Nimrod
  - 1995: Vier gewinnt

Einleitung

- Spiele als Lernumgebung für AI
  - 1951: Nimrod
  - 1995: Vier gewinnt
  - 1997: Schach

Einleitung

- Spiele als Lernumgebung für AI
  - 1951: Nimrod
  - 1995: Vier gewinnt
  - 1997: Schach
  - 2016: Go

Einleitung

- Spiele als Lernumgebung für AI
  - 1951: Nimrod
  - 1995: Vier gewinnt
  - 1997: Schach
  - 2016: Go
- AlphaGo

Einleitung

- Spiele als Lernumgebung für AI
  - 1951: Nimrod
  - 1995: Vier gewinnt
  - 1997: Schach
  - 2016: Go
- AlphaGo
- AlphaZero

Einleitung

- Spiele als Lernumgebung für AI
  - 1951: Nimrod
  - 1995: Vier gewinnt
  - 1997: Schach
  - 2016: Go
- AlphaGo
- AlphaZero
- Großer Rechenaufwand nötig: 5000+ TPUs

Zielsetzung

Zielsetzung

- Untersuche mögliche Effizienzsteigerungen

Zielsetzung

- Untersuche mögliche Effizienzsteigerungen
- Solide Baseline

Zielsetzung

- Untersuche mögliche Effizienzsteigerungen
- Solide Baseline
- Experimentiere mit Vier gewinnt

Zielsetzung

- Untersuche mögliche Effizienzsteigerungen
- Solide Baseline
- Experimentiere mit Vier gewinnt
- Evaluiere verschiedende neue Ideen

Inhalt

Monte Carlo tree search (MCTS)

Monte Carlo tree search (MCTS)

- Seit 2006 sehr verbreitet in Computer Go

Monte Carlo tree search (MCTS)

- Seit 2006 sehr verbreitet in Computer Go
- Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero

Monte Carlo tree search (MCTS)

- Seit 2006 sehr verbreitet in Computer Go
- Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
- Benötigt:

Monte Carlo tree search (MCTS)

- Seit 2006 sehr verbreitet in Computer Go
- Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
- Benötigt:
  - Eine Policy die Züge einschätzen kann

Monte Carlo tree search (MCTS)

- Seit 2006 sehr verbreitet in Computer Go
- Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
- Benötigt:
  - Eine Policy die Züge einschätzen kann
  - Lösung um die Suchtiefe zu begrenzen

Monte Carlo tree search (MCTS)

- Seit 2006 sehr verbreitet in Computer Go
- Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
- Benötigt:
  - Eine Policy die Züge einschätzen kann
  - Lösung um die Suchtiefe zu begrenzen
    - Eine sehr schnelle Rolloutpolicy

Monte Carlo tree search (MCTS)

- Seit 2006 sehr verbreitet in Computer Go
- Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
- Benötigt:
  - Eine Policy die Züge einschätzen kann
  - Lösung um die Suchtiefe zu begrenzen
    - Eine sehr schnelle Rolloutpolicy
    - Alternativ: Policy zur Positionseinschätzung

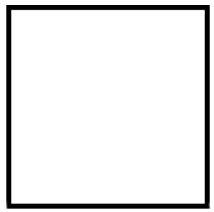
Monte Carlo tree search (MCTS)

- Seit 2006 sehr verbreitet in Computer Go
- Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
- Benötigt:
  - Eine Policy die Züge einschätzen kann
  - Lösung um die Suchtiefe zu begrenzen
    - Eine sehr schnelle Rolloutpolicy
    - Alternativ: Policy zur Positionseinschätzung
- Output: Eine bessere Policy über die Züge in der analysierten Position

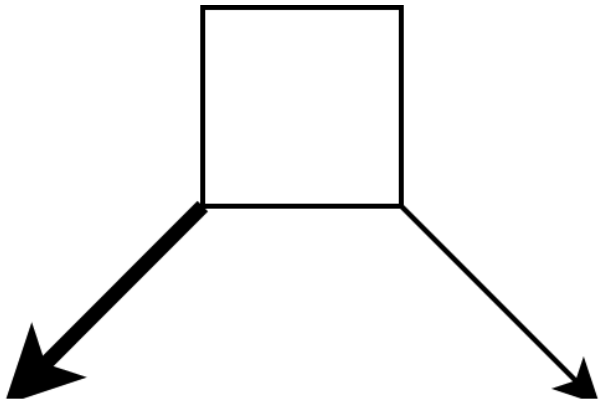
Monte Carlo tree search (MCTS)

- Seit 2006 sehr verbreitet in Computer Go
- Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
- Benötigt:
  - Eine Policy die Züge einschätzen kann
  - Lösung um die Suchtiefe zu begrenzen
    - Eine sehr schnelle Rolloutpolicy
    - Alternativ: Policy zur Positionseinschätzung
- Output: Eine bessere Policy über die Züge in der analysierten Position
- MCTS ist praktisch ein Verbesserungsoperator

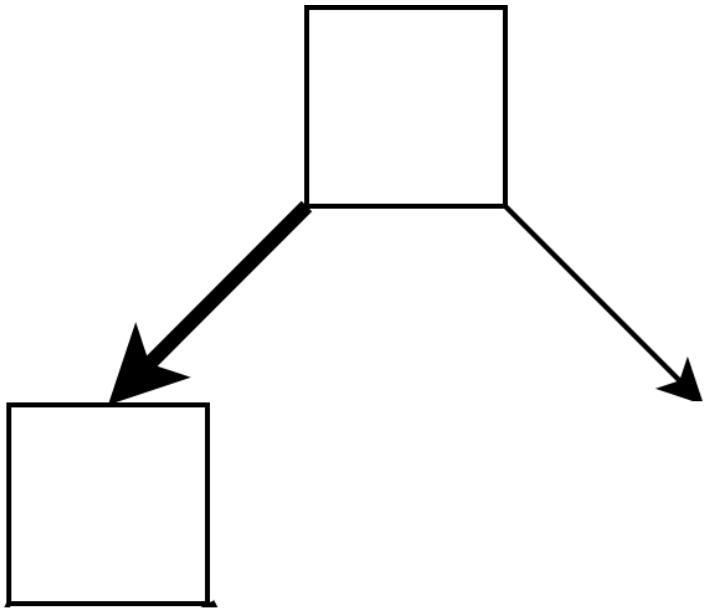
MCTS Beispiel



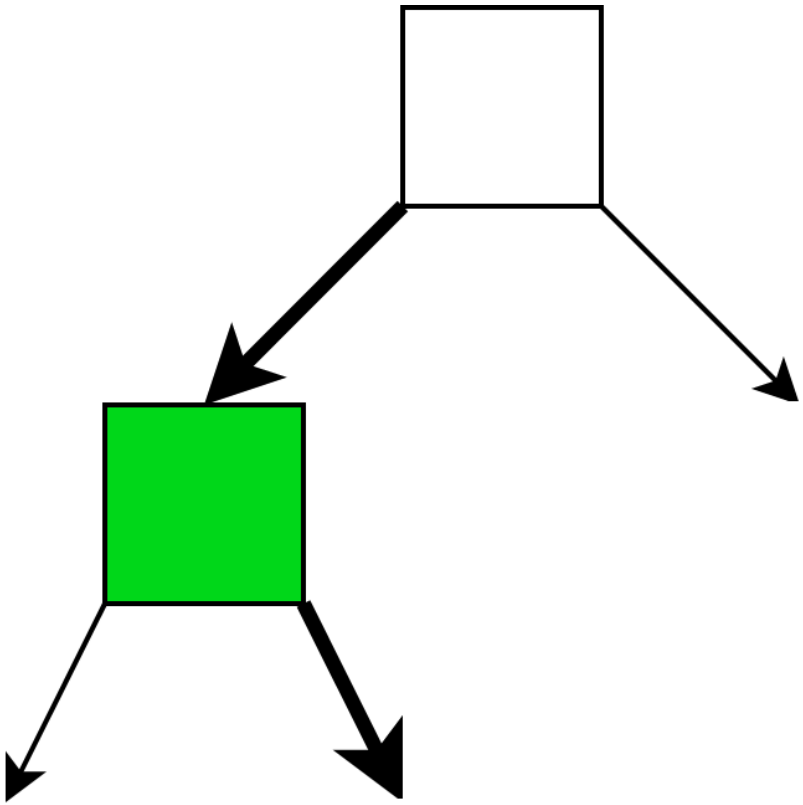
MCTS Beispiel



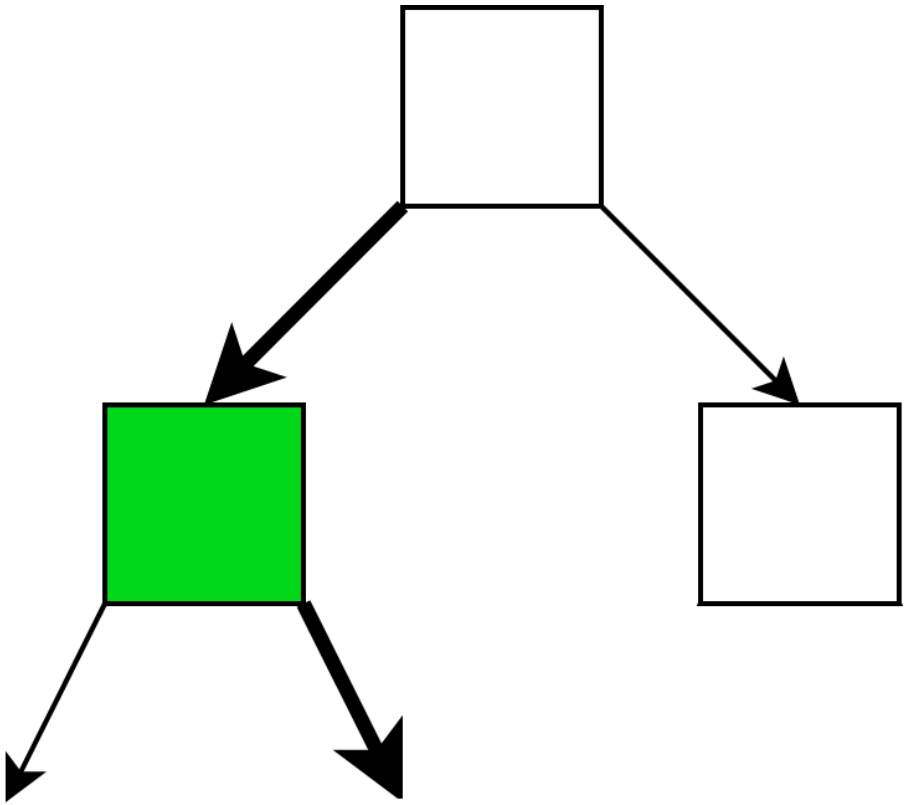
MCTS Beispiel



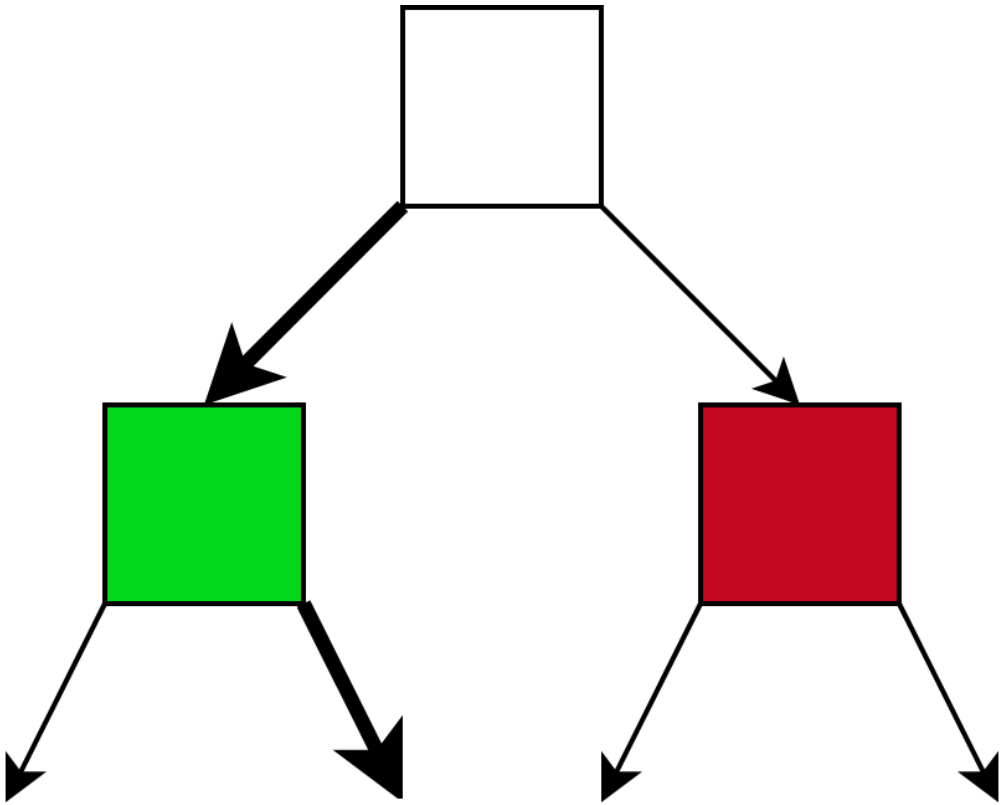
MCTS Beispiel



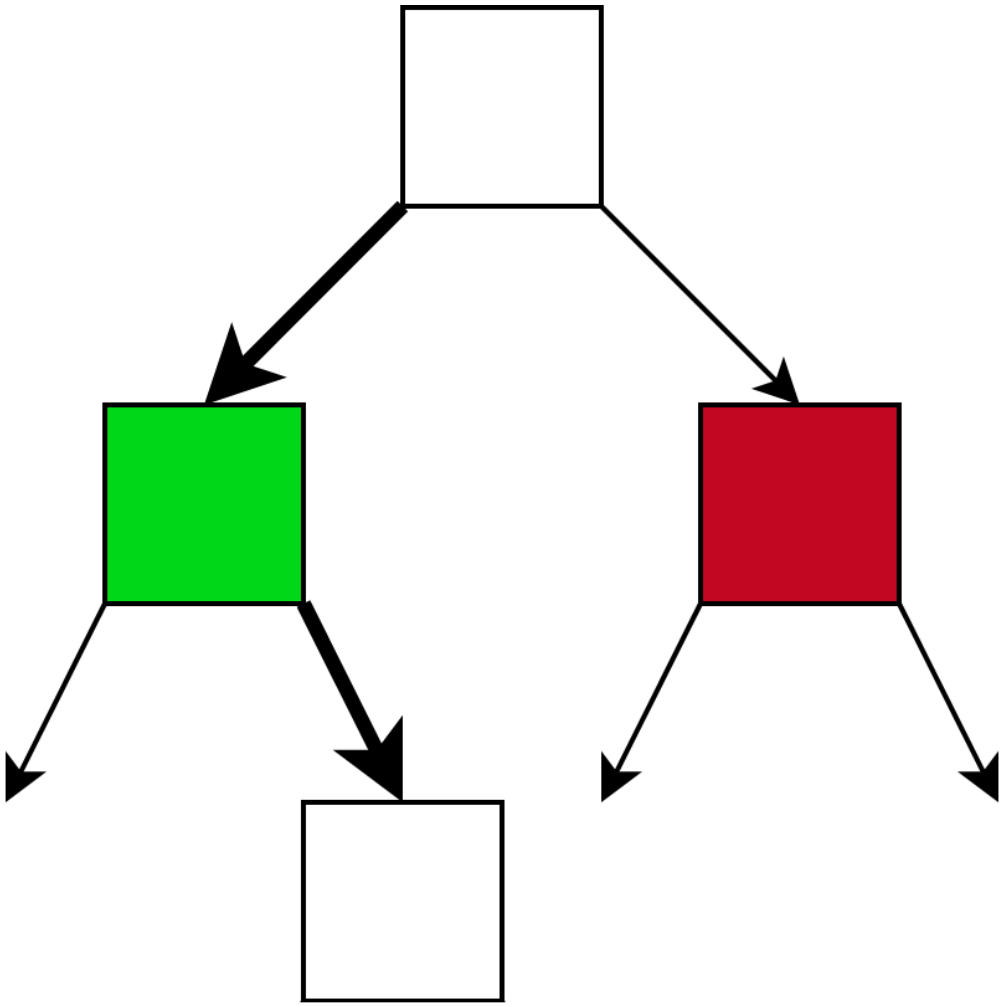
MCTS Beispiel



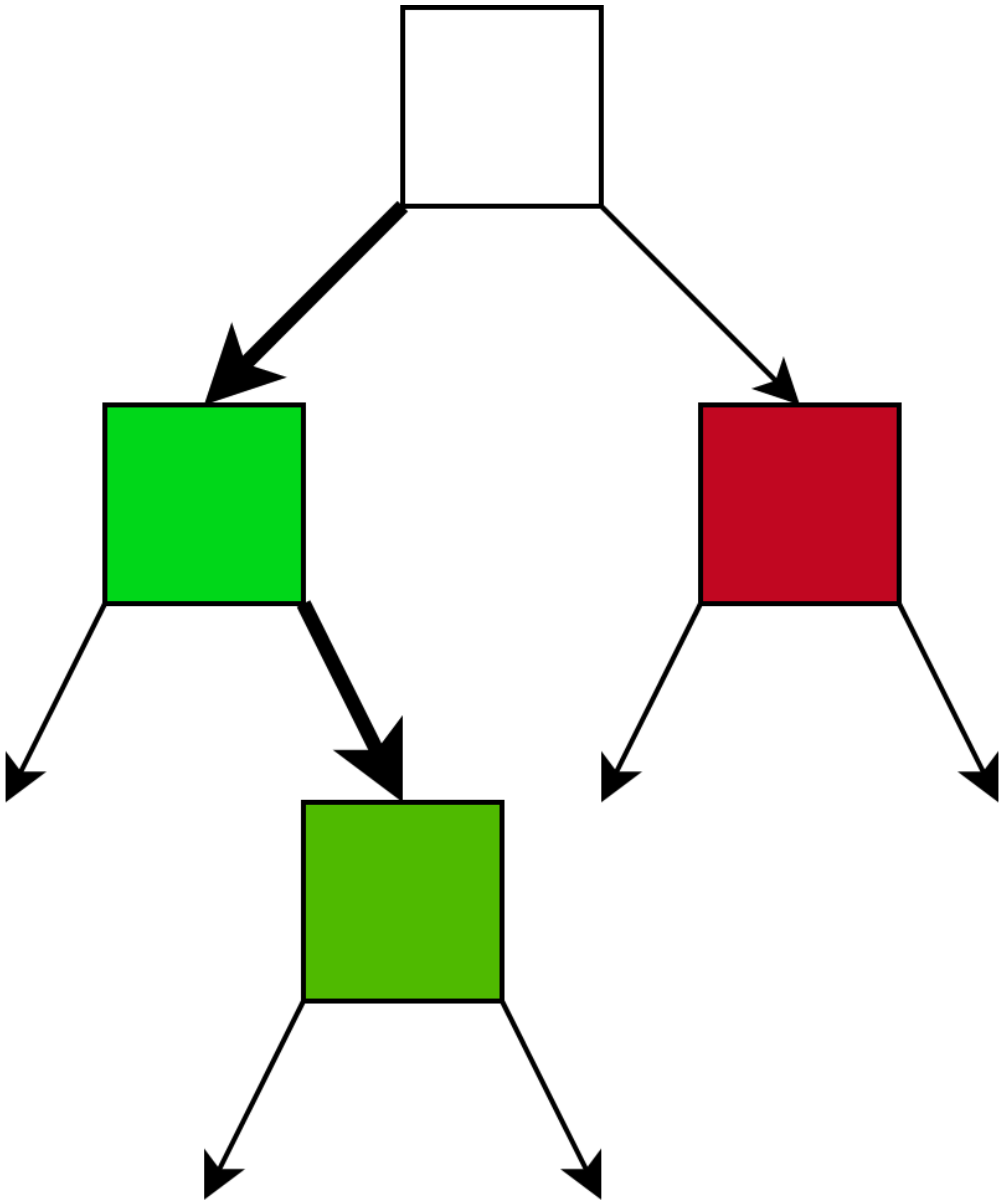
MCTS Beispiel



MCTS Beispiel



MCTS Beispiel



AlphaGo

- Kernidee: Kombinere MCTS mit Deep Learning

- Kernidee: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS

- Kernidee: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS
- Trainere mehrere Netzwerke

- Kernidee: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS
- Trainere mehrere Netzwerke
  - Supervised auf Datensatz von besten Menschen: Schnell und Langsam

AlphaGo

- Kernidee: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS
- Trainere mehrere Netzwerke
  - Supervised auf Datensatz von besten Menschen: Schnell und Langsam
  - Verbessere das langsame Netz durch RL

- Kernidee: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS
- Trainere mehrere Netzwerke
  - Supervised auf Datensatz von besten Menschen: Schnell und Langsam
  - Verbessere das langsame Netz durch RL
  - Erzeuge Datensatz für Netzwerk zur Positionsevaluierung

- Kernidee: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS
- Trainere mehrere Netzwerke
  - Supervised auf Datensatz von besten Menschen: Schnell und Langsam
  - Verbessere das langsame Netz durch RL
  - Erzeuge Datensatz für Netzwerk zur Positionsevaluierung
  - Verwende erzeugte Netzwerke um mit MCTS zu spielen

- Kernidee: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS
- Trainere mehrere Netzwerke
  - Supervised auf Datensatz von besten Menschen: Schnell und Langsam
  - Verbessere das langsame Netz durch RL
  - Erzeuge Datensatz für Netzwerk zur Positionsevaluierung
  - Verwende erzeugte Netzwerke um mit MCTS zu spielen
    - Das langsame RL-Netzwerk macht die Ersteinschätzung der Züge

AlphaGo

- Kernidee: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS
- Trainere mehrere Netzwerke
  - Supervised auf Datensatz von besten Menschen: Schnell und Langsam
  - Verbessere das langsame Netz durch RL
  - Erzeuge Datensatz für Netzwerk zur Positionsevaluierung
  - Verwende erzeugte Netzwerke um mit MCTS zu spielen
    - Das langsame RL-Netzwerk macht die Ersteinschätzung der Züge
    - Einschätzung der Spielposition: Netzwerk + Rollouts

AlphaZero

- Drastische Vereinfachung von AlphaGo

AlphaZero

- Drastische Vereinfachung von AlphaGo
- Kernidee: Verwende MCTS bereits zur Trainingsphase

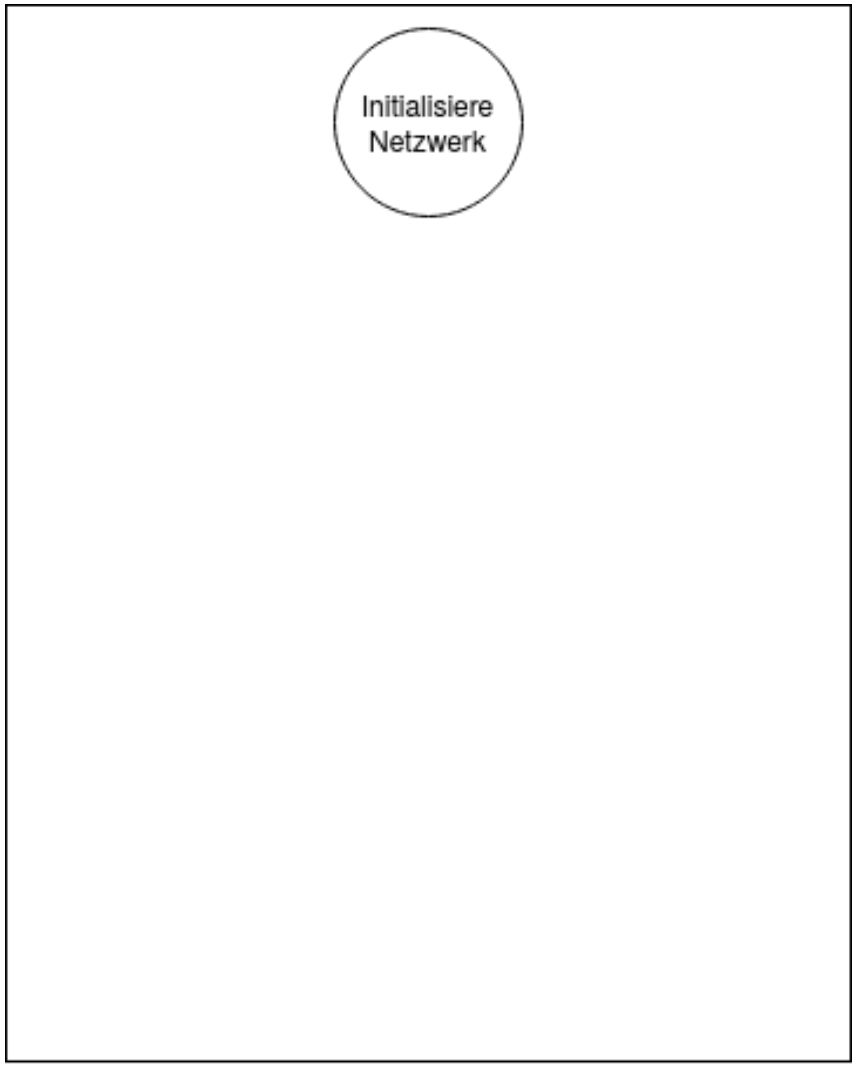
AlphaZero

- Drastische Vereinfachung von AlphaGo
- Kernidee: Verwende MCTS bereits zur Trainingsphase
- Verwendet nur ein Netzwerk: Positionsbewertung und Zugpolicy

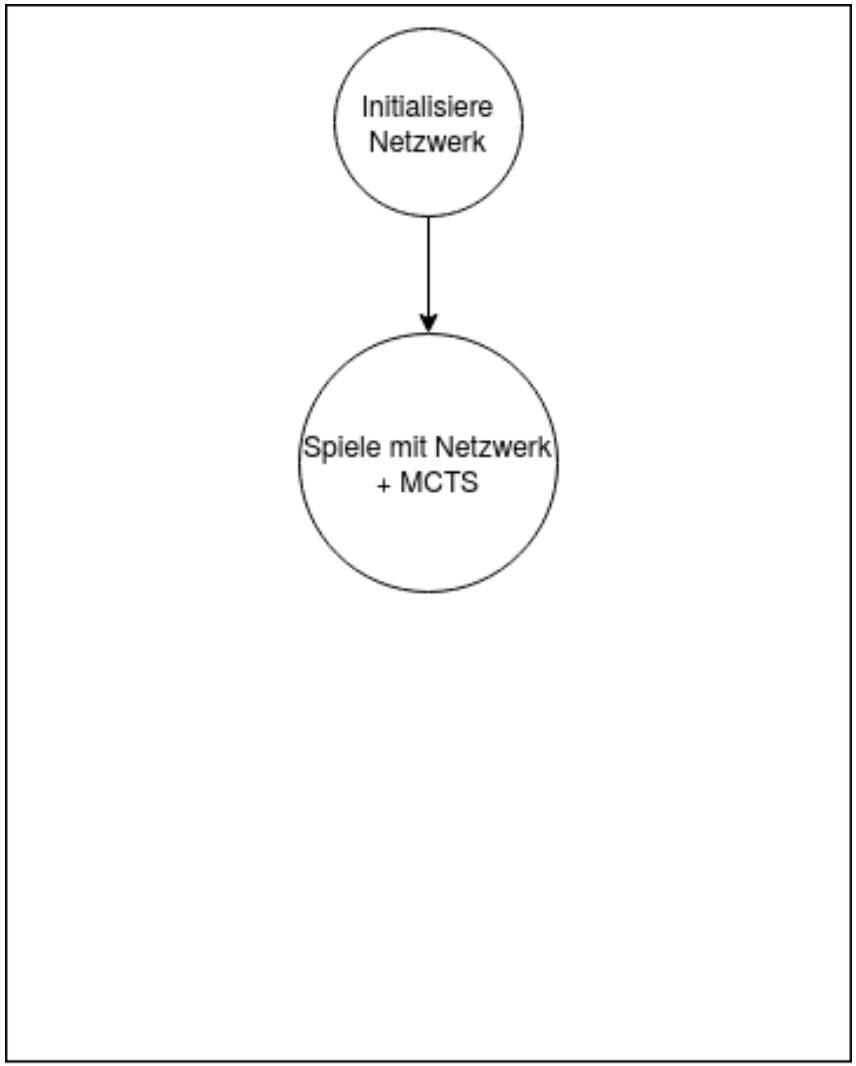
AlphaZero

- Drastische Vereinfachung von AlphaGo
- Kernidee: Verwende MCTS bereits zur Trainingsphase
- Verwendet nur ein Netzwerk: Positionsbewertung und Zugpolicy
- Kein Bedarf für Datensatz von besten Menschen

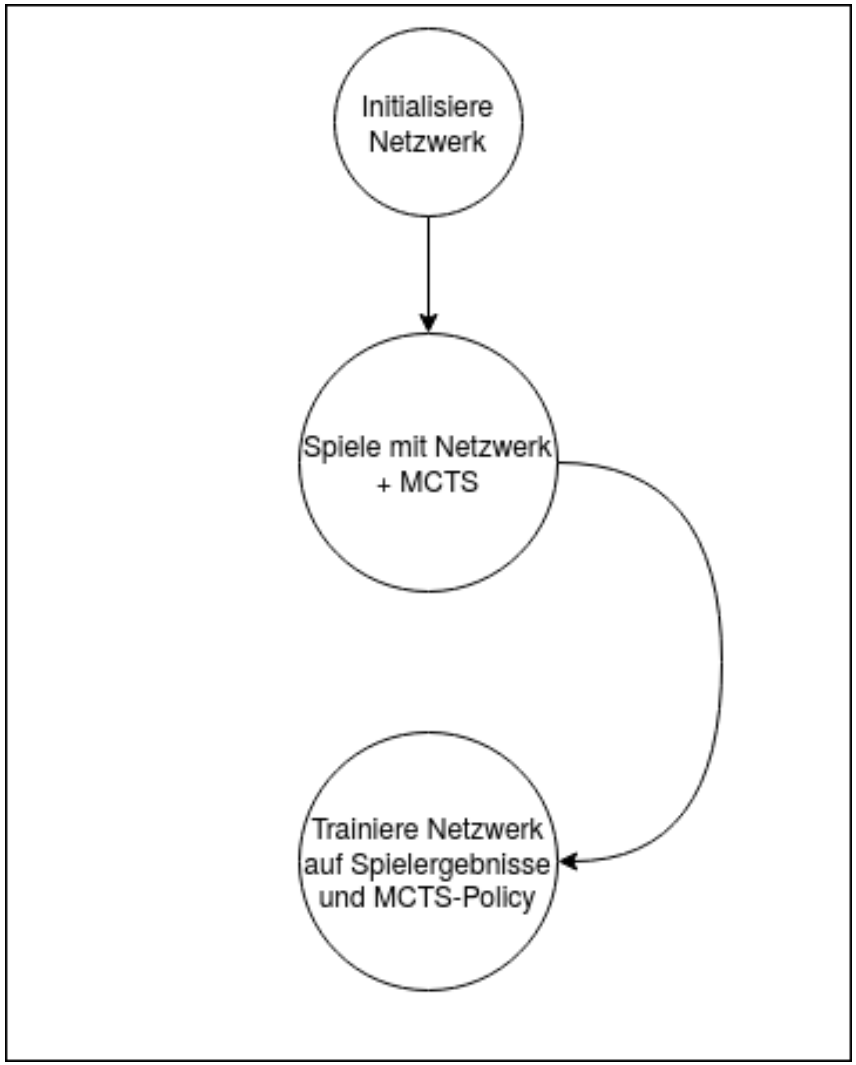
AlphaZero: Trainingsablauf



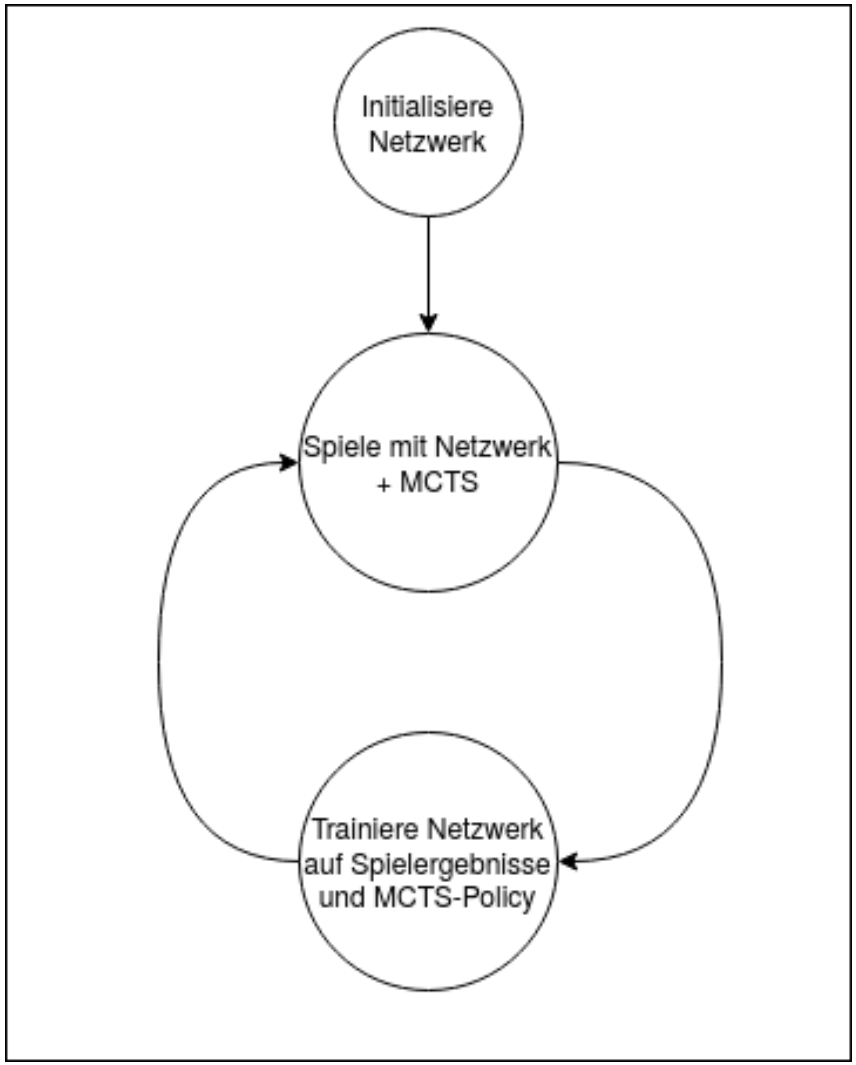
AlphaZero: Trainingsablauf



AlphaZero: Trainingsablauf



AlphaZero: Trainingsablauf



Extended baseline

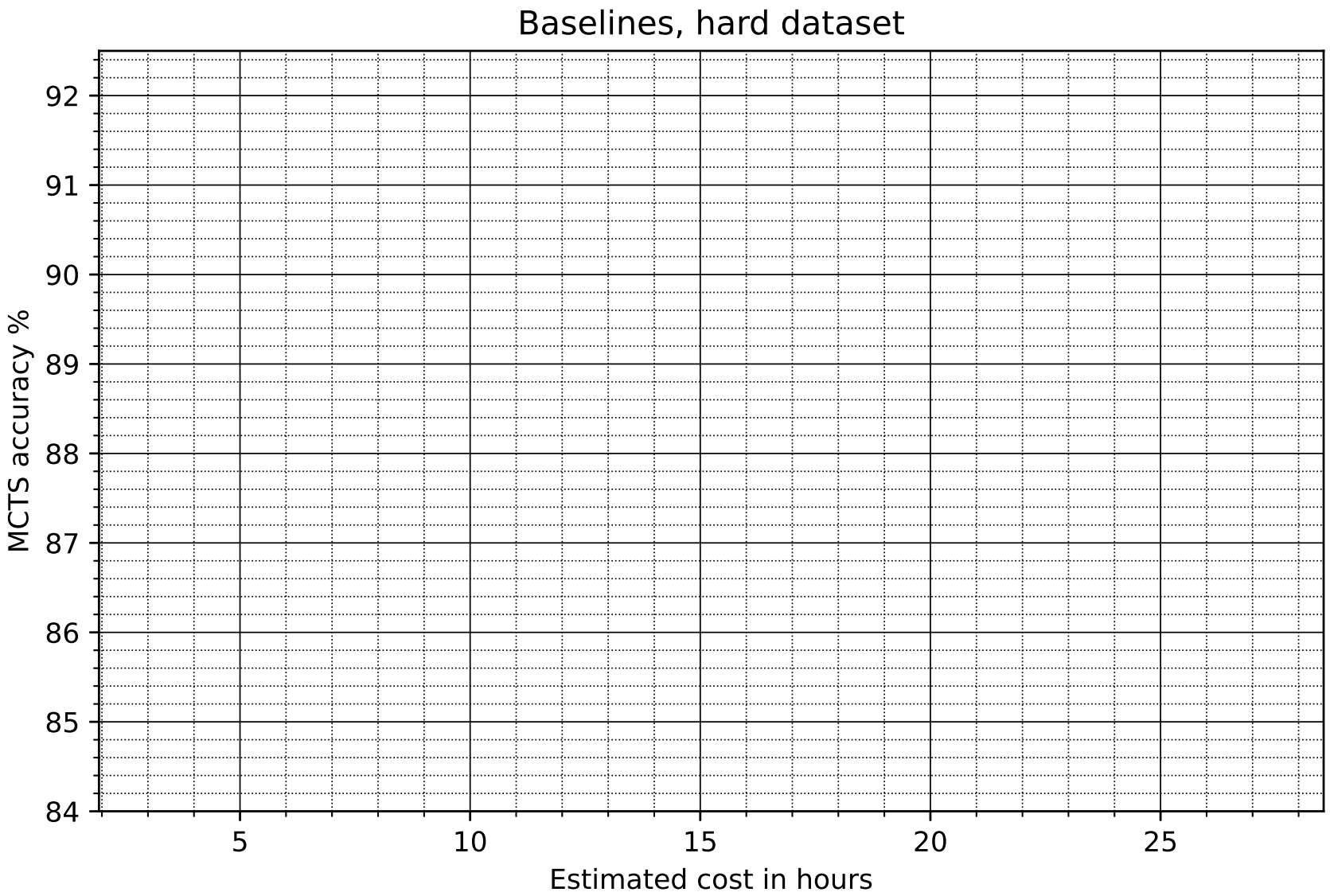
Extended baseline

- Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten

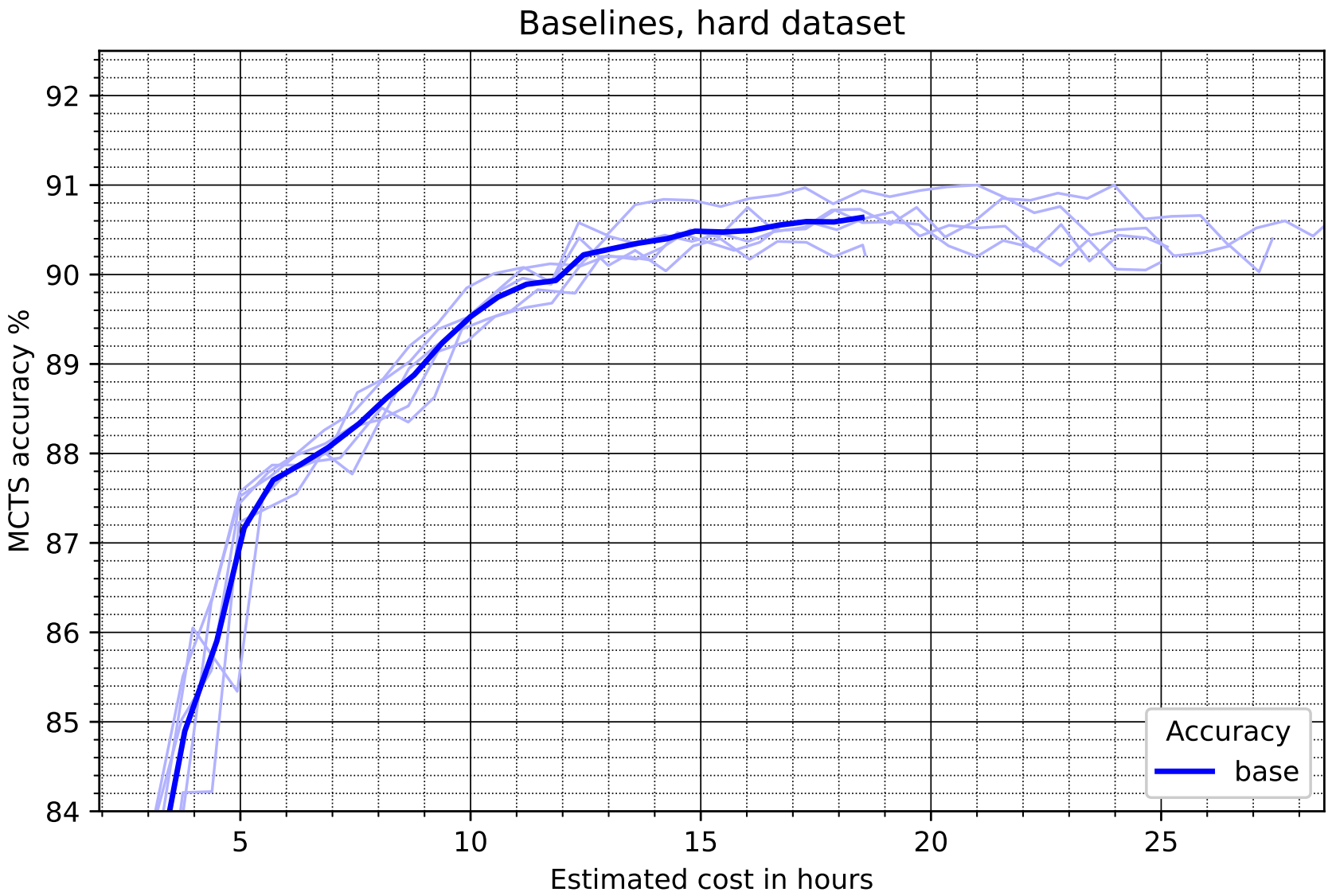
Extended baseline

- Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten
- Kombiniert ist die Verbesserung sehr erheblich

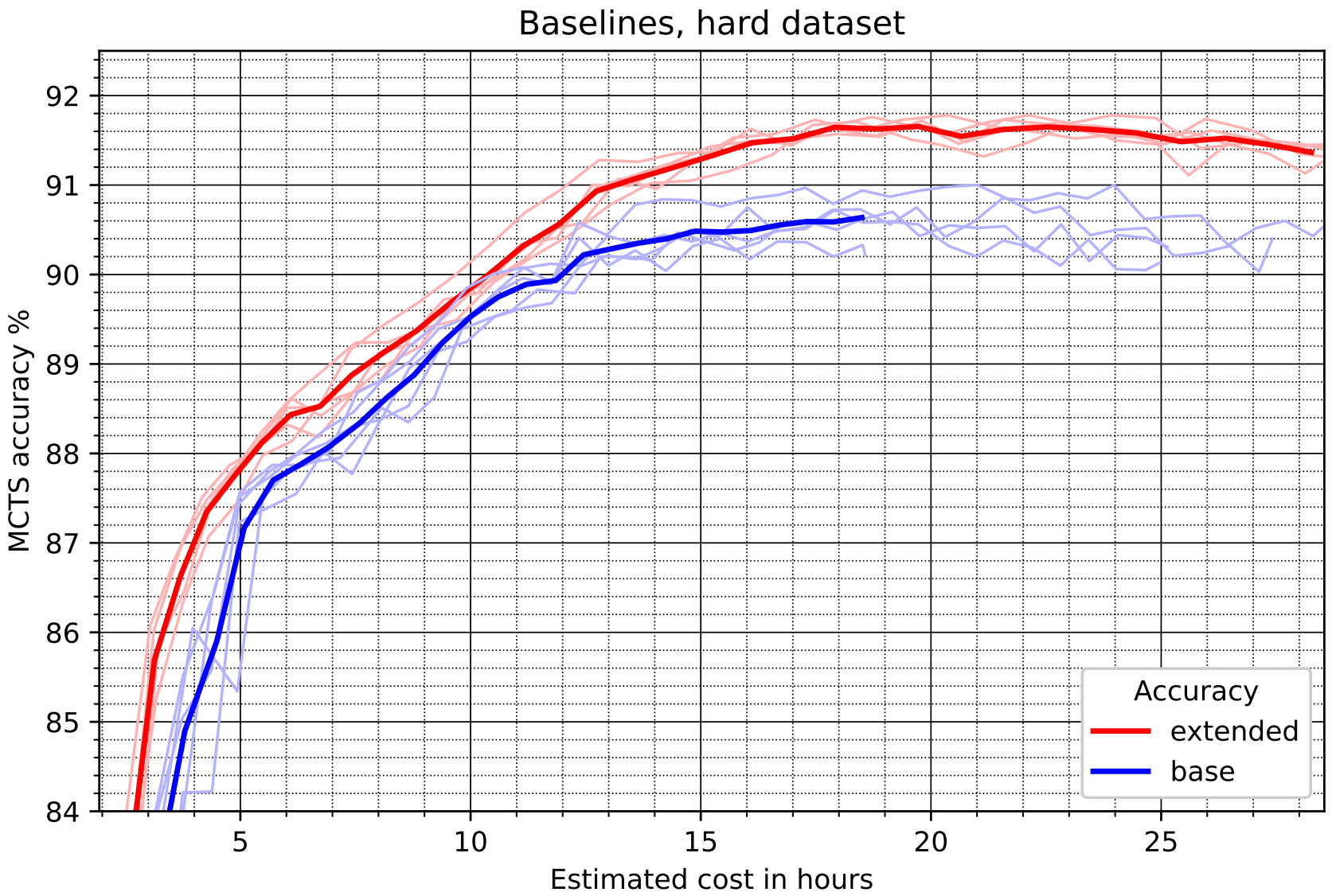
Extended baseline Ergebnisse



Extended baseline Ergebnisse



Extended baseline Ergebnisse



Inhalt

Implementierung: Evolutionary Self-play

Implementierung: Evolutionary Self-play

- Verwende die Selbstspielphase zur Evolution von Hyperparametern

Implementierung: Evolutionary Self-play

- Verwende die Selbstspielphase zur Evolution von Hyperparametern
- Implementiert als eine Liga von Spielern

Implementierung: Evolutionary Self-play

- Verwende die Selbstspielphase zur Evolution von Hyperparametern
- Implementiert als eine Liga von Spielern
- Ein Spieler ist ein Hyperparameterset

Implementierung: Evolutionary Self-play

- Verwende die Selbstspielphase zur Evolution von Hyperparametern
- Implementiert als eine Liga von Spielern
- Ein Spieler ist ein Hyperparameterset
- Bewerte Spieler mit Elo

Implementierung: Evolutionary Self-play

- Verwende die Selbstspielphase zur Evolution von Hyperparametern
- Implementiert als eine Liga von Spielern
- Ein Spieler ist ein Hyperparameterset
- Bewerte Spieler mit Elo
- Verwende Gaussian Mutation um die besten Spieler zu mutieren

Implementierung: Evolutionary Self-play

- Verwende die Selbstspielphase zur Evolution von Hyperparametern
- Implementiert als eine Liga von Spielern
- Ein Spieler ist ein Hyperparameterset
- Bewerte Spieler mit Elo
- Verwende Gaussian Mutation um die besten Spieler zu mutieren
- Zwei Arten von Hyperparametern untersucht

Implementierung: Evolutionary Self-play

- Verwende die Selbstspielphase zur Evolution von Hyperparametern
- Implementiert als eine Liga von Spielern
- Ein Spieler ist ein Hyperparameterset
- Bewerte Spieler mit Elo
- Verwende Gaussian Mutation um die besten Spieler zu mutieren

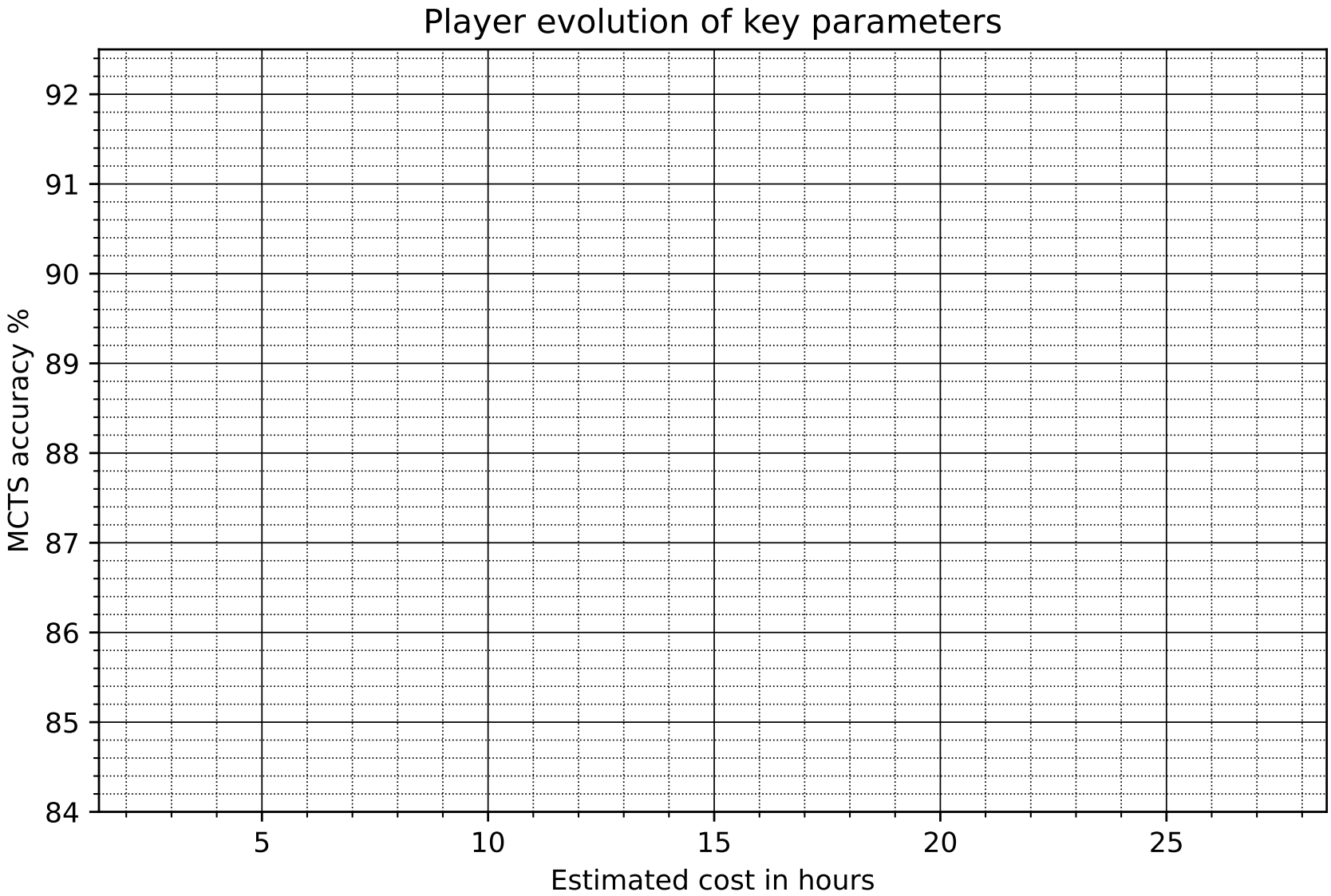
- Zwei Arten von Hyperparametern untersucht
  - Verwendung von Kullback-Leibler divergence um "Denkzeit" zu wählen.

Implementierung: Evolutionary Self-play

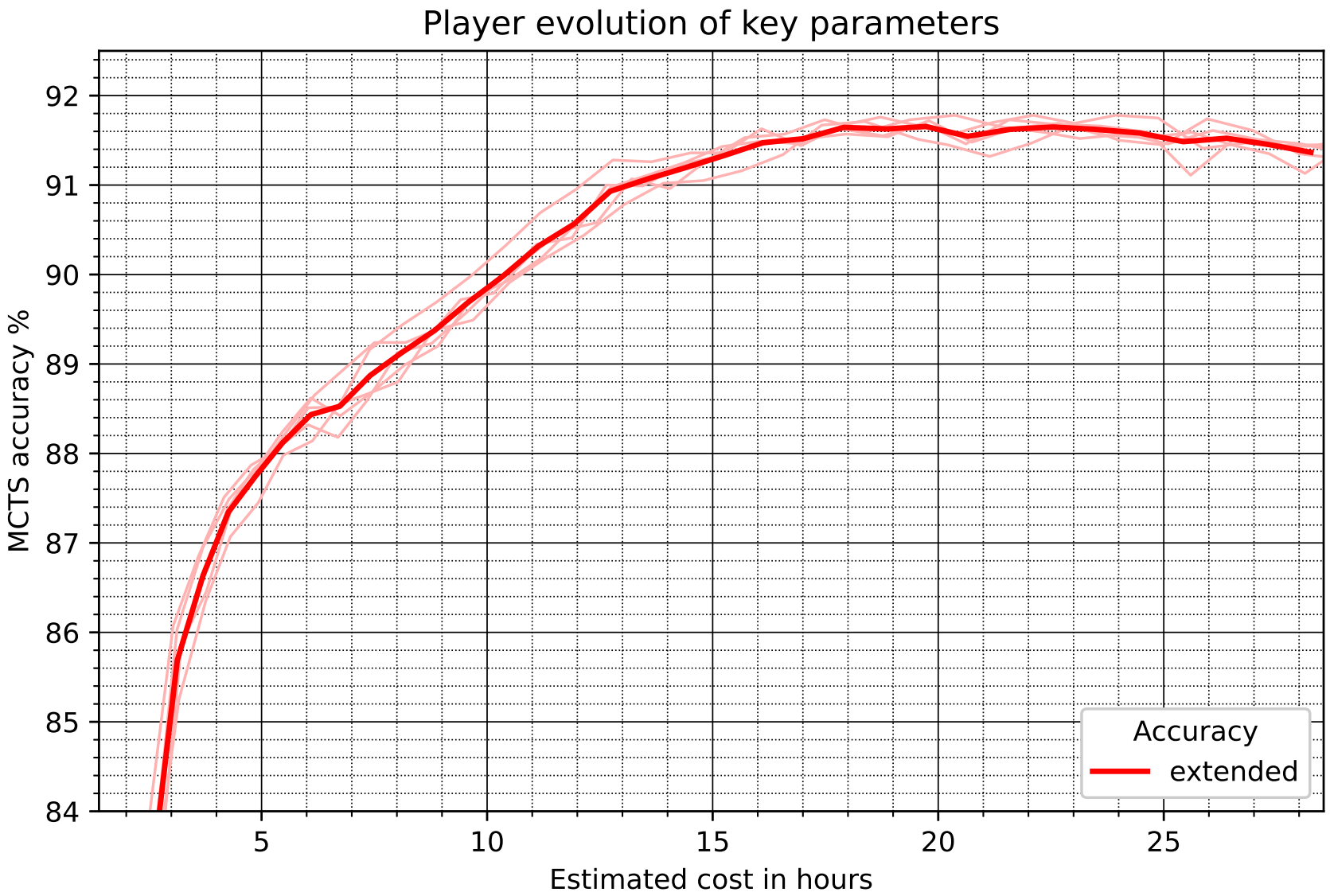
- Verwende die Selbstspielphase zur Evolution von Hyperparametern
- Implementiert als eine Liga von Spielern
- Ein Spieler ist ein Hyperparameterset
- Bewerte Spieler mit Elo
- Verwende Gaussian Mutation um die besten Spieler zu mutieren

- Zwei Arten von Hyperparametern untersucht
  - Verwendung von Kullback-Leibler divergence um "Denkzeit" zu wählen.
  - MCTS Parameter: cpuct, fpu, drawValue

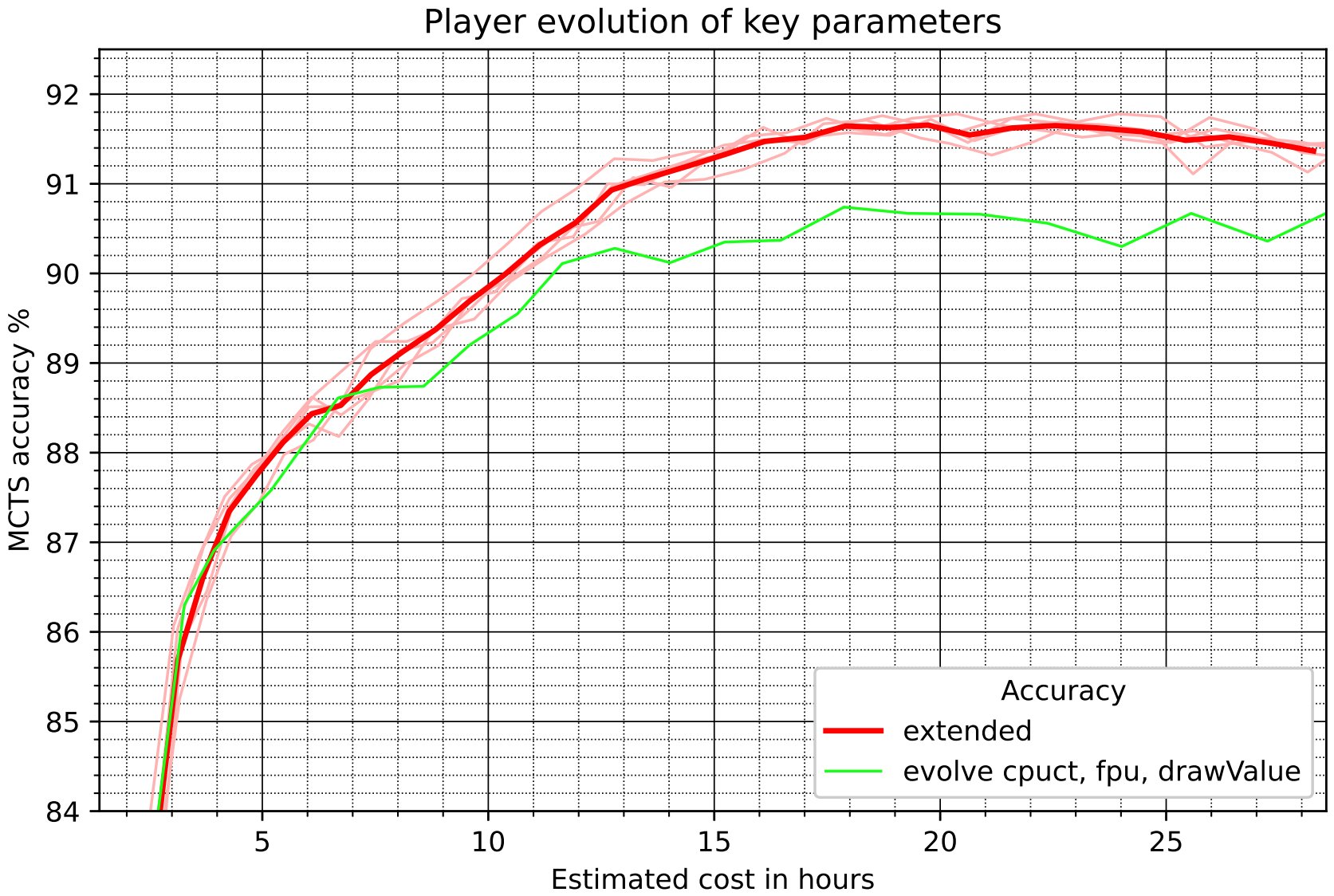
Erste Ergebnisse



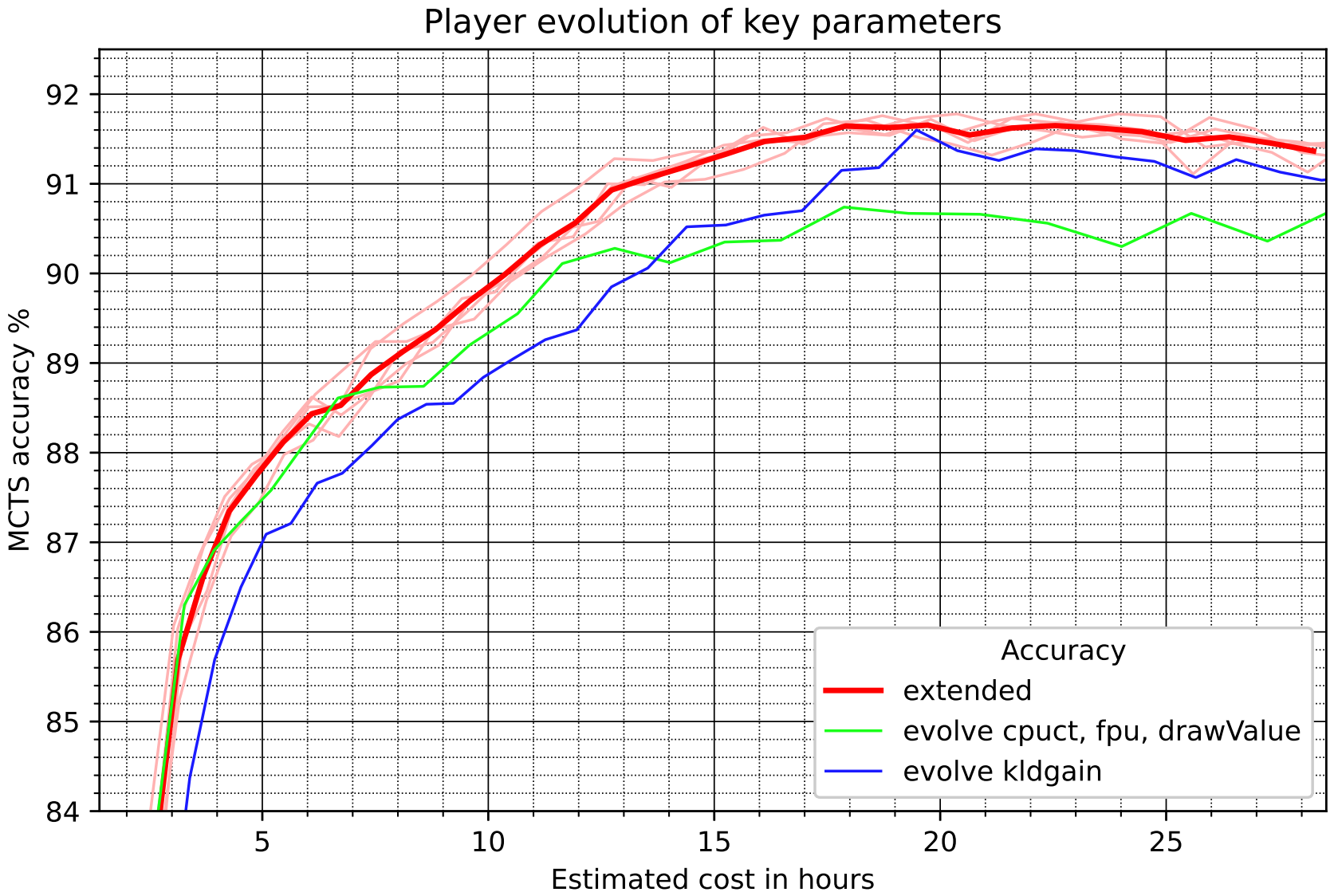
Erste Ergebnisse



Erste Ergebnisse



Erste Ergebnisse



Untersuchung

Untersuchung

- Bedingungen für erfolgreiche Evolution:

Untersuchung

- Bedingungen für erfolgreiche Evolution:
  - Die Liga muss gute Parameter erkennen

Untersuchung

- Bedingungen für erfolgreiche Evolution:
  - Die Liga muss gute Parameter erkennen
    - Sie funktioniert

Untersuchung

- Bedingungen für erfolgreiche Evolution:
  - Die Liga muss gute Parameter erkennen
    - Sie funktioniert
  - Viele Siege müssen sich übertragen auf schnelleren Lernfortschritt

Untersuchung

- Bedingungen für erfolgreiche Evolution:
  - Die Liga muss gute Parameter erkennen
    - Sie funktioniert
  - Viele Siege müssen sich übertragen auf schnelleren Lernfortschritt
    - Dies ist das Problem

Untersuchung

- Bedingungen für erfolgreiche Evolution:
  - Die Liga muss gute Parameter erkennen
    - Sie funktioniert
  - Viele Siege müssen sich übertragen auf schnelleren Lernfortschritt
    - Dies ist das Problem
- Viele gewonne Spiele bedeuten also nicht hoher Lernfortschritt

Untersuchung

- Bedingungen für erfolgreiche Evolution:
  - Die Liga muss gute Parameter erkennen
    - Sie funktioniert
  - Viele Siege müssen sich übertragen auf schnelleren Lernfortschritt
    - Dies ist das Problem
- Viele gewonne Spiele bedeuten also nicht hoher Lernfortschritt
- Nach hoher Siegesrate zu optimieren ist also nicht zielführend

Implementierung: Games as trees

Implementierung: Games as trees

- Exploration durch Zurücksetzen an kritische Positionen

Implementierung: Games as trees

- Exploration durch Zurücksetzen an kritische Positionen
- Spiele als MCTS-Baum

Implementierung: Games as trees

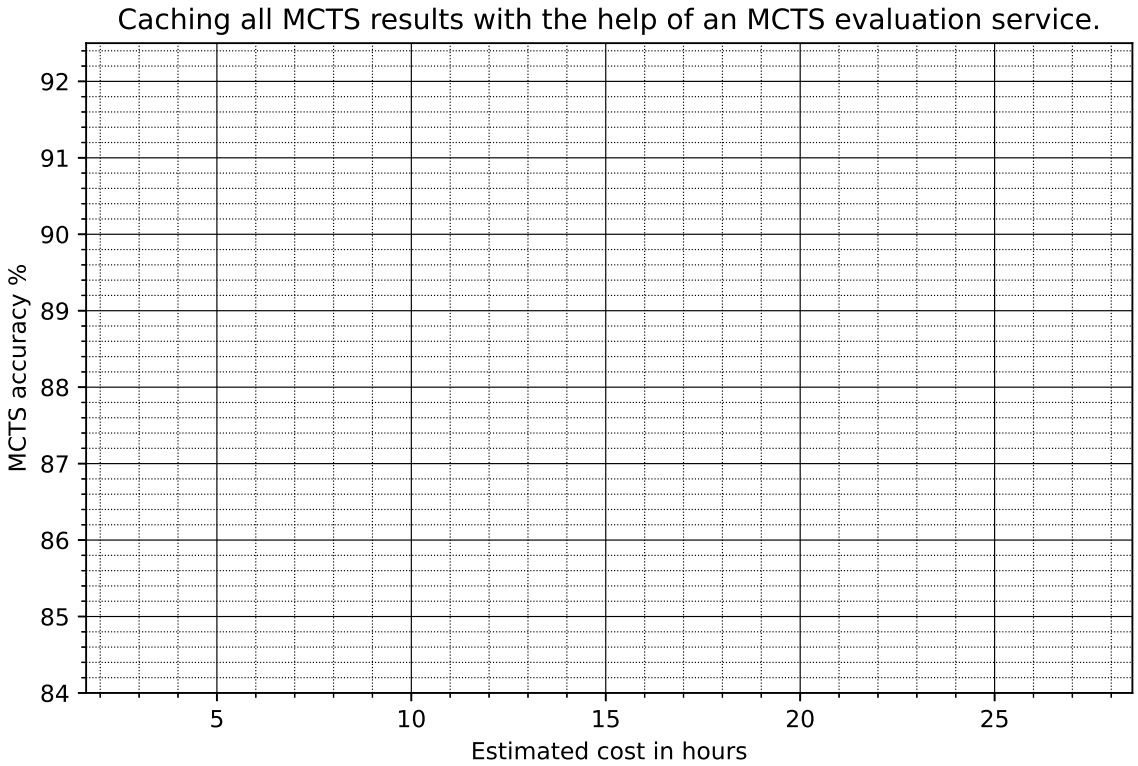
- Exploration durch Zurücksetzen an kritische Positionen
- Spiele als MCTS-Baum
- Notwendigkeit für MCTS-Evaluation Service

Implementierung: Games as trees

- Exploration durch Zurücksetzen an kritische Positionen
- Spiele als MCTS-Baum
- Notwendigkeit für MCTS-Evaluation Service
- Nebeneffekt: Keine doppelte Auswertung von Positionen

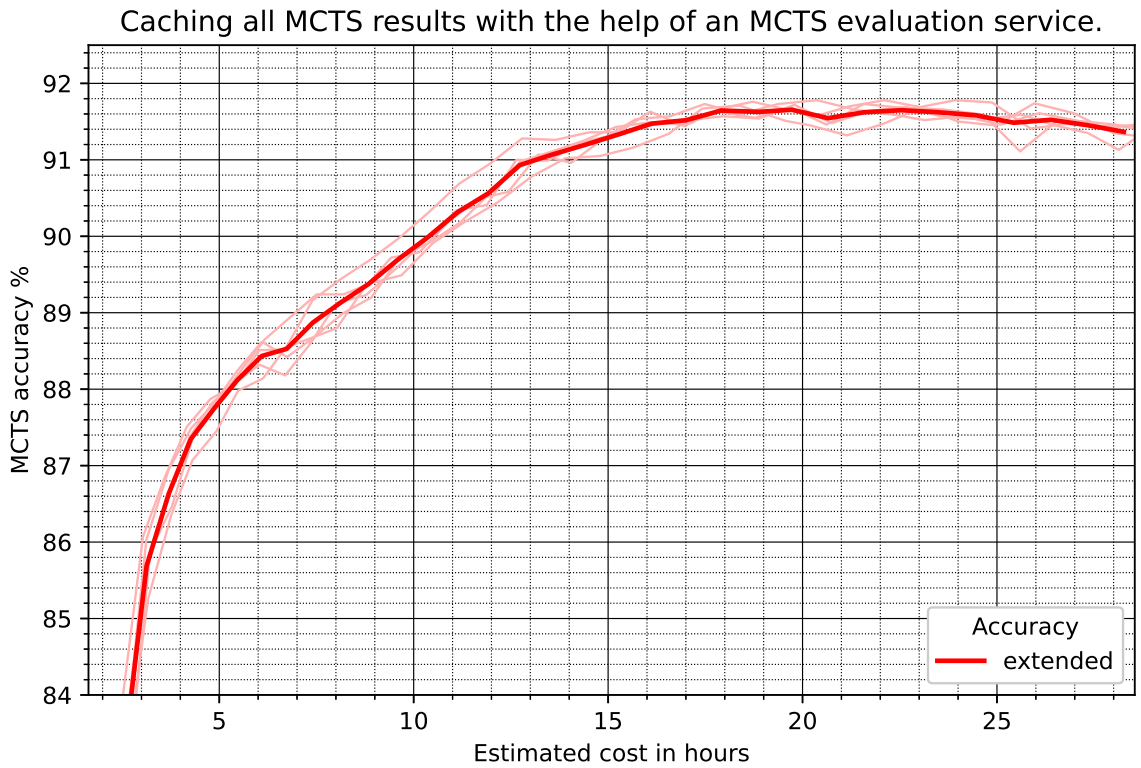
Implementierung: Games as trees

- Exploration durch Zurücksetzen an kritische Positionen
- Spiele als MCTS-Baum
- Notwendigkeit für MCTS-Evaluation Service
- Nebeneffekt: Keine doppelte Auswertung von Positionen



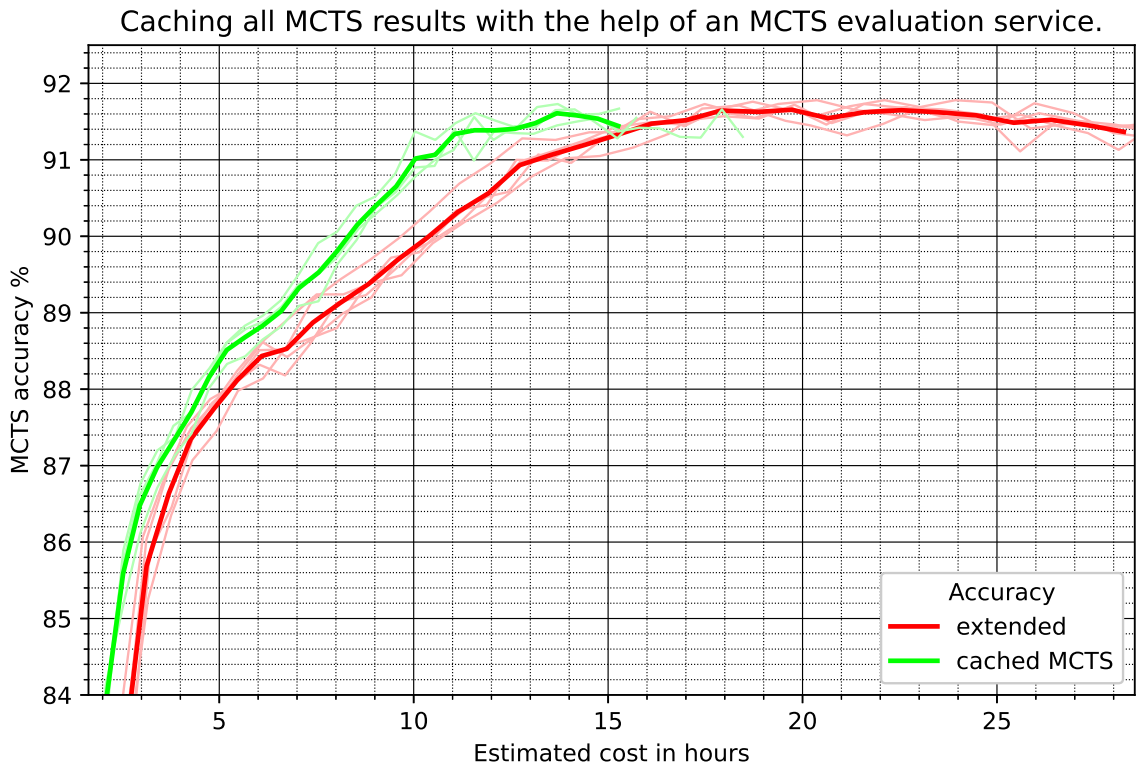
Implementierung: Games as trees

- Exploration durch Zurücksetzen an kritische Positionen
- Spiele als MCTS-Baum
- Notwendigkeit für MCTS-Evaluation Service
- Nebeneffekt: Keine doppelte Auswertung von Positionen



Implementierung: Games as trees

- Exploration durch Zurücksetzen an kritische Positionen
- Spiele als MCTS-Baum
- Notwendigkeit für MCTS-Evaluation Service
- Nebeneffekt: Keine doppelte Auswertung von Positionen



Zurücksetzen auf kritische Position

Zurücksetzen auf kritische Position

- Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen

Zurücksetzen auf kritische Position

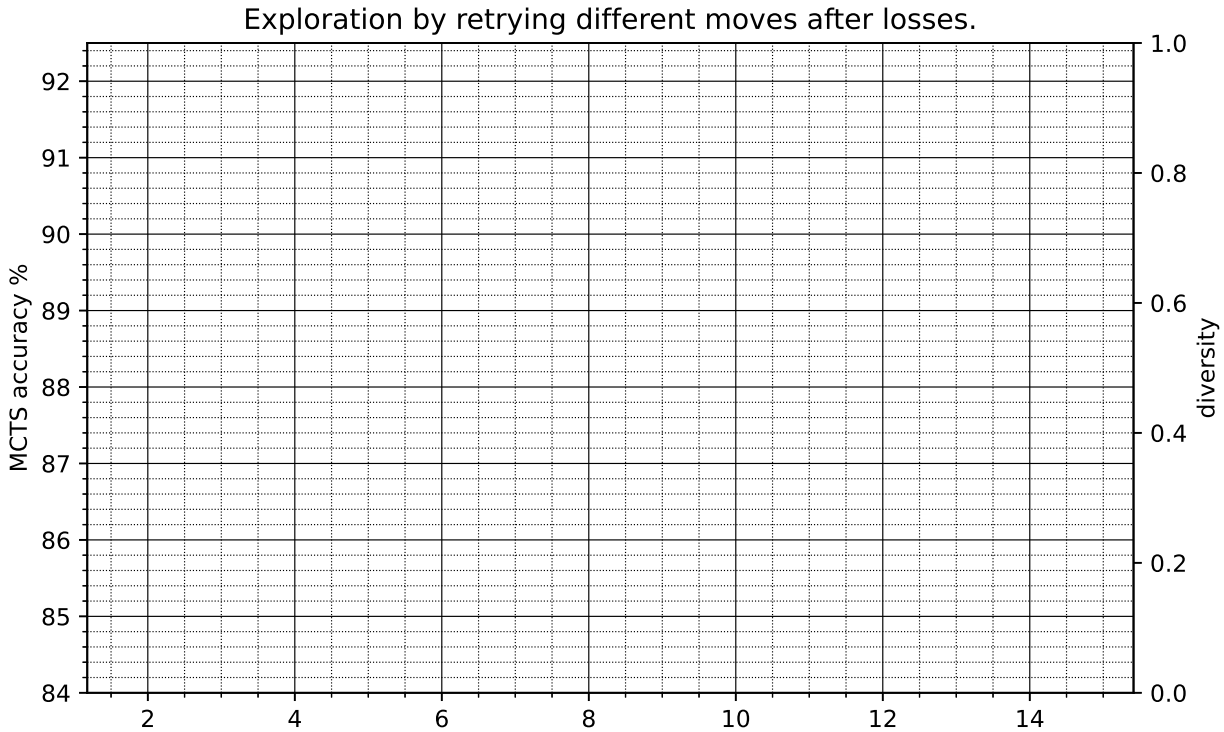
- Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen
- Wähle Position anhand der Entwicklung der Positionsevaluation

Zurücksetzen auf kritische Position

- Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen
- Wähle Position anhand der Entwicklung der Positionsevaluation
- Beginne neues Spiel in dieser Position

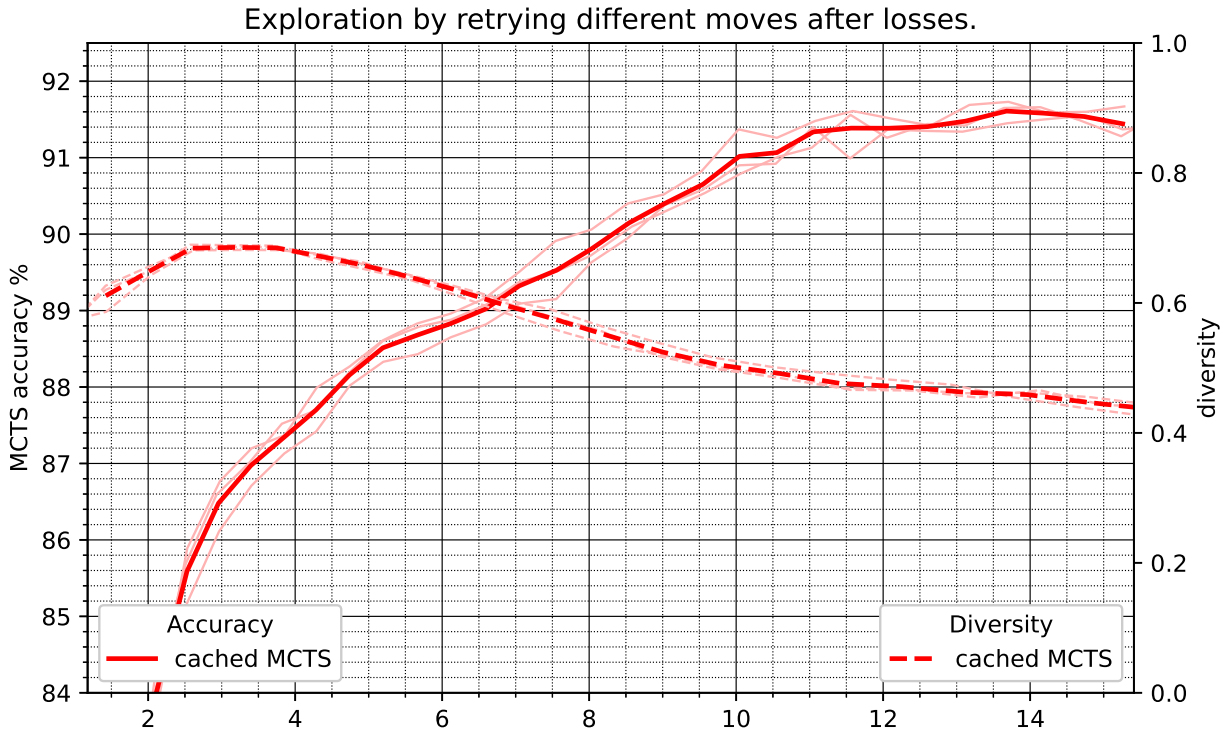
Zurücksetzen auf kritische Position

- Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen
- Wähle Position anhand der Entwicklung der Positionsevaluation
- Beginne neues Spiel in dieser Position



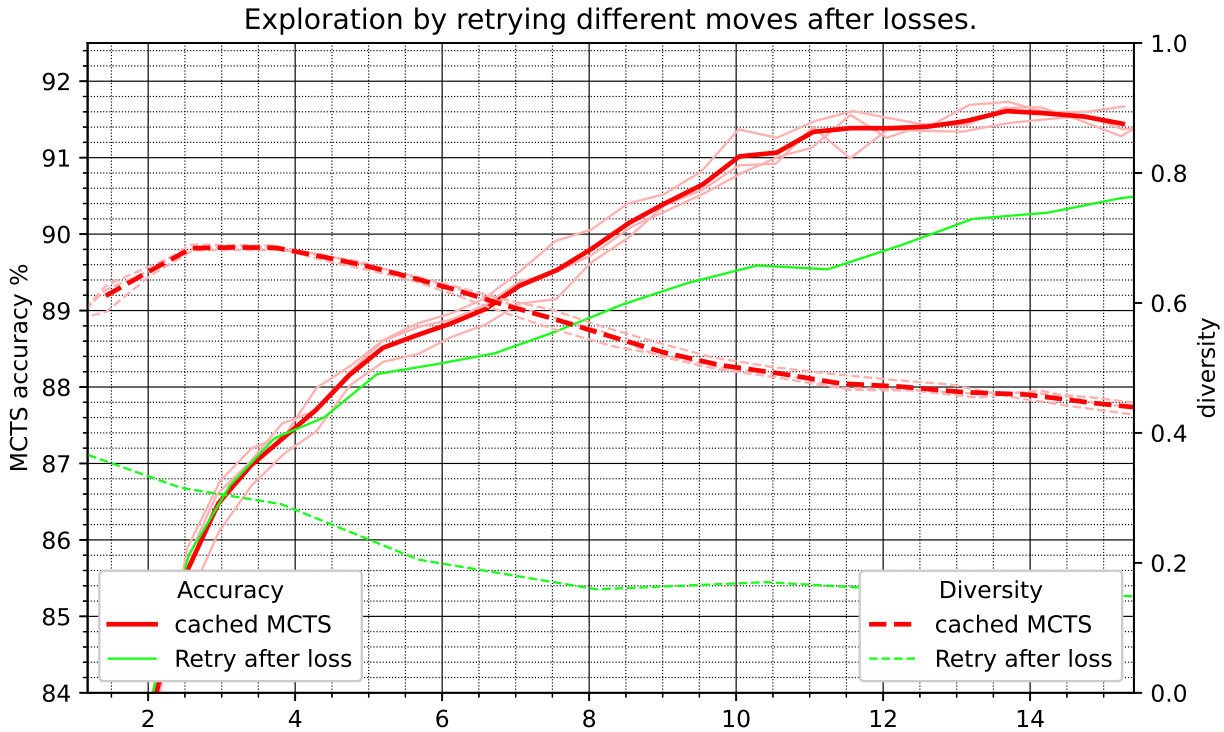
Zurücksetzen auf kritische Position

- Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen
- Wähle Position anhand der Entwicklung der Positionsevaluation
- Beginne neues Spiel in dieser Position



Zurücksetzen auf kritische Position

- Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen
- Wähle Position anhand der Entwicklung der Positionsevaluation
- Beginne neues Spiel in dieser Position



Spiele als MCTS-Baum

Spiele als MCTS-Baum

- Exploration-Exploitation: MCTS macht das

Spiele als MCTS-Baum

- Exploration-Exploitation: MCTS macht das
- Baue einen einzigen MCTS Baum

Spiele als MCTS-Baum

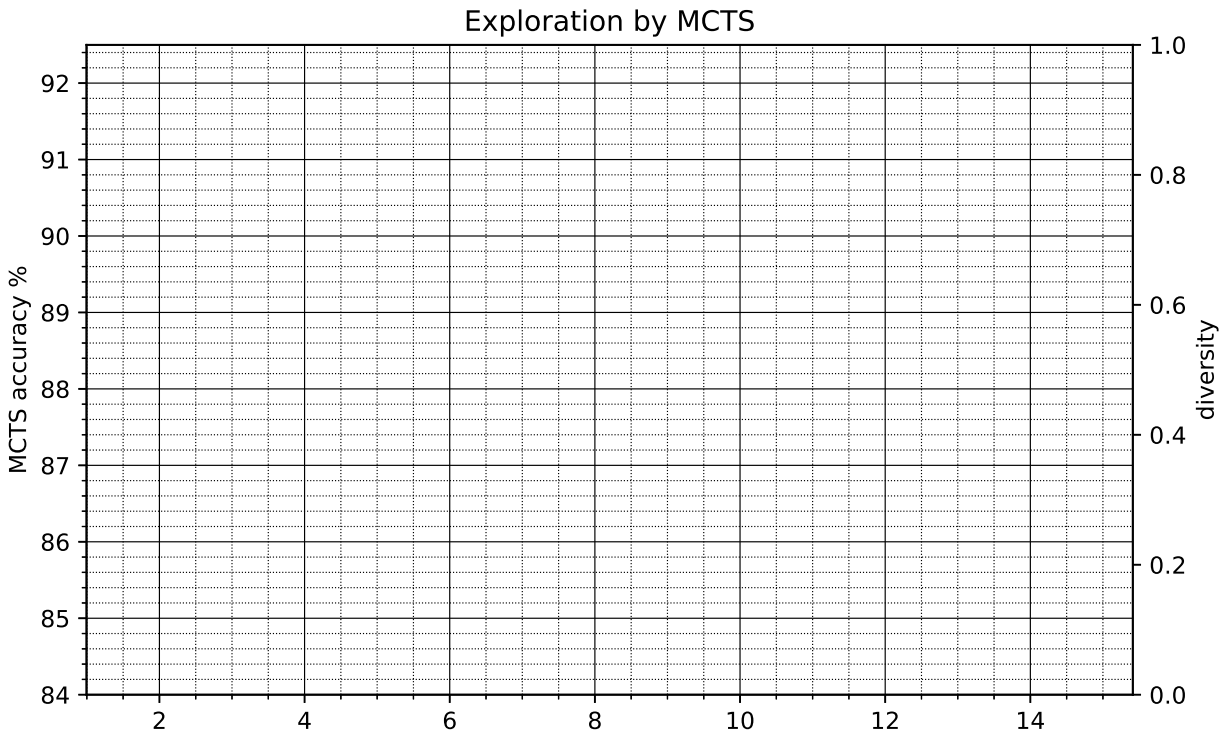
- Exploration-Exploitation: MCTS macht das
- Baue einen einzigen MCTS Baum
  - 150k+ Knoten

Spiele als MCTS-Baum

- Exploration-Exploitation: MCTS macht das
- Baue einen einzigen MCTS Baum
  - 150k+ Knoten
- Reporte die Positionen in den Knoten als Trainingspositionen

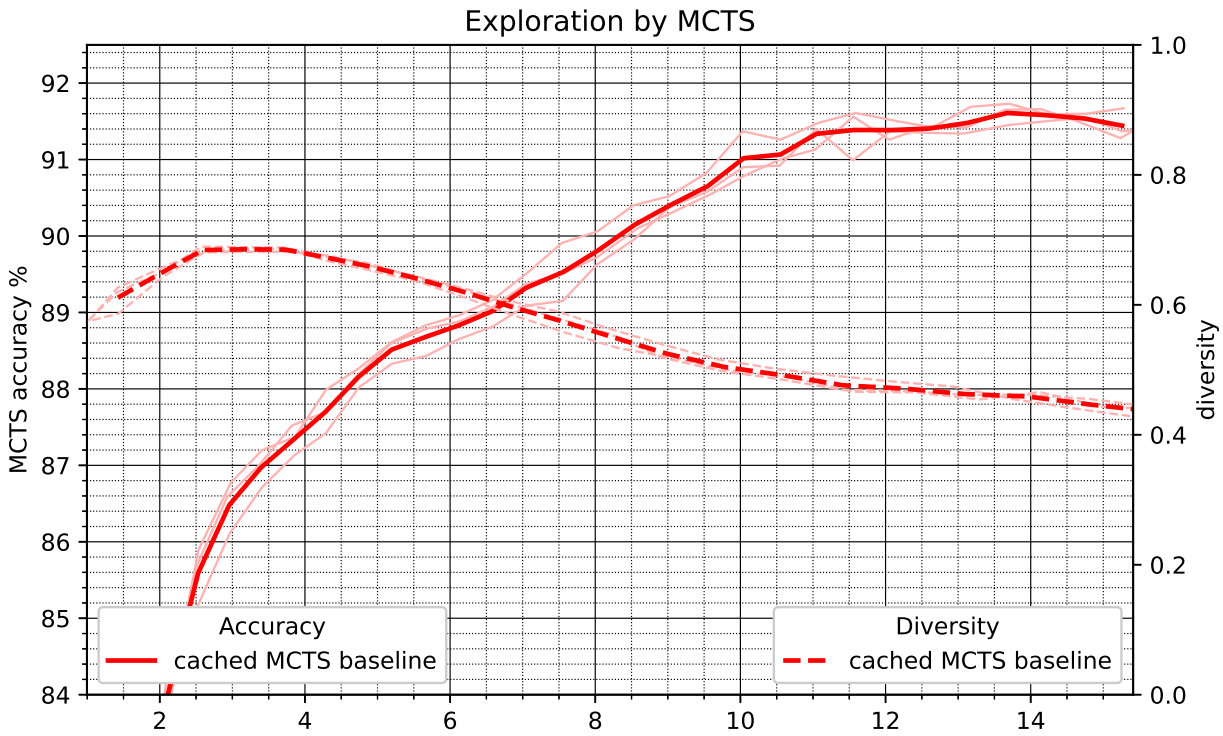
Spiele als MCTS-Baum

- Exploration-Exploitation: MCTS macht das
- Baue einen einzigen MCTS Baum
  - 150k+ Knoten
- Reporte die Positionen in den Knoten als Trainingspositionen



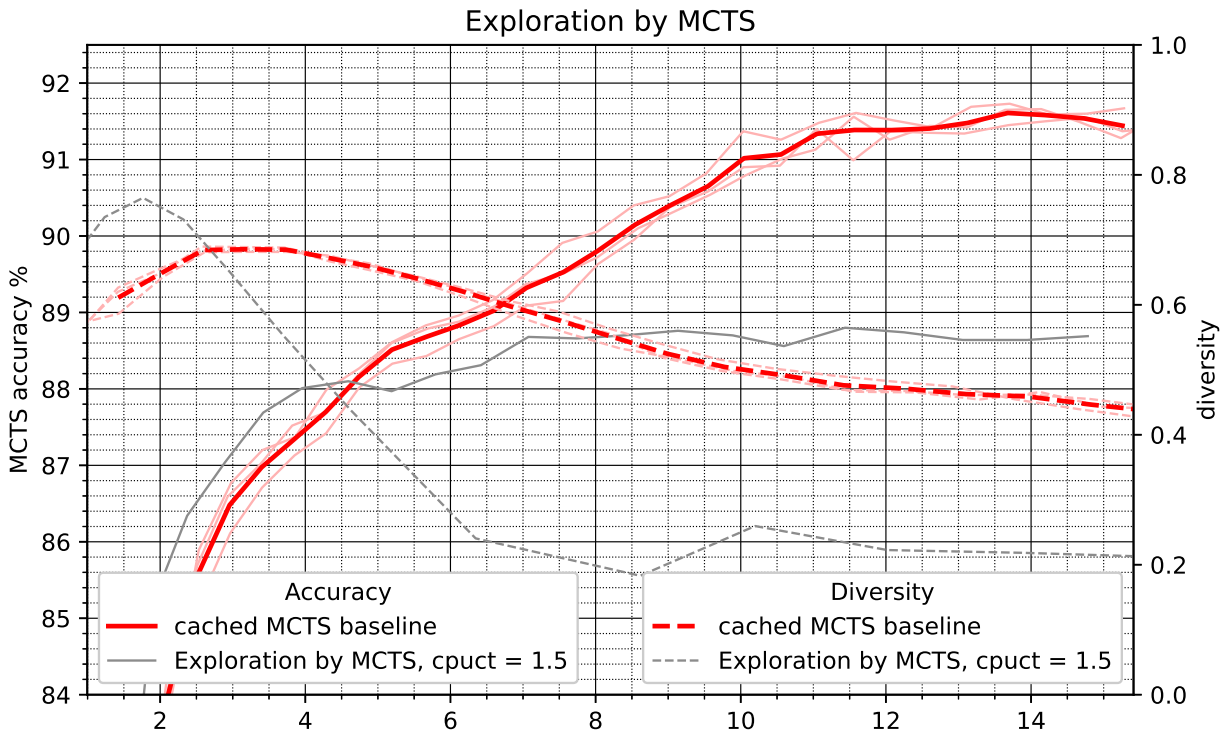
Spiele als MCTS-Baum

- Exploration-Exploitation: MCTS macht das
- Baue einen einzigen MCTS Baum
  - 150k+ Knoten
- Reporte die Positionen in den Knoten als Trainingspositionen



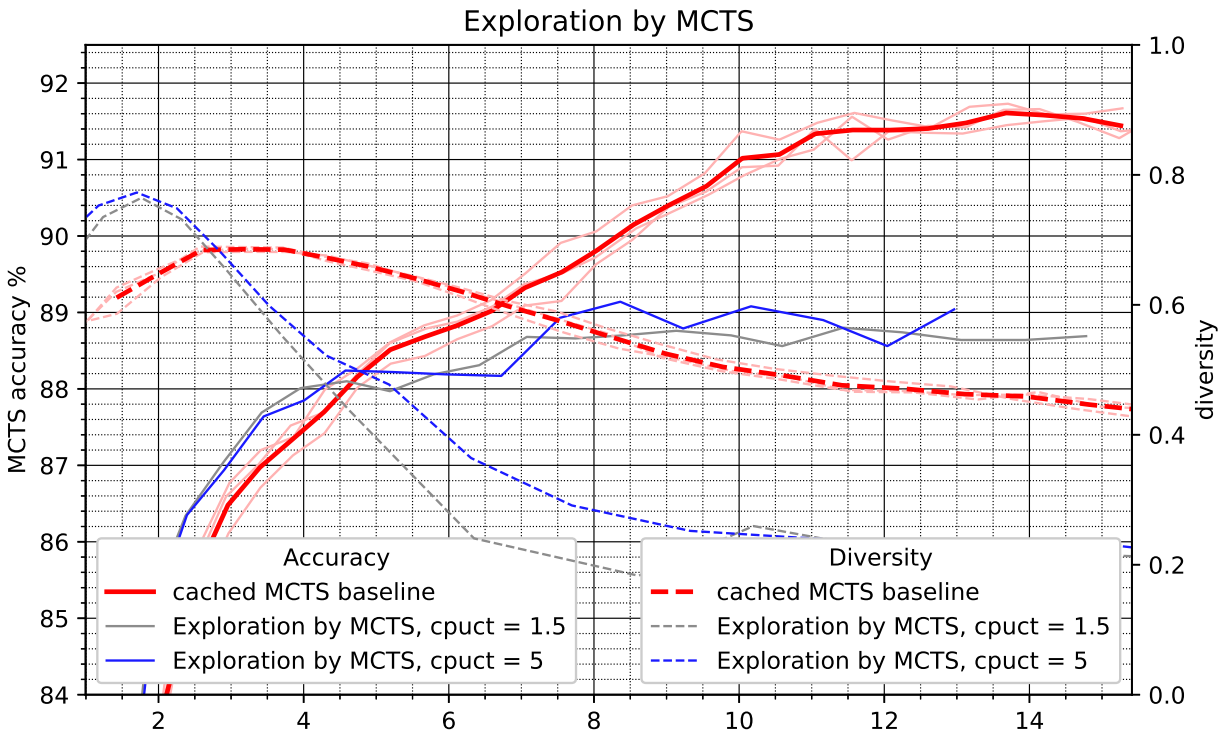
Spiele als MCTS-Baum

- Exploration-Exploitation: MCTS macht das
- Baue einen einzigen MCTS Baum
  - 150k+ Knoten
- Reporte die Positionen in den Knoten als Trainingspositionen



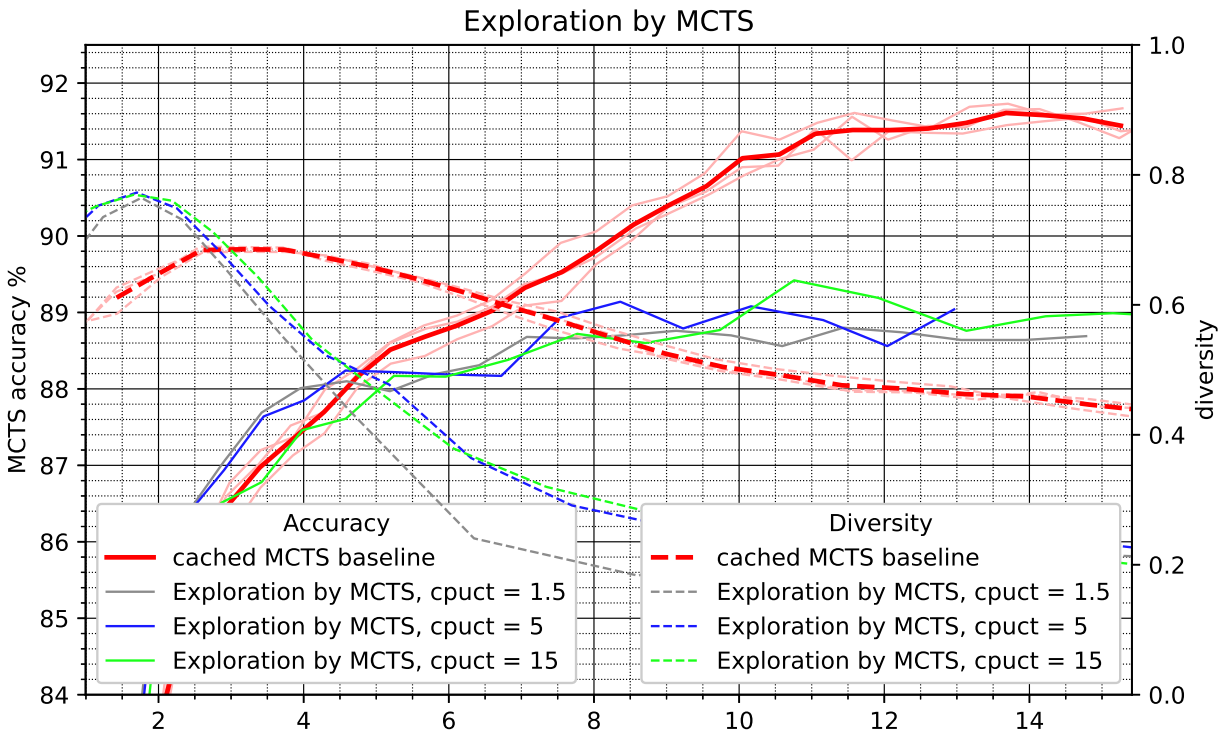
Spiele als MCTS-Baum

- Exploration-Exploitation: MCTS macht das
- Baue einen einzigen MCTS Baum
  - 150k+ Knoten
- Reporte die Positionen in den Knoten als Trainingspositionen



Spiele als MCTS-Baum

- Exploration-Exploitation: MCTS macht das
- Baue einen einzigen MCTS Baum
  - 150k+ Knoten
- Reporte die Positionen in den Knoten als Trainingspositionen



Implementierung: Auxiliary features

Implementierung: Auxiliary features

- Trainiere ein kleines Netzwerk, ca. 70k Parameter

Implementierung: Auxiliary features

- Trainiere ein kleines Netzwerk, ca. 70k Parameter
- Verwende interne Features aus diesem Netzwerk zur Regularisierung des großen Netzwerkes

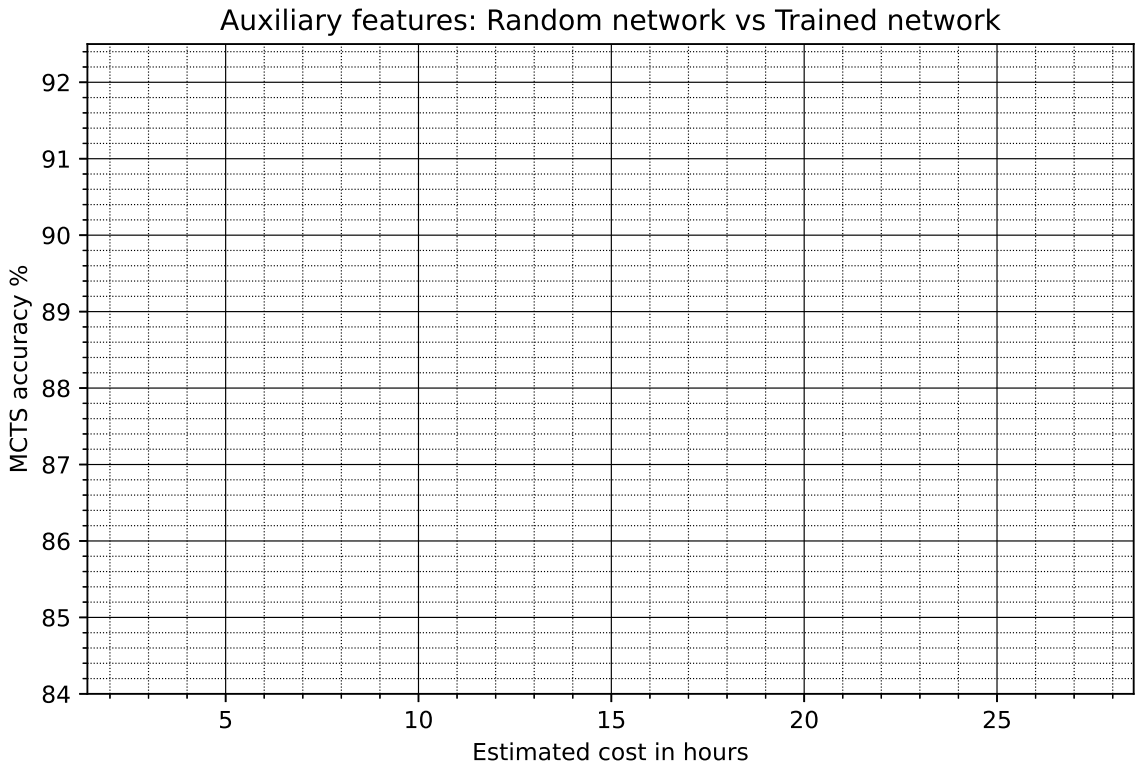
Implementierung: Auxiliary features

- Trainiere ein kleines Netzwerk, ca. 70k Parameter
- Verwende interne Features aus diesem Netzwerk zur Regularisierung des großen Netzwerkes
- Verschiedene Optionen wurden im Supervised Setting vorselektiert

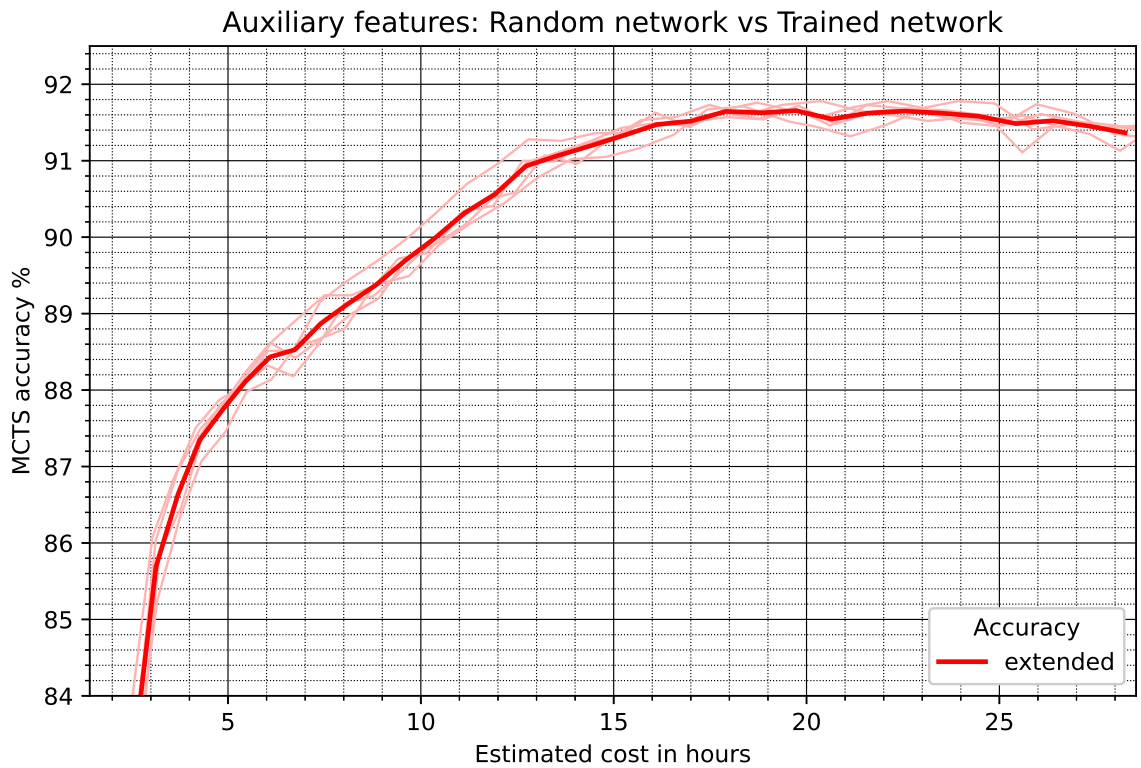
Implementierung: Auxiliary features

- Trainiere ein kleines Netzwerk, ca. 70k Parameter
- Verwende interne Features aus diesem Netzwerk zur Regularisierung des großen Netzwerkes
- Verschiedene Optionen wurden im Supervised Setting vorselektiert
- Kleine Gewinne im Supervised Setting

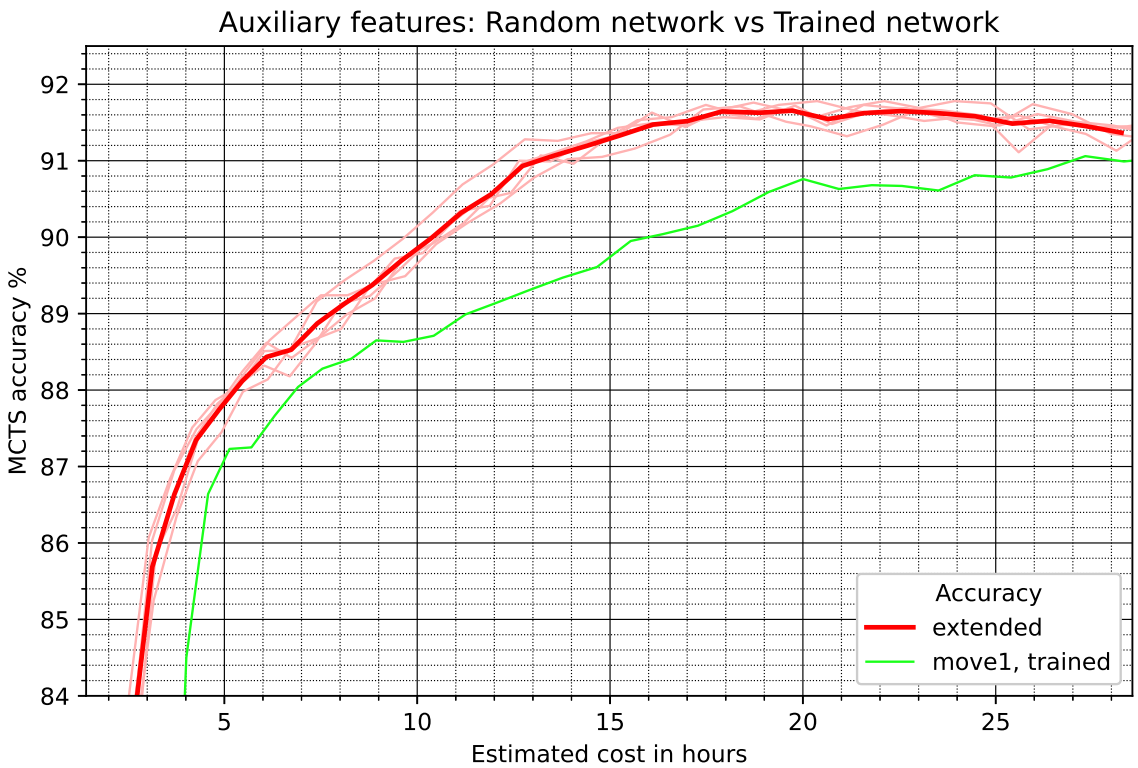
Untersuchung



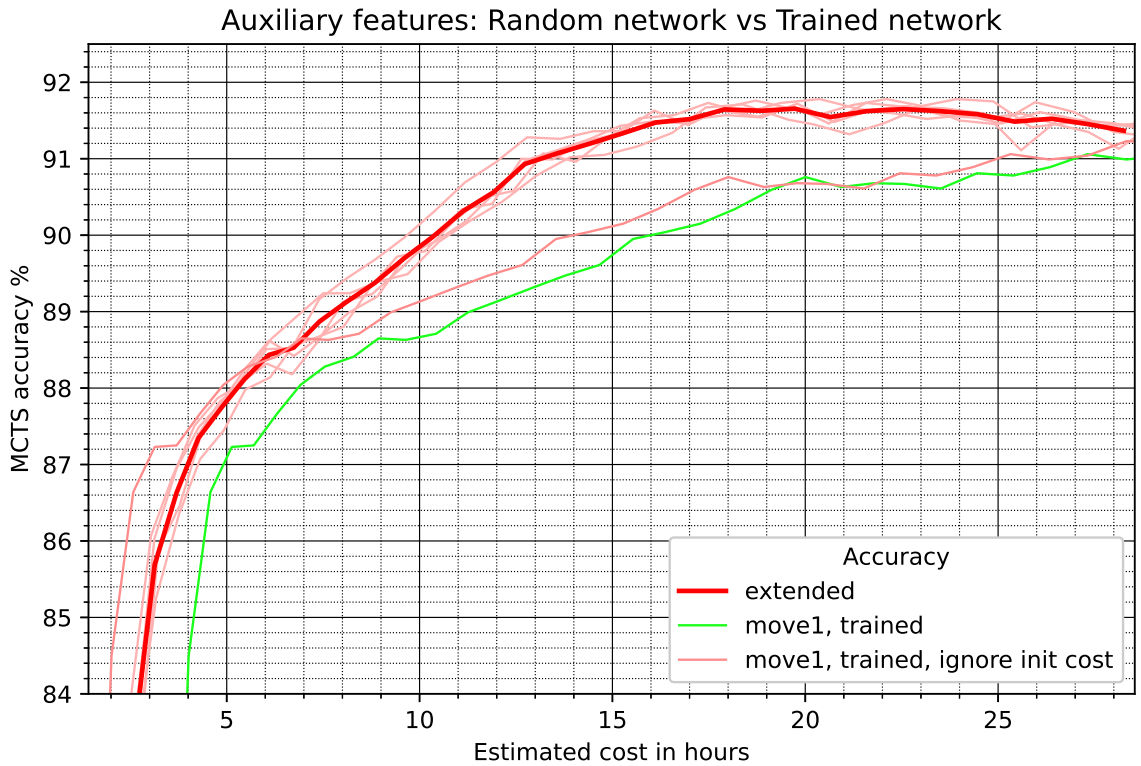
Untersuchung



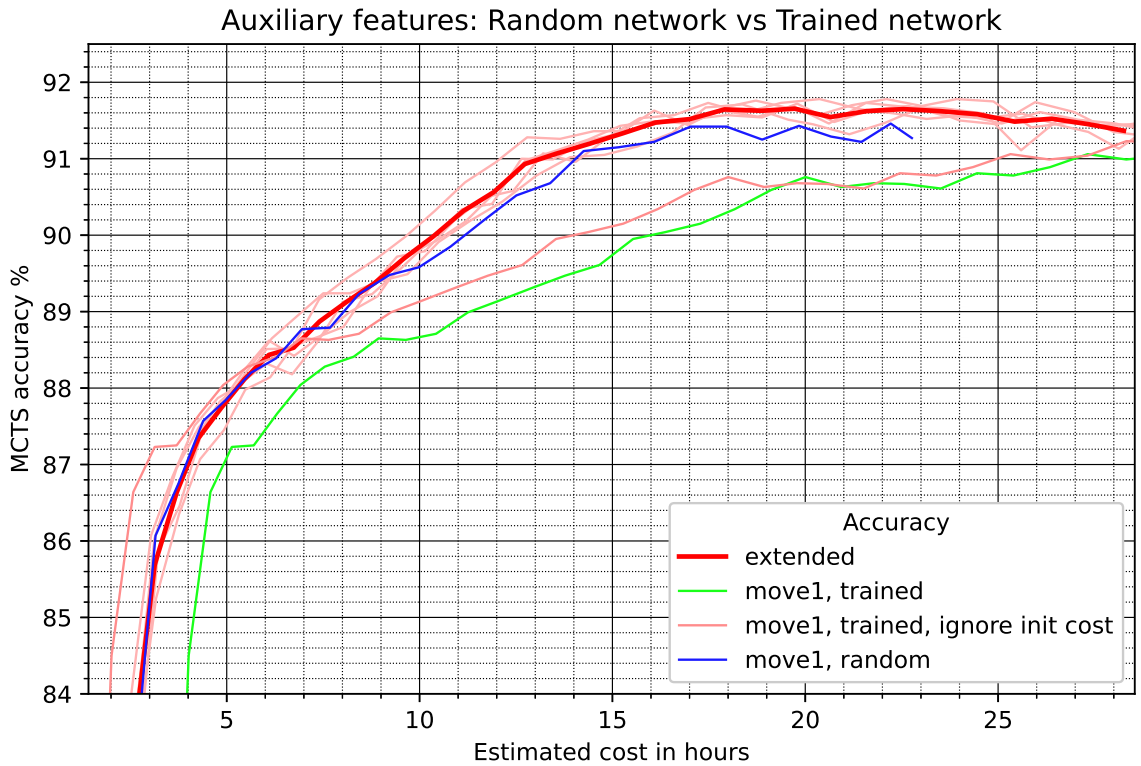
Untersuchung



Untersuchung



Untersuchung



Probleme

Probleme

- Zwei Probleme mit dem Ansatz:

Probleme

- Zwei Probleme mit dem Ansatz:
  - Trainingskosten des kleinen Netzwerks

Probleme

- Zwei Probleme mit dem Ansatz:
  - Trainingskosten des kleinen Netzwerks
    - Das Netzwerk mit dem Trainingslauf wachsen lassen hilft

Probleme

- Zwei Probleme mit dem Ansatz:
  - Trainingskosten des kleinen Netzwerks
    - Das Netzwerk mit dem Trainingslauf wachsen lassen hilft
  - Finale Spielstärke wird gestört

Probleme

- Zwei Probleme mit dem Ansatz:
  - Trainingskosten des kleinen Netzwerks
    - Das Netzwerk mit dem Trainingslauf wachsen lassen hilft
  - Finale Spielstärke wird gestört
    - Keine Lösung gefunden

Inhalt

Fazit

Fazit

- AlphaZero Experimentalframework entwickelt

Fazit

- AlphaZero Experimentalframework entwickelt
- Keine großen Verbesserungen gefunden

Fazit

- AlphaZero Experimentalframework entwickelt
- Keine großen Verbesserungen gefunden
- Trotzdem einiges interessante Erkenntnisse

Fazit

- AlphaZero Experimentalframework entwickelt
- Keine großen Verbesserungen gefunden
- Trotzdem einiges interessante Erkenntnisse
  - Evolution für Hyperparameter funktioniert

Fazit

- AlphaZero Experimentalframework entwickelt
- Keine großen Verbesserungen gefunden
- Trotzdem einiges interessante Erkenntnisse
  - Evolution für Hyperparameter funktioniert
    - Nur eine gute Fitnessfunktion fehlt

Fazit

- AlphaZero Experimentalframework entwickelt
- Keine großen Verbesserungen gefunden
- Trotzdem einiges interessante Erkenntnisse
  - Evolution für Hyperparameter funktioniert
    - Nur eine gute Fitnessfunktion fehlt
    - Unterschied zwischen Lernfortschritt und Spielstärke

Fazit

- AlphaZero Experimentalframework entwickelt
- Keine großen Verbesserungen gefunden
- Trotzdem einiges interessante Erkenntnisse
  - Evolution für Hyperparameter funktioniert
    - Nur eine gute Fitnessfunktion fehlt
    - Unterschied zwischen Lernfortschritt und Spielstärke
  - Alternative Explorationsmethoden zeigen vor allem wie gut die einfache Standardversion funktioniert

Fazit

- AlphaZero Experimentalframework entwickelt
- Keine großen Verbesserungen gefunden
- Trotzdem einiges interessante Erkenntnisse
  - Evolution für Hyperparameter funktioniert
    - Nur eine gute Fitnessfunktion fehlt
    - Unterschied zwischen Lernfortschritt und Spielstärke
  - Alternative Explorationsmethoden zeigen vor allem wie gut die einfache Standardversion funktioniert
  - Auxiliary Features aus dem inneren eines kleineren Netzwerks sind nur schwer nutzbar

Fazit

- AlphaZero Experimentalframework entwickelt
- Keine großen Verbesserungen gefunden
- Trotzdem einiges interessante Erkenntnisse
  - Evolution für Hyperparameter funktioniert
    - Nur eine gute Fitnessfunktion fehlt
    - Unterschied zwischen Lernfortschritt und Spielstärke
  - Alternative Explorationsmethoden zeigen vor allem wie gut die einfache Standardversion funktioniert
  - Auxiliary Features aus dem inneren eines kleineren Netzwerks sind nur schwer nutzbar
- Vorschläge für weitere Forschung

Fazit

- AlphaZero Experimentalframework entwickelt
- Keine großen Verbesserungen gefunden
- Trotzdem einiges interessante Erkenntnisse
  - Evolution für Hyperparameter funktioniert
    - Nur eine gute Fitnessfunktion fehlt
    - Unterschied zwischen Lernfortschritt und Spielstärke
  - Alternative Explorationsmethoden zeigen vor allem wie gut die einfache Standardversion funktioniert
  - Auxiliary Features aus dem inneren eines kleineren Netzwerks sind nur schwer nutzbar
- Vorschläge für weitere Forschung
  - Suche nach Fitnessfunktion für Evolution

Fazit

- AlphaZero Experimentalframework entwickelt
- Keine großen Verbesserungen gefunden
- Trotzdem einiges interessante Erkenntnisse
  - Evolution für Hyperparameter funktioniert
    - Nur eine gute Fitnessfunktion fehlt
    - Unterschied zwischen Lernfortschritt und Spielstärke
  - Alternative Explorationsmethoden zeigen vor allem wie gut die einfache Standardversion funktioniert
  - Auxiliary Features aus dem inneren eines kleineren Netzwerks sind nur schwer nutzbar
- Vorschläge für weitere Forschung
  - Suche nach Fitnessfunktion für Evolution
  - Das Konzept des Netzwerkwachstums sollte weiter erforscht werden

Referenzen

- Levente Kocsis et al.: Bandit based monte-carlo planning, 2006.
- David Silver et al.: Mastering the game of go with deep neural networks and tree search, 2016
- David Silver et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, 2018

Danke für Ihre Aufmerksamkeit.