Vol.2022-SLDM-199 No.11
2022/11/11

深層 Q 学習を用いた HLS 向け 最適スケジューリング

松岡尚典[†1]　瀬戸謙修[†1]

概要： 高位合成は, LSI の大規模化と複雑化に対処する設計技術として注目されている. その 1 工程であるスケジューリングは, 生成される RTL 回路の面積や性能に大きな影響を与える. 従来手法では, 大規模な DFG に対する最適性低下や処理時間増大が問題になっていた. そこで, 強化学習と深層学習を組み合わせたアルゴリズムである深層強化学習, 特に深層 Q 学習を用いたスケジューリングアルゴリズムを検討する.

1. はじめに

大規模集積回路(LSI: Large Scale Integrated circuits)の設計はレジスタ転送レベル(RTL: Resister Transfer Level)で詳細に記述する必要があるが, LSI の大規模化, 高集積化に伴い, 人手による RTL 記述は時間的コスト的に困難になっている. そこで注目されている技術である高位合成は, 高級言語により記述されたアルゴリズム記述から, 性能制約や面積制約を満たす回路を合成する.

高位合成の 1 工程であるスケジューリングは, アルゴリズム記述から生成された各演算間のデータの流れを表す DFG(Data Flow Graph)に対して, 各演算の実行ステップを決定する処理であり, 実際に合成される回路の性能や面積に大きな影響を与える. 演算器の数を制約として実行サイクル数を最小化するリソース制約スケジューリング問題は, NP 困難な組み合わせ最適化問題であり, 整数線形計画法(ILP: Integer Linear Programming)を使用して定式化することで最適解を求められるが, 現実的な規模の LSI 設計に対して最適解を得ることは難しい. そのため, 現在の HLS ツールでは FDS やリストスケジューリングといったヒューリスティック手法が用いられているが, グラフが大きくなった場合に局所最小解にはまりやすくなる難点がある.

本研究では, このリソース制約スケジューリング問題に対して機械学習, 特に強化学習とニューラルネットワークを組み合わせたアルゴリズムである深層強化学習の Deep Q-Network(DQN)を用いた手法の検討を行う.

2. 既存手法

リストスケジューリングは, 対象 DFG の各ノードに優先順位をつけリストとして保存しておき, 優先度が高いものから順にスケジューリングを行うヒューリスティック手法である[1]. ノード数が増えた場合に優先順位の付け方のパターンが増大する上, 優先順位の付け方が結果に大きく影響するため, 最適性の保証が難しくなる.

ILP ベースのスケジューリングでは与えられた制約式の中で, 目的関数を最大化または最小化する変数の組み合わせを ILP により求める.[2] ILP はそれ自身が NP 困難な組み合わせ最適問題であり, 小規模問題は高速で最適解を生成できる一方, 最適性を維持しつつ大規模な問題に適用するのが難しい[2].

強化学習によるスケジューリング手法として, [3]の時間制約付きスケジューリングへの適用がある. この研究では強化学習を行うにあたり, 各ノード間の依存関係を保つため, ランダム生成した多数のグラフの ILP スケジューリング結果を教師データとして学習させたニューラルネットワークを組み合わせる手法を提案している. ASAP スケジューリングに対しては平均 54.4%のリソース数の削減が行えているが, ILP で求めた最適解に対しては数十ノードの小規模問題においても平均 34%のリソース数の増加がある. また, 事前にニューラルネットワークを教師データにより学習させるため, 事前学習に数百時間と多大な時間がかかっている.

そこで本研究では, 深層強化学習において事前学習を行わずに, 依存関係を保ったスケジューリングを行う手法を検討する.

3. Q 学習と深層 Q 学習

強化学習の手法である Q 学習は, 時間 t における環境の状態 S_t でエージェントが行動 A_t を選択した時の行動価値関数 $Q(S_t, A_t)$を, 式(1)に従って更新することで学習を行う.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left\{R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right\} \quad (1)$$

ここで, R_t: 行動選択の報酬, α : 学習率, γ : 割引率である. その際, 各状態および各行動の Q 値を保存するために状態×行動のマトリクスに Q 値を当てはめた Q テーブルを作成するが,状態空間, 行動空間の大きな環境では Q テーブルで Q 値を管理するのが難しくなる.

そこで, 行動価値関数を Q テーブルの代わりにニューラルネットワークを用いた近似で求めるのが, 深層 Q 学習である. 深層 Q 学習は Q 学習と深層学習に基づくアルゴリズムであり, ある状態に対して取り得る全ての行動に対しての Q 値を同時に推定し行動を決定できるため, 探索空間の大きい巡回セールスマン問題等の組み合わせ最適化問題に

1 東京都市大学
Tokyo City University

Vol.2022-SLDM-199 No.11
2022/11/11

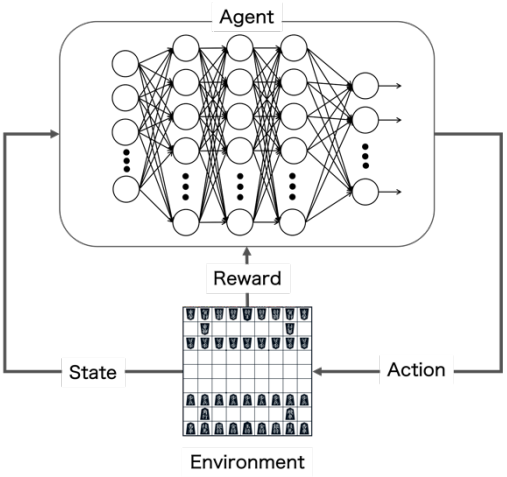

図 1. 深層 Q 学習概要

```
1.  while in episodes:
2.      環境をリセット
3.      while in step:
4.          if random number <= epsilon:
5.              ランダムに行動選択
6.          else:
7.              Q-Network に基づき行動選択
8.          実際に行動し環境観測
9.          報酬を獲得
10.         教師データとして T=R_t+γ max_a Q(S_t+1, a)を計算
11.         T を教師データ，環境の状態 St と行動 a を
            入力としてニューラルネットワークを更新
12.         step を 1 進める
```

図 2. 深層 Q 学習疑似コード

対して有効である[4].

深層 Q 学習では, 図 1 に示すように環境の状態をニューラルネットワークの入力とし, 各行動に対しての Q 値を学習する. 環境の状態とは, 将棋における局面すなわち棋譜の 1 場面のようなものである. (1)式中$R_t + \gamma \max_a Q(S_{t+1}, a)$を T と置くことで(1)式は次式(2)のように表される.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\{T - Q(S_t, A_t)\} \quad (2)$$

(2)式は次式(3)のように変形できる.

$$Q(S_t, A_t) \leftarrow (1-\alpha)Q(S_t, A_t) + \alpha T \quad (3)$$

(3)式は $Q(S_t, A_t)$を更新するたびに, 比率 α で $Q(S_t, A_t)$が T に近づくことを表している. そのため, T を正解ラベル, S_t, A_t を入力と見做すことでニューラルネットワークによる Q 値の学習が行える. また, エージェントが行動を選択する際にニューラルネットワークからのみではなく, 確率 ε でランダムに行動を選択する ε -greedy 法を用いることで, 局所最適解に陥ることを防ぐ.

図 2 に疑似コードを示す.

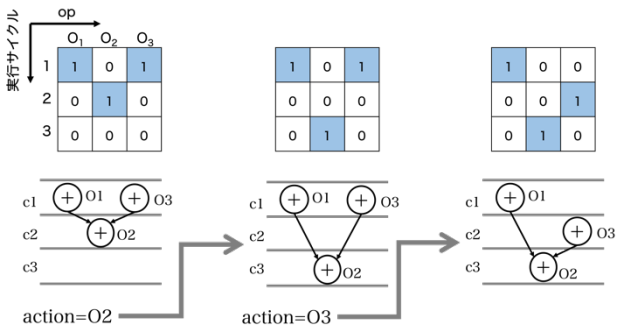

図 3. 上: 環境, 下: エージェントの行動

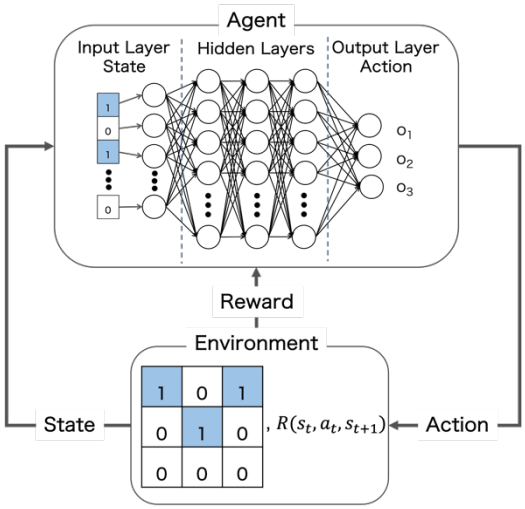

図 4. スケジューリング問題への深層 Q 学習の適用

4. リソース制約スケジューリングの深層 Q 学習向けモデル化の検討

ここでは, リソース数を制約として実行サイクル数を最小化するリソース制約スケジューリング問題を対象とした, 深層 Q 学習向けの学習モデルの検討を行う.

本研究ではスケジューリング問題を, 図 3 上に示すような各演算と実行サイクルのマトリクスを環境として定めた. 実行タイミングの上限値はリストスケジューリングを行った上で決定する. エージェントの行動空間 A は A={$o_0, o_1, \ldots, o_n$}とし, o_i は図 3 下に示すように, 演算リソース i の実行タイミングを 1 ステップ遅らせるものとした. この際図 3 に示すように, 現在演算リソースが配置されている場所は演算種ごとに定めた正整数が設定され, それ以外の場所全て 0 で埋められる. 各演算の取り得る実行サイクルの範囲は, o_i の ASAP スケジューリングでの実行タイミングを c^A_i, リストスケジューリングでの実行タイミングを c^L_i として, [c^A_i, c^L_i]の

表 1. 1 サイクル演算 1 つ, 2 サイクル演算 1 つの制約下でのスケジューリング結果

	実行サイクル数		
	ILP	LIST	提案手法
10ノード	13	13	13.2
20ノード	20	20	20.5

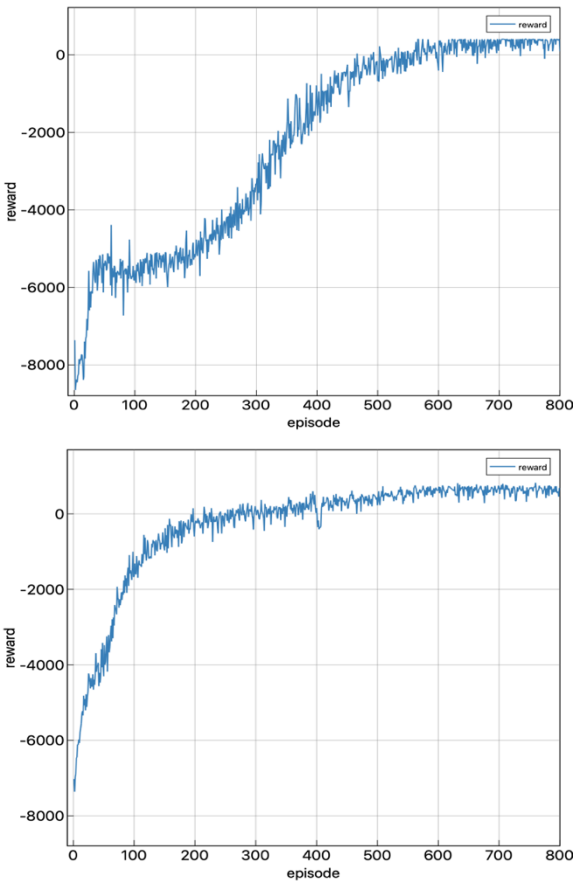

図 5. 1 サイクル演算 1 つ, 2 サイクル演算 1 つの制約下でエージェントが獲得した報酬の 30 回平均
上:10 ノード, 下: 20 ノード

表 2. 1 サイクル演算 1 つ, 2 サイクル演算 2 つの制約下でのスケジューリング結果

	実行サイクル数		
	ILP	LIST	提案手法
10ノード	8	9	8.2
20ノード	14	16	14.4

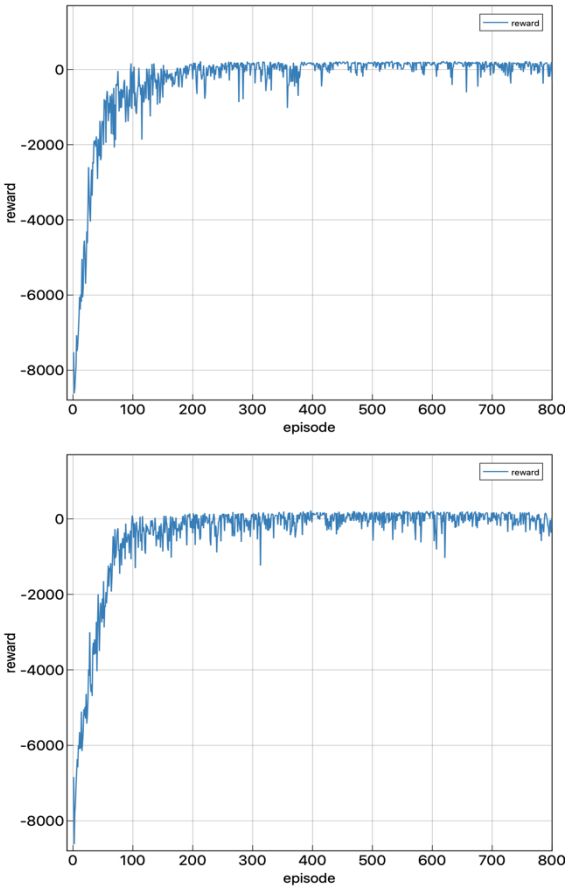

図 6. 1 サイクル演算 1 つ, 2 サイクル演算 2 つの制約下でエージェントが獲得した報酬の 30 回平均
上:10 ノード, 下: 20 ノード

範囲を移動可能範囲として制限した.

報酬は行動後の最大必要リソース数が変化しない場合を-1, 変化した場合は次式(4)とした.

$$R = (pCycle - nCycle) + 1.5 * (pOp_i - nOp_i) \quad (4)$$

ここで pCycle:行動前実行サイクル数, nCycle: 行動後実行サイクル数, pOpi: 行動前の演算種 i の所要リソース数, nOpi: 行動後の演算種 i の所要リソース数であり, (4)式右辺第 1 項と第 2 項はそれぞれ行動前後での実行サイクル数と, 演算種 i の所要リソース数の差分を表している. また, 行動後の最大必要リソース数が変化しない場合の報酬を-1 としたのは, エージェントが無駄な行動を多く取らないようにするためである. ASAP スケジューリング後の状態を初期状態として行動を開始するため, 初期状態ではリソース制約を満たしていない. そこで所要リソース数の削減を第一として, リソース数差分に 1.5 の重みを付けている.

今回, 行動を ε -greedy 法により選択するようにしたため, 各ノードの依存関係を無視する違法な行動を選択する場合がある. そのような違法手を選択した場合には報酬を-100 とし, 環境が制約条件を満たした場合の報酬は+100 とした.

学習対象のネットワークは全 5 層の全結合ネットワークで, 環境の状態を入力としてエージェントが選択する行動 o_i を出力する. 入力層は図 3 上に示したように環境として定めたマトリクスの要素数と同数のニューロンで ReLU 関数を持ち, 第 2-4 層は 64 ニューロンで同様に ReLU 関数を持つ. 出力層は行動空間と同数のニューロンで線形関数を持つ.

3 個の演算を含む DFG を例に, 深層 Q 学習への適用の概

表 3. 2 種類の制約条件下での
学習にかかった処理時間

	処理時間 [s]	
	10ノード	20ノード
1サイクル演算×1 2サイクル演算×1	102	414
1サイクル演算×1 2サイクル演算×2	63	289

要を図 4 に示す.

今回は 2 種類のリソース制約条件の下で実験を行った. 1 種類目はリソース制約条件として, 1 サイクル演算と 2 サイクル演算の各リソース 1 つずつとした. ランダム生成した 10 ノードと 20 ノードの問題での ILP スケジューリング, リストスケジューリング, 提案手法 30 回平均での結果を表 1 に示す. また, 図 5 に 10 ノードと 20 ノードでの, エージェントの獲得報酬の推移を示す.

2 種類目のリソース制約条件は 1 サイクル演算 1 つ, 2 サイクル演算 2 つとした. 1 種類目の時と同様にランダム生成した 10 ノードと 20 ノードの問題での ILP スケジューリング, リストスケジューリング, 提案手法 30 回平均での結果と獲得報酬の推移を表 2, 図 6 に示す. また, 2 種類の制約条件下での学習にかかった処理時間を表 3 に示す.

2 種類の制約下において, どちらもエピソードの進行に伴い獲得報酬が増加していることから, エージェントがスケジューリング問題に対して学習を行なっていることがわかる. 制約条件として 1 サイクル演算 1 つ, 2 サイクル演算を 2 つとした場合にリストスケジューリングに対して平均 9% の削減が行えている.

5. まとめ

リソース制約スケジューリングにおいて, 提案する深層 Q 学習アルゴリズムを用いることで実際に学習が行えること, 小さい問題では ILP の最適解に対して 3%の増加に留まる解を求められることが確認できた. また, 制約条件を 1 サイクル演算 1 つ, 2 サイクル演算 2 つとした時はリストスケジューリングに対して平均 9%の削減が行えており, リソース制約付きスケジューリング問題に対して深層 Q 学習が有効であることが確認できた.

しかし, 今回は行動の選択に全ての行動に対する ε-greedy 法を用いているため, ノード数の増大に伴う各ノード間の依存関係の複雑化により学習中に違法手を選択する確率が高まり, 学習時間が増大する問題がある. リソース制約スケジューリング 問題では合法手に対して違法手が多いため, 学習を効率的に行うために違法手の選択を抑えることが重要である.

今後, 予め状態 S_t に対する合法手の集合を構成しておき, その中から行動を選択する手法, また構成した合法手の集合から合法手を 1, 違法手を 0 としたマスク $M=\{m_1, m_2, \ldots, m_n\}$を, ニューラルネットワークの出力 A に対してかけることで, ニューラルネットワークが違法手を出力しないように制限し, 誤ったスケジューリング結果の生成を防ぐ手法の検討を行う.

参考文献

[1] Keith D. Cooper, Philip J. Schielke, and Devika Subramanian. (1998). An Experimental Evaluation of List Scheduling. Tech. rep., Deptartment of Computer Science, Rice University.
[2] Giovanni De Micheli. (1994). Synthesis and Optimization of Digital Circuits. McGraw-Hill Higher Education. pp. 198–202.
[3] H. Chen and M. Shen, "A Deep-Reinforcement-Learning-Based Scheduler for FPGA HLS," 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2019, pp. 1-8.
[4] Khalil, E., H. Dai, Y. Zhang, B. Dilkina, L. Song. 2017. Learning combinatorial optimizationalgorithms over graphs. Advances in Neural Information Processing Systems 30 6348–6358.