



Universiteit
Leiden
The Netherlands

Structured parallel programming for Monte Carlo Tree Search

Mirsoleimani, S.A.

Citation

Mirsoleimani, S. A. (2020, June 17). *Structured parallel programming for Monte Carlo Tree Search*. *SIKS Dissertation Series*. Retrieved from <https://hdl.handle.net/1887/119358>

Version: Publisher's Version

License: [Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden](#)

Downloaded from: <https://hdl.handle.net/1887/119358>

Note: To cite this publication please use the final published version (if applicable).

Cover Page



Universiteit Leiden



The handle <http://hdl.handle.net/1887/119358> holds various files of this Leiden University dissertation.

Author: Mirsoleimani, S.A.

Title: Structured parallel programming for Monte Carlo tree search

Issue Date: 2020-06-17

Samenvatting

Het proefschrift maakt deel uit van een groter project, het HEPGAME (High Energy Physics Game) project. Het hoofddoel van HEPGAME is het gebruik van AI-oplossingen, met name door MCTS te gebruiken voor de vereenvoudiging van HEP-berekeningen. Een van de problemen is het oplossen van relevant wiskundige expressies met miljoenen termen. Deze berekeningen kunnen worden verricht met het FORM-programma, dat is een specifiek software-pakket voor symbolische manipulatie. Omdat de berekeningen rekenintensief zijn en daardoor veel tijd kosten, is het FORM-programma parallel uitgevoerd om berekeningen binnen een redelijke tijd te executeren. Daarom moet elk nieuw algoritme op basis van MCTS ook parallel kunnen worden uitgevoerd. Deze eis ligt direct onder de probleemstelling van het proefschrift: “Hoe ontwerpen we een gestructureerde, op patronen gebaseerde parallele programma-aanpak voor efficiënt parallelisme van MCTS voor zowel *multi-core* als *manycore* machines met een gedeeld geheugen?”. Om deze vraag te beantwoorden, benadert het proefschrift de MCTS de paralleliserings problemen op drie niveaus: (1) implementatieniveau, (2) datastructuurniveau, en (3) algoritmeniveau. Op het implementatieniveau hebben we parallelisatie op taakniveau verkozen boven parallelisatie op *thread*niveau. Parallelisatie op taakniveau biedt ons efficiënte paralleliteit voor MCTS om kernen (*cores*) te gebruiken op zowel *multi-core* als *manycore* machines. Op het niveau van de datastructuur hebben we een *lock-free* datastructuur voorgesteld die de korektheid garandeert. Een *lock-free* gegevensstructuur (1) verwijdt de synchronisatie-overhead wanneer een parallel programma veel taken nodig heeft om cores te gebruiken en (2) verbetert zowel de prestaties als de schaalbaarheid. Op het algoritmeniveau hebben we eerst uitgelegd hoe een pijplijnpatroon moet worden gebruikt voor parallelisatie van MCTS om de search overhead te overwinnen. Daarna konden we via een stapsgewijze aanpak de gestructureerde parallele programmering voor Monte Carlo Tree Search gestalte geven.

