<심사용 논문>

MCTS 기반 소자 배치를 위한 전략적 회귀 알고리즘에 관한 연구

*하 영 서, **심 재 창

*(주) 티에스이

**국립경국대학교 컴퓨터공학과

A Study on Strategic Backtracking Algorithms for MCTS-based Component Placement

*Yeongseo Ha, **Jaechang Shim

*TSE Co., Ltd

**Dept. of Computer Engineering, Kyungguk National University

● 투고분야 : ICT 산업융합

● Corresponding Author : Jaechang Shim
- Address : 1375 Gyeongdong-ro, Andong-si, Gyeongsangbuk-do, Republic of Korea (36729)
- TEL : +82-54-820-5645
- FAX : +82-54-820-6164
- E-mail : jcshim@andong.ac.kr

● This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-000XXXX)

※ 교신저자 및 연구비 지원사항은 모두 영문으로 작성되어야 함.!!!!

MCTS 기반 소자 배치를 위한 전략적 회귀 알고리즘에 관한 연구

A Study on Strategic Backtracking Algorithms for MCTS-based Component Placement

Abstract

With the increasing demand for high-performance and compact electronic devices, spatial optimization techniques for the optimal placement of components within restricted areas have become crucial. This paper proposes a Strategic Backtracking and Pruning algorithm based on Monte Carlo Tree Search (MCTS) to solve the component placement problem characterized by a vast search space. While conventional MCTS suffers from reduced search efficiency when encountering deadlocks, the proposed algorithm overcomes this limitation through non-sequential backtracking based on an elite candidate pool. Furthermore, an integer grid system and GPU kernel acceleration were implemented to minimize computational overhead. Experimental results demonstrate that the proposed method achieves a 98.7% placement rate within an execution time of 212.8 seconds for 500,000 iterations, proving its superior performance in both search speed and success rate compared to existing methods.

Keyword: Monte Carlo Tree Search (MCTS), Component Placement, Strategic Backtracking, Pruning

1. 서론

최근 전자 기기의 고성능화와 소형화 추세에 따라 제한된 영역 내에 다수의 소자를 최적으로 배치하는 공간 최적화 기술이 ICT 산업 전반에서 핵심적인 과제로 대두되고 있다[1]. 특히 반도체 설계 자동화(EDA) 및 정밀 회로 기판 배치 공정에서는 수많은 소자의 크기와 위치 제약 조건을 만족하면서도 배치 효율을 극대화해야 한다[2]. 이러한 공간 배치 문제는 전형적인 조합 최적화 문제(Combinatorial Optimization Problem)로, 소자의 개수가 증가함에 따라 탐색해야 할 경우의 수가 지수적으로 증가하는 복잡성을 지닌다. 이를 해결하기 위해 기존에는 유전 알고리즘이나 담금질 기법 등이 활용되었으나, 실시간으로 변화하는 설계 환경에 대응하기에는 계산 효율성과 수렴 속도 면에서 한계를 보여왔다[3-4].

이러한 한계를 극복하기 위해 최근에는 통계적 확률에 기반하여 유망한 탐색 경로를 집중적으로 추적하는 몬테카를로 트리 탐색(Monte Carlo Tree Search, MCTS) 기법이 대안으로 주목받고

있다[5-6]. MCTS는 방대한 탐색 공간 내에서 효율적인 의사결정을 지원하지만, 소자 배치 문제에 적용할 경우 특정 지점에서 더 이상 배치가 불가능한 교착 상태에 빠질 경우 탐색 효율이 급격히 저하되는 고유의 문제를 가지고 있다. 기존의 MCTS 구조에서는 이러한 상황 발생 시 단순히 부모 노드로 한 단계 복귀하는 순차적 회귀 방식을 취하는데, 이는 이미 실패가 예견된 경로 근처를 반복 탐색하게 하여 전체적인 배치 성공률을 낮추는 원인이 된다.

본 논문에서는 MCTS 기반의 배치 탐색 중 발생하는 지역 최적점(Local Optima) 및 탐색 실패 구간을 효과적으로 탈출하기 위한 전략적 회귀알고리즘을 제안한다. 제안하는 알고리즘은 탐색 중 배치가 불가능한 상태에 도달했을 때, 단순히 이전 단계로 돌아가는 대신 트리 내에서 방문 가치와 성공 확률이 높았던 엘리트 노드와 탐색의 다양성을 보존하기 위한 무작위 후보군을 포함하는 후보 풀을 구성한다. 이후 현재 노드의 상태 정보를 바탕으로 확률적 샘플링을 통해 가장 유망한 재시작 지점으로 비순차적 점프를 수행함으로써 탐색의 유연성을 확보하였다. 또한, 대규모 소자 배치 시 발생하는 연산 부하를 줄이기 위해 정수 그리드 체계를 도입하고, GPU 커널 가속을 통해 실시간에 가까운 배치 최적화 시스템을 구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구의 대상인 소자 배치 문제를 조합 최적화 관점에서 정의하고, 이를 해결하기 위한 핵심 도구인 MCTS 알고리즘의 기본 원리를 기술한다. 3장에서는 본 논문의 핵심 기여점인 전략적 회귀 알고리즘의 설계 원리와 후보군 관리 체계, 그리고 효율적인 재탐색을 위한 노드 전이 확률 모델을 상세히 설명한다. 4장에서는 제안한 알고리즘을 다양한 배치 환경에 적용하여 기존의 순차 회귀 기반 MCTS 및 타 최적화 기법들과 배치 성공률, 탐색 속도 측면에서 객관적인 비교 실험을 수행하고 성능을 평가한다. 마지막으로 5장에서 본 연구의 성과를 정리하고 결론을 맺는다.

2. 이론

2.1 조합 최적화 문제의 소자 배치

조합 최적화는 주어진 유한한 객체들의 집합 내에서 특정 제약 조건을 만족하면서 목적 함수를 최대화하거나 최소화하는 최적의 조합을 찾는 수학적 문제를 의미한다. 전자 소자 배치는 이러한 조합 최적화의 대표적인 사례 중 하나로, 한정된 2차원 평면 내에 다양한 크기와 형상을 가진 소자들을 겹침 없이 배치하여 공간 효율성을 극대화하는 것을 목표로 한다.

본 연구에서 다루는 소자 배치 문제는 다음과 같은 복합적인 제약 조건을 포함한다. 첫째, 모든 소자는 배치 영역의 경계를 벗어나선 안 되며, 기배치된 타 소자와 물리적 점유 영역이 중첩되지 않아야 한다. 둘째, 각 소자는 설계 의도에 따라 지정된 앵커 핀(Anchor Pin)으로부터 일정 거리 이내에 배치되어야 하는 거리 제약을 가진다. 셋째, 소자의 배치 순서와 각 단계에서의 회전 각도 0°, 90°에 따라 이후 단계의 배치 가능 영역이 동적으로 변화하는 가변성을 지닌다.

이러한 배치 문제는 전형적인 빈 패킹 문제(Bin Packing Problem) 또는 컨테이너 터미널 배치 문제와 유사하며, 결정론적 방법으로는 최적해를 찾는 것이 불가능에 가까운 NP-난해(NP-hard) 문제로 분류된다. 보드에 배치할 소자의 수가 많을수록, 소자가 배치될 수 있는 위치가 넓을수록, 전체 탐색 공간은 크게 증가한다. 예를 들어, 150개의 소자에 대해 각 소자의 거리 제약, 충돌 등을 고려하여 배치할 경우 발생하는 경우의 수는 전수 조사가 불가능한 수준에 도달한다.

따라서 기존의 정수 계획법이나 단순 휴리스틱 기법은 소자 개수가 적은 초기 단계에서는 유효

할 수 있으나, 배치가 진행됨에 따라 탐색 공간이 급격히 제약되는 차원의 저주 현상으로 인해 수렴 속도가 급격히 저하된다[14]. 특히, 특정 소자의 배치가 이후 소자들의 배치 가능성을 원천적으로 차단하는 교착 상태를 사전에 예측하기 어렵다는 점은 이 문제의 난이도를 더욱 높이는 핵심 요인이다. 본 연구에서는 이러한 방대한 조합 최적화 공간 내에서 효율적으로 유망한 경로를 탐색하기 위해 통계적 확률에 기반한 지능형 탐색 알고리즘을 도입하고자 한다.

2.2 MCTS

MCTS는 방대한 탐색 공간을 가진 의사결정 문제에서 통계적 샘플링을 통해 최적의 경로를 찾아내는 알고리즘이다. 전수 조사가 불가능한 복잡한 게임인 바둑이나 체스와 같은 조합 최적화 문제에서 탁월한 성능을 발휘하며, 특히 별도의 학습 데이터 없이도 시뮬레이션 결과를 바탕으로 자가 학습에 가까운 탐색을 수행한다는 장점이 있다[7-8].

소자 배치 문제에 적용된 MCTS는 각 소자의 배치를 하나의 상태로 정의하고, 가능한 배치 좌표와 회전 각도를 액션으로 취급하여 트리 구조를 형성한다. MCTS의 탐색 과정은 다음의 반복적인 4단계 사이클로 구성된다. MCTS의 탐색 과정을 그림 1에 나타내었다.

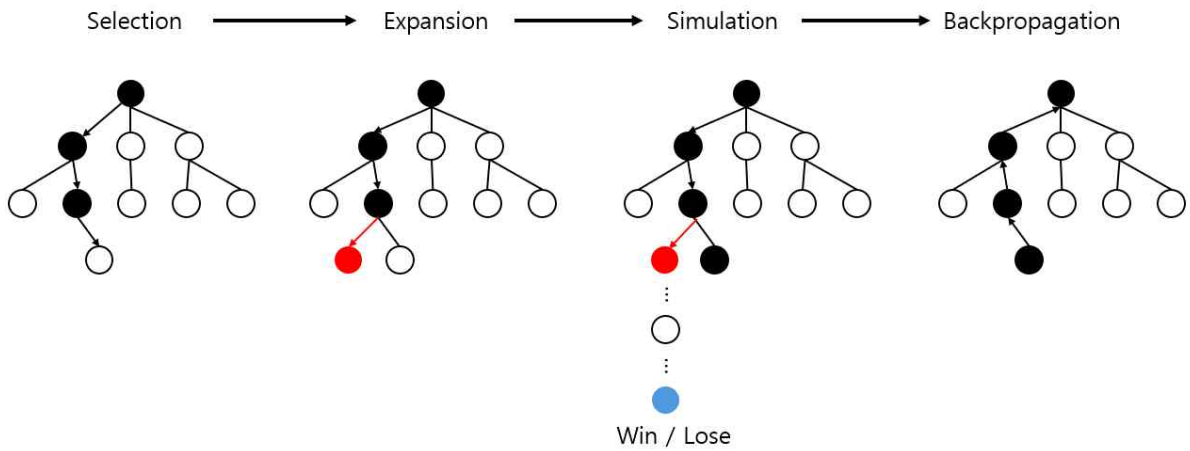

Fig. 1. Monte Carlo Tree Search Process.

그림 1을 보면 선택(Selection), 확장(Expansion), 시뮬레이션(Simulation), 역전파(Backpropagation) 순으로 MCTS가 진행되는 것을 볼 수 있다. 선택은 최상단 루트 노드에서 시작하여 현재까지 축적된 통계 정보를 바탕으로 가장 유망한 자식 노드를 선택한다. 확장은 선택된 노드가 단말 노드(Leaf Node)이고 아직 탐색 되지 않은 가능한 배치 액션이 남아 있다면, 하나 이상의 자식 노드를 생성하여 트리에 추가한다. 시뮬레이션은 새롭게 확장된 노드로부터 게임의 끝까지 무작위 혹은 휴리스틱 정책에 따라 빠르게 배치를 진행하여 결과값을 얻는다. 역전파는 시뮬레이션에서 얻은 최종 보상을 현재 경로에 있는 모든 노드에 전달하여 방문 횟수와 가치 정보를 갱신한다.

소자 배치 문제에서 MCTS는 이전 단계의 배치가 이후 단계에 미치는 영향력을 통계적으로 학습함으로써, 장기적인 관점에서 배치 성공률이 높은 영역을 우선적으로 탐색한다. 그러나 고밀도 배치 환경에서는 탐색 중반 이후 배치 가능한 공간이 급격히 줄어들며, 특정 노드에서 더 이상 확

장이 불가능한 교착 상태가 빈번하게 발생한다. 일반적인 MCTS는 이러한 상황에서 단순히 부모 노드로 돌아가 다른 자식을 탐색하는 방식을 취하지만, 이는 복잡한 제약 조건이 얽힌 배치 문제에서 탐색 효율을 저하시키는 주요 원인이 된다. 본 연구에서는 이러한 MCTS의 한계를 극복하기 위해 기존의 순차적 회귀 방식을 개선한 전략적 회귀 알고리즘을 제안한다.

3. 제안한 방법

3.1 제안한 알고리듬의 개요

본 장에서는 MCTS의 탐색 효율을 극대화하기 위해 제안하는 전략적 회귀 기반의 소자 배치 최적화 알고리즘의 전체 구조를 설명한다. 기존 MCTS는 탐색 중 배치가 불가능한 교착 상태에 직면할 경우, 단순히 직전 부모 노드로 돌아가 다른 자식 노드를 탐색하는 선형적 회귀 방식을 사용한다. 그러나 소형 고성능 기기의 소자 배치와 같이 제약 조건이 까다로운 환경에서는 특정 단계의 잘못된 배치가 이후 수십 단계 뒤의 실패로 이어지는 경우가 빈번하다. 이 경우 단순 회귀는 실패 원인이 된 근본적인 노드까지 도달하는 데 방대한 반복 연산을 소모하며, 결국 지역 최적점에 갇히는 한계를 보인다.

이를 해결하기 위해 본 논문에서 제안하는 알고리즘은 탐색 실패 시 현재 경로에 국한되지 않고, 전체 트리에서 가장 유망한 지점으로 비순차적인 이동을 수행한다. 알고리즘의 전체 프로세스는 다음과 같은 차별점을 지닌다. 첫째, 탐색 과정에서 우수한 성과를 보인 노드들을 실시간으로 선별하여 후보군 풀을 유지한다. 둘째, 교착 상태 발생 시 현재 노드와의 거리와 노드의 방문 희소성을 고려한 전이 확률 모델을 통해 최적의 재시작 지점을 선정한다. 셋째, 이러한 복잡한 탐색 과정을 실시간으로 처리하기 위해 GPU 병렬 연산을 통해 가속화한다. 소자 배치 최적화 문제의 경우 한 소자 배치에도 많은 경우의 수가 존재하여 여러 경우의 수를 빠르게 탐색하는 것이 중요하다[9-12].

아래 표 1은 제안한 알고리듬과 기존 MCTS의 특성을 비교한 표다.

Table 1. Comparison Between Conventional MCTS and the Proposed Method.

Aspect	Conventional MCTS	Proposed Method
Improvement Focus	Selection policy refinement	Structural redesign of backtracking
Deadlock Handling	Sequential return to parent node	Non-sequential jump to elite candidates
Failed Node Treatment	Re-visitable	Explicit pruning (No Re-access)
Exploration Diversity	Statistically adjusted	Structurally enhanced via elite transitions

3.2 후보군 관리 및 전이 확률 모델

전략적 회귀의 핵심은 어디로 돌아갈 것인가를 지능적으로 결정하는 데 있다. 본 연구에서는 이를 위해 엘리트 노드 관리와 확률적 전이 메커니즘을 도입하였다.

후보군 구성 및 업데이트는 탐색 중 생성되는 모든 노드 중 상위 10%의 보상 가치를 지닌 엘

리트 노드와 탐색의 다양성을 확보하기 위해 무작위로 추출된 랜덤 노드를 별도의 후보군 풀에 저장한다. 이 풀은 확장 단계마다 실시간으로 갱신되며, 현재 탐색 경로가 막혔을 때 즉시 참조할 수 있는 '검증된 대안 경로'의 집합 역할을 한다. 다음 식 1에 후보군 구성 및 업데이트에 대한 식을 나타내었다.

$$S_{reg} = \alpha \cdot \frac{1}{dist(curr, e)} + \beta \cdot (1 - \frac{V_e}{\sum_{i \in S} V_i}) \quad (1)$$

여기서 $\sum_{i \in S} V_i$ 는 후보 집합 S에 속한 각 노드 i의 방문 횟수 V_i 를 전부 합한 총 방문 횟수를 의미한다. 따라서 $\frac{V_e}{\sum_{i \in S} V_i}$는 후보 집합 내에서 노드 e가 선택 및 방문된 상대적 빈도이며, $(1 - \frac{V_e}{\sum_{i \in S} V_i})$항은 상대적으로 덜 방문된 노드에 더 큰 가중치를 부여하여 탐색의 다양성을 유도한다. 거리 함수 $dist(curr, e)$ 는 두 노드 간의 트리 깊이 차이로 정의되며 가중치 α와 β는 각각 회귀 비용과 방문 희소성의 중요도를 조절하는 하이퍼파라미터이다. 본 연구에서는 실험적 검증을 통해 가중치 α와 β는 각각 1.0, 0.7을 사용하였다. 두 계수는 정규화하지 않았으며, 이는 회귀 비용 항을 상대적으로 더 중요하게 반영하기 위함이다.

$$dist(curr, e) = depth(curr) - depth(e) \quad (2)$$

이 함수는 회귀 시 되돌려야 하는 배치 단계 수와 동일하며, 실질적으로는 미배치 상태로 복원되는 컴포넌트 수를 의미한다. 소자 배치 특성상 하나의 컴포넌트 배치를 위해서는 이미 배치된 다른 컴포넌트와의 충돌 및 거리 제약을 동시에 고려해야 하므로, 과도한 회귀는 다수의 컴포넌트를 재배치해야 하는 계산 비용을 초래한다. 따라서 $curr$ 노드와 상대적으로 가까운 계층의 노드에 높은 가중치를 부여하는 것은 재탐색 비용 최소화 관점에서 합리적이다.

본 연구에서 회귀를 하는 기준은 가지치기이다. 어떤 컴포넌트가 더 이상 배치될 수 있는 자리가 없을 때 회귀를 진행하며, 해당 노드는 가지치기되어 추후에 접근하지 못하도록 막는다. 다음 그림 2에 전략적 회귀를 나타내었다.

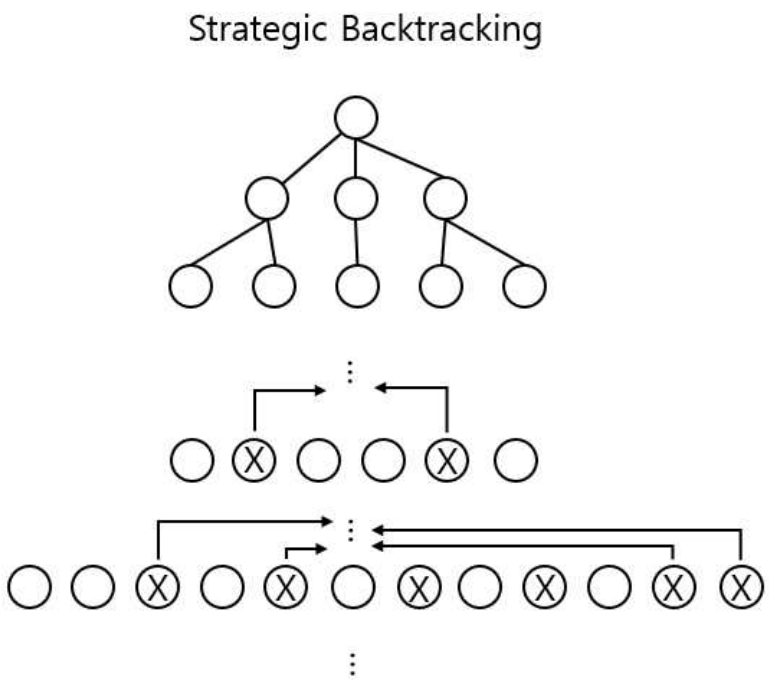

Fig. 2. Strategic Backtracking.

그림 2를 보면 X 표시된 노드는 더 이상 컴포넌트를 배치하지 못하여 가지치기 된 노드이다. 가지치기가 되면 해당 노드는 접근이 불가능하게 되며, 식 1을 통해서 이전 계층의 엘리트 노드로 다시 거슬러 올라가서 재배치를 진행한다. 가지치기로 해당 노드 접근을 막는 이유는 소자 배치 특성상 한 소자가 배치될 수 있는 자리는 방대하다. 이런 이유로 인해 너무 많은 경우의 수가 존재할 수 있으므로 경우의 수를 줄이기 위해 가지치기를 진행한다. 이러한 기법은 소자 배치 문제의 교착 상태를 해결하는 데 도움이 되며, 엘리트 후보 외에도 랜덤 후보로 회귀하여 엘리트 후보가 교착 상태에 빠져 회귀 불능 상황을 사전에 방지한다.

한편, UCT 기반 MCTS는 UCB1 탐험 항을 통해 동일 부모 하의 자식 노드 선택에서 방문 횟수가 적은 노드를 우선 탐색하도록 유도한다. 그러나 이는 액션 선택 단계에서의 탐험 - 활용 균형을 조절하는 메커니즘이며, 교착 상태 이후의 재시작 지점 선택 문제와는 적용 수준이 다르다. 또한 표준 UCT는 실패 상태에 대한 재방문을 확률적으로 허용하지만, 본 연구에서는 확장 불가능한 노드를 명시적으로 가지치기하여 해당 상태로의 재접근을 차단한다. 따라서 제안 방식은 교착 상태 탈출을 구조적으로 설계하고, 반복 실패 경로를 제거하며, 재배치 비용과 탐색 다양성을 동시에 고려한다는 점에서 기존 MCTS 변형과 차별화된다.

3.3 CUDA 커널 기반 병렬 배치 검사 및 연산 최적화

MCTS의 탐색 효율은 단위 시간당 수행되는 시뮬레이션 횟수에 비례하며, 이는 곧 개별 노드에서 유효 액션을 추출하는 속도에 의해 결정된다. 고밀도 배치 환경에서는 주변의 탐색 범위 내에서 수백 개의 후보 좌표를 검사해야 하며, 좌표마다 소자의 형상에 따른 충돌 검사가 동반되어야 한다.

본 연구에서는 이 과정을 CPU의 순차 연산 대신 GPU의 대규모 병렬 처리 아키텍처를 활용하

여 가속화하였다.

먼저, 스레드-그리드 매핑 최적화를 위해 핀 주변의 탐색 영역을 2차원 블록 구조로 추상화하였다. 핀의 중심 좌표를 기준으로 설정된 최대 탐색 거리를 GPU의 스레드 인덱스에 직접 매핑함으로써, 불필요한 조건문 연산을 제거하고 모든 스레드가 동시에 독립적인 후보 좌표의 유효성을 검사하도록 설계하였다.

둘째, 메모리 접근 및 데이터 전송 오버헤드 최소화를 위해 데이터 상주 전략을 도입하였다. 소자의 형상 정보와 같이 정적인 데이터는 시뮬레이션 시작 전 GPU의 상수 메모리 또는 글로벌 메모리에 미리 할당하여, 매 루프마다 발생하는 CPU-GPU 간의 데이터 복사 비용을 차단하였다. 또한, 결과값을 저장하는 버퍼를 재사용 가능한 구조로 사전 할당함으로써 메모리 할당 및 해제에 따른 지연 시간을 제거하였다.

이러한 가속화는 MCTS가 제한된 시간 내에 더 깊고 넓은 탐색 트리를 형성할 수 있게 함으로써, 최종적인 배치 성공률을 높이는 핵심 동력이 된다[13].

4. 실험 및 비교

4.1 실험 및 비교

표 1에 해당 실험에 사용된 PC 사양을 나타내었다.

Table 2. PC Spec.

Category	Specification
CPU	Intel(R) Core(TM) i9-12900
GPU	NVIDIA GeForce RTX 3090Ti
RAM	128GB
OS	Windows 11

소자 배치에 사용될 보드 및 소자는 임의로 정의하여 실험이 사용하였다. 각 소자는 앵커 핀에 연결된다. A 소자는 가로 2, 세로 1의 크기를, B 소자는 가로 4, 세로 3의 크기를 가진다. 각 소자의 최대 배치 거리는 A의 경우 5, B의 경우 10으로 정의하고 맨해튼 거리 계산을 통해 마름모의 반경 내부에 배치될 수 있게 설정하였다. 다음 그림 3에 임의로 제작한 보드를 나타내었다.

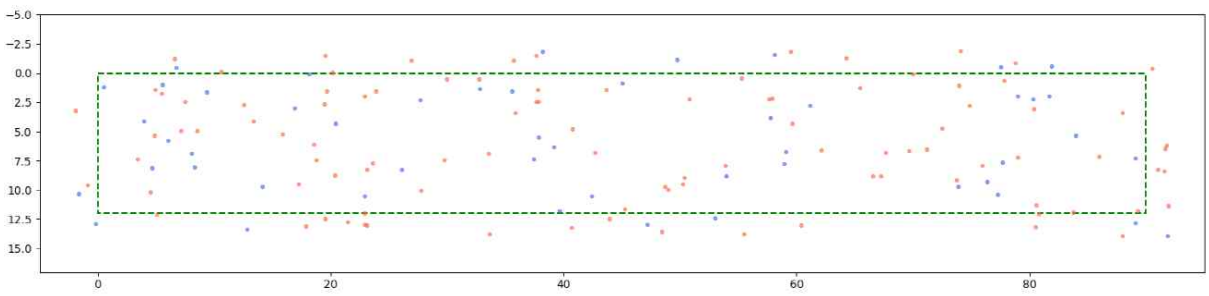

Fig. 3. Board Layout and Pin Locations.

그림 3에 있는 녹색 점선은 보드의 외곽을 의미하며, 빨간 점은 A 소자의 앵커 핀, 파란 점은 B 소자의 앵커 핀을 의미한다. 각 앵커 핀을 기준으로 맨해튼 거리 계산을 통해 각 소자가 배치될 수 있는 마름모 반경을 생성하여 배치한다. 다음 그림 4에는 A, B 소자를 나타내었다.

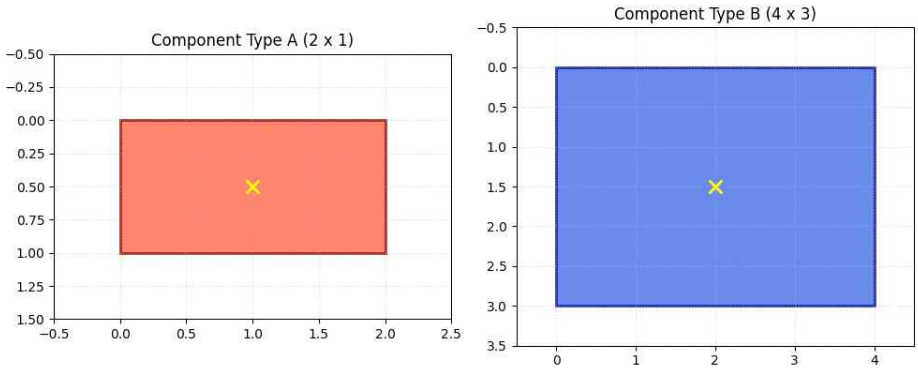

Fig. 4. A Component(left), B Componet(right).

A 소자는 가로 2, 세로 1의 크기를 가지고 B 소자는 가로 4, 세로 3의 크기를 가진다. 각 소자의 중심에 있는 X는 앵커 핀과 연결되는 소자 내부 핀의 위치를 나타낸 것이다. 이 소자 내부 핀은 소자마다 조금씩 다르게 적용했다. 그 이유는 소자의 다양성을 고려하기 위함이다. 이 두 소자는 0°, 90° 두 각도로 회전하면서 보드에 배치된다.

실험은 기준은 배치 순서와 크게 기존 MCTS 방식과 전략적 회귀 MCTS 방식으로 나뉜다. 각 방식에서 MCTS 방식에서는 1) 배치 순서 랜덤 적용, 2) 크기가 큰 컴포넌트 먼저 배치, 3) 크기가 큰 컴포넌트 먼저 배치 + 1x1 그리드 사용으로 실험하였으며, 전략적 회귀 MCTS도 마찬가지로 1) 배치 순서 랜덤 적용, 2)크기가 큰 컴포넌트 먼저 배치, 3) 크기가 큰 컴포넌트 먼저 배치 + 1x1 그리드 사용으로 실험하였다.

MCTS 반복 횟수는 모두 50만 회로 동일하게 적용하였으며, MCTS 탐색 평가는 충돌, 밀착 두 가지만 고려하였다. 다음 그림 5에 각 방식에 따른 배치 결과 그림을 나타내었다.

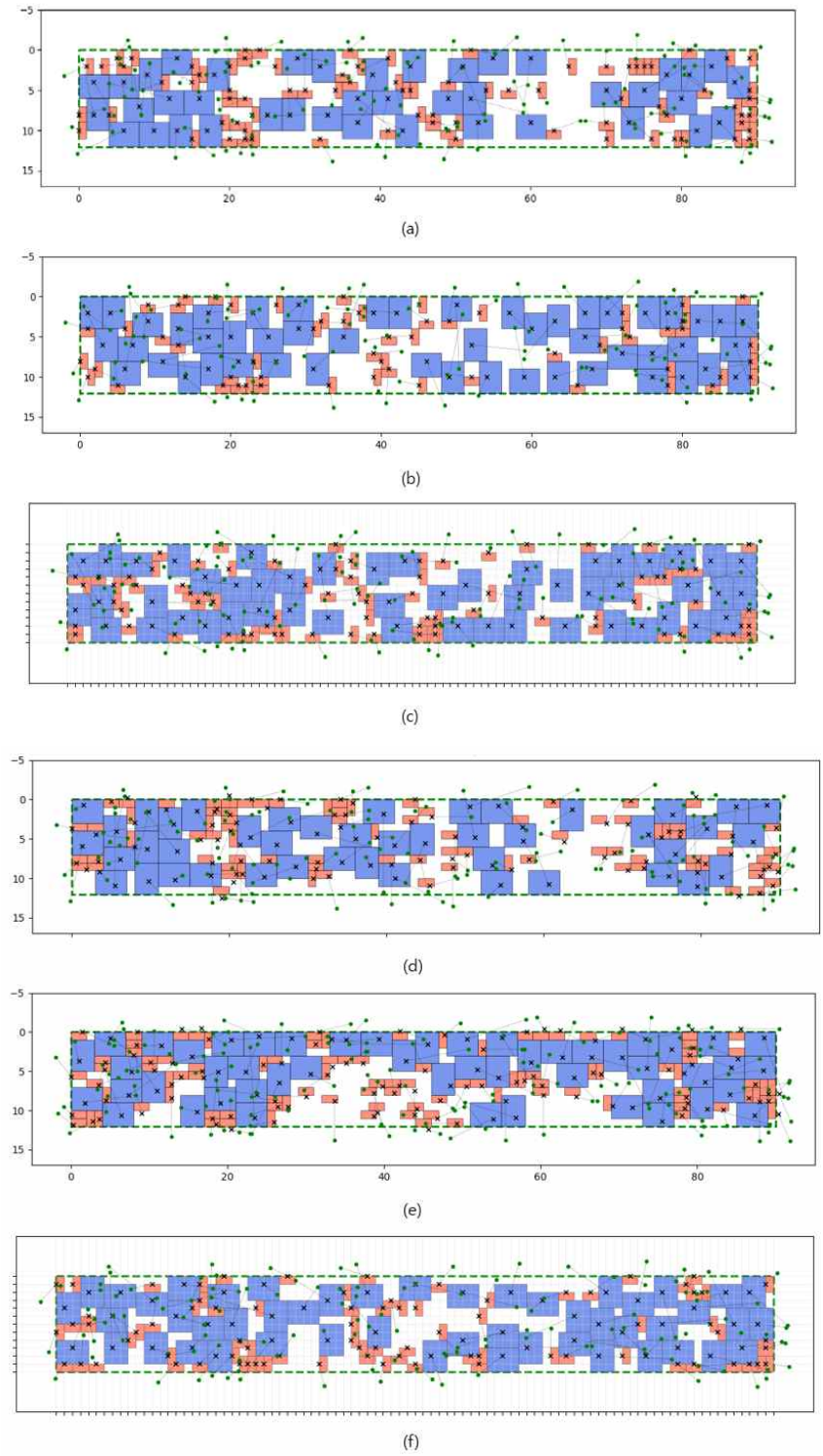

Fig. 5. Placement Results (a) Standard MCTS, (b) Standard MCTS with Large-First Strategy, (c) Standard MCTS with Strategy and Integer Grids, (d) Proposed Method, (e) Proposed Method with Large-First Strategy, (f) Proposed Method with Large-First Strategy and Integer Grids.

다음 표 2에 각 배치 결과 수치로 나타내었다.

Table. 2. Placement Results.

	Placed Components	Execution Time(s)	Success Rate(%)
Standard MCTS a	121/150	3888.8	80.7
Standard MCTS b	107/150	4135.9	71.3
Standard MCTS c	127/150	2827.8	75.9
Proposed Method a	128/150	254.9	85.3
Proposed Method b	**148/150**	**212.8**	**98.7**
Proposed Method c	127/150	2179.7	75.8

표 2에서 a는 MTCS 만 적용, b는 MTCS + 큰 소자 먼저 배치, c는 MTCS + 큰 소자 먼저 배치 + 1x1 그리드 적용을 나타낸다. 모든 비교군에 대해 MTCS 탐색 횟수는 50만 번을 적용하였으며, 그 결과 Proposed Method b 방식이 실행 시간 212.8초, 배치율 98.7%로 가장 좋은 결과를 보여주었다.

Standard MCTS b는 큰 소자를 먼저 배치하면서 다른 비교군 대비 교착 상태가 이르게 나타난 것으로 보인다. 그 결과 좋은 탐색을 하기 위해 시간은 오래 걸렸지만, 결과는 가장 좋지 못한 상태로 나타났다. Standard MCTS c는 Standard MCTS 비교군 중 가장 짧은 탐색 시간을 보였으며, 이 경우에는 Standard MCTS b에 비해 교착 상태가 늦게 나타난 것으로 보인다.

전략적 회귀 방식을 통해 교착 상태를 빠르게 탈출하고 MCTS가 더 좋은 탐색 방향을 찾을 수 있도록 하여 기존 MCTS 방식보다 실행속도가 급격하게 줄어든 모습을 볼 수 있다. 또한 기존 MCTS는 가지치기가 적용되지 않아 탐색에 실패한 노드 접근을 막지 않는다. 그래서 이전에 탐색에 실패했던 노드라도 다시 탐색을 허용한다. 이러한 동작이 반복되면 더 좋은 탐색 결과를 얻지 못하고 실패를 반복하여 전체 탐색 시간이 늘어난다. 반대로 전략적 회귀 방식은 탐색에 실패한 노드를 가지치기하여 전체적인 경우의 수 감수를 통해 실행 시간을 급격하게 감소시킨다.

소자 배치 결과에서도 전략적 회귀 방식은 기존 MCTS 방식보다 더 많은 소자를 배치한 모습을 보여주었으며, 엘리트 후보를 이용한 회귀가 소자 배치에 효과적인 방법이라는 것을 시사한다.

다만, 현재 소자의 크기는 소수점이 없는 정수 형태로 그리드 적용이 크게 효과를 발휘하지 못하였다. 또한 Proposed Method c의 경우에는 그리드 적용이 큰 도움이 되지 못하였으나, 그리드 단위로 충돌 및 밀착 평가를 하는 과정이 포함되면서 전략적 회귀 방식 중에서 가장 오랜 시간이 걸렸다. 만약, 소자의 크기가 소수였다면 그리드 방식이 소자 배치 위치의 경우의 수를 크게 줄이면서 속도 및 밀착에 효율성을 가져다주었을 것으로 예상된다.

5. 결론

본 연구에서는 고집적 회로 및 시스템 설계의 핵심 과제인 소자 배치 최적화 문제를 해결하기 위해, 몬테카를로 트리 탐색 알고리즘을 고도화한 전략적 회귀 및 가지치기 메커니즘을 제안하였다. 소자 배치 문제는 탐색 공간이 방대하고 소자 간의 간섭 및 핀과의 거리 등 복합적인 제약 조건을 만족해야 하는 NP-난해(NP-hard) 문제의 특성을 가진다. 기존의 MCTS 방식은 확률적 탐색을 통해 최적해에 근접하는 능력을 보여주었으나, 배치 불가능한 상태인 교착 상태에 빠졌을 때

의 탈출 효율성이 낮고, 이미 실패한 노드를 중복 탐색하는 비효율성이 한계로 지적되어 왔다.

본 연구의 핵심 기여는 이러한 기존 MCTS의 한계를 전략적 회귀와 명시적 가지치기를 통해 극복한 점에 있다. 실험 결과에 따르면, 단순히 MCTS만을 적용한 경우보다 큰 소자를 우선 배치하는 휴리스틱을 결합하고, 여기에 전략적 회귀 방식을 적용했을 때 비약적인 성능 향상이 관찰되었다. 특히 Proposed Method b 방식은 50만 번의 탐색 횟수 내에서 212.8초라는 짧은 실행 시간 동안 98.7%라는 극히 높은 배치율을 기록하였다. 이는 기존 방식들이 탐색 효율 저하로 인해 막대한 연산 시간을 소모하거나 배치율이 정체되었던 것과 대조적인 성과이다.

이러한 성능 향상의 주된 원인은 가지치기 메커니즘을 통한 탐색 공간의 효율적 제어에 있다. 표준적인 MCTS는 실패한 노드에 대해서도 확률적인 접근을 완전히 차단하지 않기 때문에, 불필요한 연산이 반복되는 경향이 있다. 그러나 본 연구에서 도입한 가지치기 방식은 탐색에 실패한 노드에 대한 재접근을 물리적으로 제한함으로써, 알고리즘이 보다 유망한 노드 확장에 집중할 수 있는 환경을 제공하였다. 또한, 교착 상태 발생 시 단순히 루트 노드로 돌아가는 것이 아니라, 엘리트 후보를 선택하여 회귀하는 전략은 MCTS가 지역 최적해에 매몰되지 않고 더 나은 탐색 방향을 신속하게 재설정할 수 있도록 돕는 핵심 동력이 되었다.

실험 과정에서 고찰된 또 다른 중요한 요소는 그리드 적용의 효과성이다. 본 실험에서는 소자의 크기가 정수 형태로 구성되어 1x1 그리드 적용이 배치 효율에 미치는 영향이 미미한 것으로 나타났으나, 이는 역설적으로 본 알고리즘의 확장 가능성을 시사한다. 소자의 크기가 소수점 단위의 미세 치수를 가질 경우, 탐색해야 할 위치의 경우의 수는 기하급수적으로 증가하게 된다. 이때 본 연구에서 제안한 그리드 기반 좌표 제어 방식과 전략적 회귀 모델이 결합된다면, 연속적인 탐색 공간을 이산화하여 연산 복잡도를 획기적으로 낮추고 밀착 배치의 정밀도를 높이는 데 결정적인 역할을 할 것으로 기대된다.

결론적으로, 본 연구에서 제안한 전략적 회귀 및 가지치기 기반 MCTS 모델은 소자 배치 최적화 분야에서 실행 속도와 성공률이라는 두 마리 토끼를 동시에 잡을 수 있는 실전적인 해법임을 입증하였다. 엘리트 후보군을 활용한 회귀 전략은 대규모 컴포넌트 배치 문제에서 알고리즘의 수렴 속도를 가속화하는 유효한 수단임을 시사한다. 향후 연구에서는 정수형 소자를 넘어 다양한 비정형 형태와 소수점 단위의 정밀도를 가진 소자 모델로 실험 범위를 확장할 예정이며, 이를 통해 보다 복잡한 실제 하드웨어 설계 환경에서도 본 알고리즘의 범용성과 강건성을 검증하고자 한다.

REFERENCE

[1] S.N. Adya and I.L. Markov, “Fixed-outline floorplanning: Enabling hierarchical design,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 11, No. 6, pp. 1120-1135, 2004.

[2] A.B. Kahng, J. Lienig, I.L. Markov, and J. Hu, VLSI Physical Design: From Graph Partitioning to Timing Closure, Springer, Dordrecht, 2011.

[3] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, “Optimization by Simulated Annealing,” Science, Vol. 220, No. 4598, pp. 671-680, 1983.

[4] J.P. Cohoon, S.U. Hegde, W.N. Martin, and D.S. Richards, “Distributed Genetic Algorithms for the Floorplan Design Problem,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 10, No. 4, pp. 483-492, 1991.

[5] 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.

[6] J.H. Cho and N.H. Kim, "Circuit Board Component Placement Optimization using Monte Carlo Tree Search Algorithm," Journal of Korea Multimedia Society, Vol. 22, No. 8, pp. 883-891, 2019.

[7] J.H. Cho and N.H. Kim, "Optimization of Component Placement on Printed Circuit Boards using Deep Learning Techniques," Journal of Korea Multimedia Society, Vol. 21, No. 12, pp. 1436-1443, 2018.

[8] D. Silver, A. Huang, C.J. Maddison, et al., "Mastering the Game of Go with Deep Neural Networks and Tree Search," Nature, Vol. 529, No. 7587, pp. 484-489, 2016.

[9] P.N. Iyer and A.J. Walker, "A Genetic Algorithm for 2D Bin Packing with One-Dimensional Line Constraints," Proceeding of the IEEE International Conference on Evolutionary Computation, pp. 144-149, 1997.

[10] T.C. Chen and Y.W. Chang, "Modern Floorplanning Based on Fast-B*-tree Representation," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, No. 4, pp. 637-650, 2006.

[11] G.M. Chaslot, M.H. Winands, and H.J. van den Herik, "Progressive Strategies for Monte-Carlo Tree Search," New Mathematics and Natural Computation, Vol. 4, No. 3, pp. 343-357, 2008.

[12] L. Kocsis and C. Szepesvari, "Bandit Based Monte-Carlo Planning," Proceeding of the 17th European Conference on Machine Learning (ECML), pp. 282-293, 2006.

[13] A. Barriga, M. St-Hilaire, T. Kunz, and J.P. Corriveau, "Parallelizing Monte Carlo Tree Search on GPU," Proceeding of the 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1941-1947, 2014.

[14] A. Mirhoseini, A. Goldie, M. Yazgan, J.W. Jiang, E. Songhori, S. Wang, Y.J. Lee, E. Johnson, O. Pathak, A. Nova, J.W. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, Q.V. Le, J. Laudon, R. Ho, R. Carpenter, J. Dean, "A Graph Placement Methodology for Fast Chip Design", Nature, Vol. 594, pp. 207-212, 2021.