超大規模グラフにおける最大独立集合の求解手法の開発と情報通信への展開

研究代表者　Finnerty Patrick　神戸大学　大学院システム情報学研究科　講師
共同研究者　太田 能　神戸大学　大学院システム情報学研究科　教授

1 非直行多元接続（NOMA）割当問題について

1-1 NOMA における公平性及び消費電力を考慮した定式化について

本研究の文脈において、NOMA（Non-Orthogonal Multiple Access）は、限られた周波数資源を複数端末で共有することで高い接続密度を実現する重要技術である。しかし、端末の組み合わせや割り当て方によって通信容量や消費電力が大きく変動するため、NOMA のスケジューリングは本質的に複雑な組合せ最適化問題となる。この問題を効率的に解くため、近年は QUBO（Quadratic Unconstrained Binary Optimization）形式への変換と、専用ソルバーによる高速探索が注目されている。

既存研究では[1]、NOMA の割り当て問題を QUBO に変換し、通信容量の最大化や接続端末数の最大化を目的とした定式化が提案されてきた。しかし、これらの手法は消費電力を考慮していないという課題があった。その結果、QoS（Quality of Service）を満たす一方で、システム全体の電力消費が過大になるケースが多く、実運用を想定した最適化としては不十分であった。

この問題に対し、QoS を満たしつつ総消費電力を最小化する新しい QUBO 定式化（Minimize Power）を提案している。この手法では、QUBO の重み行列に「通信容量」ではなく「電力」を直接反映させることで、電力効率を重視した探索が可能となる。都市部マクロセル環境を模擬したシミュレーションでは、従来手法（Weighted, SR-weighted）が約 0.2 W 程度の電力を消費するのに対し、提案手法は約 0.1 W に抑えられ、約 50%の電力削減を達成している（Figure 1）。また、Figure 2 に示されている結果から、消費電力を大きく削減しても、依然として高いスループットが維持されていることが読み取れる。特に、Weighted や SR-weighted と比較すると、Minimize Power は電力削減を目的とした定式化であるにもかかわらず、スループットの中央値は依然として 10 Mbit/s 程度を確保しており、極端な性能劣化は発生していない。

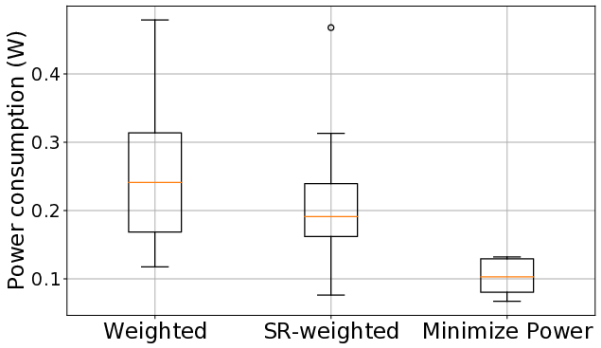

Figure 1 Comparison of the achieved power consumption between the original "Weighted" and "Sum-Rate weighted" methods and the proposed power-aware allocation method

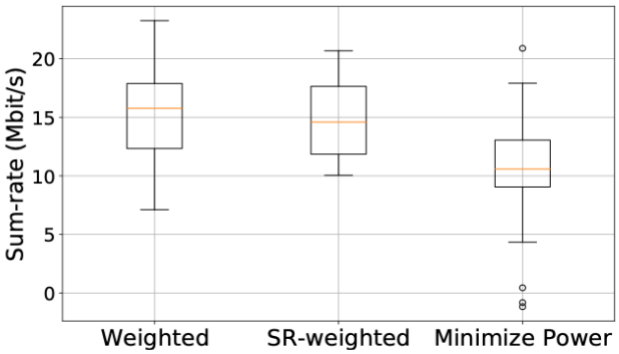

Figure 2 Comparison of the achieved sum-rate between the original "Weighted" and "Sum-Rate weighted" methods and the proposed power-aware method

この結果は、NOMA の割り当て問題において、目的関数の設計がシステム特性に大きく影響することを示している。また、QUBO 形式を用いることで、目的関数の変更や制約条件の追加が比較的容易であり、今回のように「電力最小化」という新たな最適化指標を柔軟に取り込める点は、QUBO ベースのアプローチの大きな利点である。上記の成果は電子情報通信学会の 2026 年総合大会にて発表された[2]。

1-2 グラフニューラルネットワーク及び GPGPU による高速化

提案した QUBO 定式化を PyQUBO で解く実験を行っているが、その結果、PyQUBO は正しい割り当てを返すものの、計算時間が実用上問題となるほど長いことが明らかになった。特に、ユーザ数 18・チャネル数 6 とい

う設定では、QUBO の変数数が 1000 を超え、1 回の計算に数秒から 10 秒以上を要することが明らかになった。論文[3]では、組み合わせ最適化問題を「グラフとして表現された構造」に対して、グラフニューラルネットワーク（GNN）を用いて、GPGPU の並列度を活用して直接解くことができるという点を示されている。NOMA 割当問題に適用した結果をFigure 3に示す。

チャネルコヒーレンス時間は、実用的なスケジューリングにおいて約 20 ms というリアルタイム制約から程遠い状況であった。改善に向けて、アルゴリズムの工夫及び FPGA による高速化を試みた。

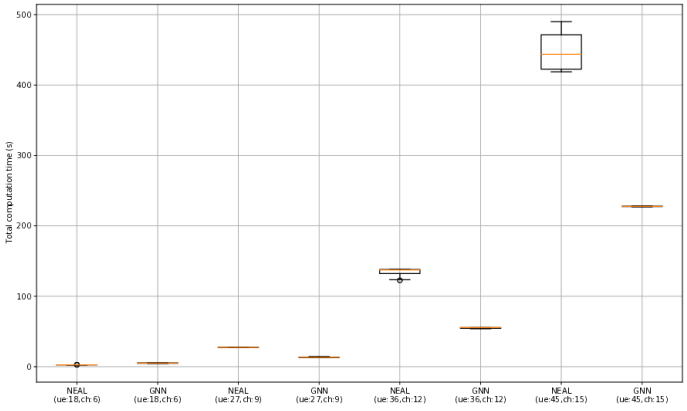

Figure 3 Comparison between the execution time of the PyQUBO (NEAL) solver and the Graph Neural Network (GNN) methods on various sizes of the NOMA allocation problem

2 最大独立集合問題の求解高速化

2-1 ハイブリッド制約方式と Swap-based Simulated Annealing による探索改善

本研究の第一の貢献は、ハイブリッド制約処理戦略（Hybrid Constraint Handling）の提案である。これは、QUBO に含まれる制約のうち、サブキャリアごとの one-hot 制約をソルバーの探索機構そのもので常に満たすようにし、UE 重複制約のみをペナルティとして QUBO に残すという分離戦略である。探索空間が大幅に縮小されることを示している。

第二の貢献は、この戦略を実装するための Swap-based Simulated Annealing アルゴリズムの提案である。Swap 操作は、同一サブキャリア内でアクティブ変数と非アクティブ変数を入れ替えることで、常に one-hot 制約を満たす。この戦略により、PyQUBO 従来の「Flip」操作では頻発していた非実行可能状態への遷移を one-hot 制約内の「Swap」操作によって完全に排除し、収束を大幅に高速化する（Figure 4）。

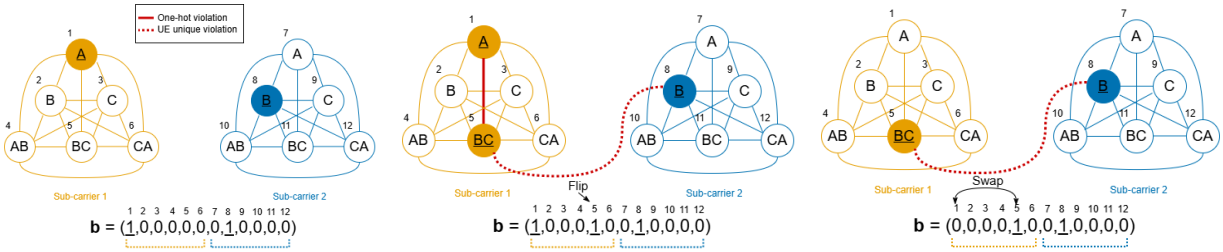

Figure 4 Formulation of the NOMA allocation problem as a Maximum Weighted Independent Set problem with three UEs (A, B, C) and two sub-carriers. The figure on the left shows an initial state with two candidate nodes selected for the MWIS. The center figure shows the result of the classic “flip” operation on node BC, causing two constraints to be broken (shown in red). On the contrary, the “swap” operation we propose (shown in the right figure) selects BC at the same time node A is deselected within the one-hot constraint constituted by the Sub-carrier 1, resulting in a single constrain to be broken during the exploration.

2-2 FPGA による小規模グラフの高速求解

第三の貢献は、この Swap-based SA を Xilinx Zynq UltraScale+ ZCU104 FPGA 上に実装し、実時間処理に近い高速化を達成した点である。FPGA 実装は、CPU 実装 C++版の 5.11 倍、PyQUBO の 49.2 倍の高速化を達成した（Figure 5）。さらに、FPGA の消費電力は約 3.63 W と低く、基地局のような電力制約環境に適していることも示した。上記 Swap-based SA 及び FPGA による加速化は国際会議にて発表済みである[4]。

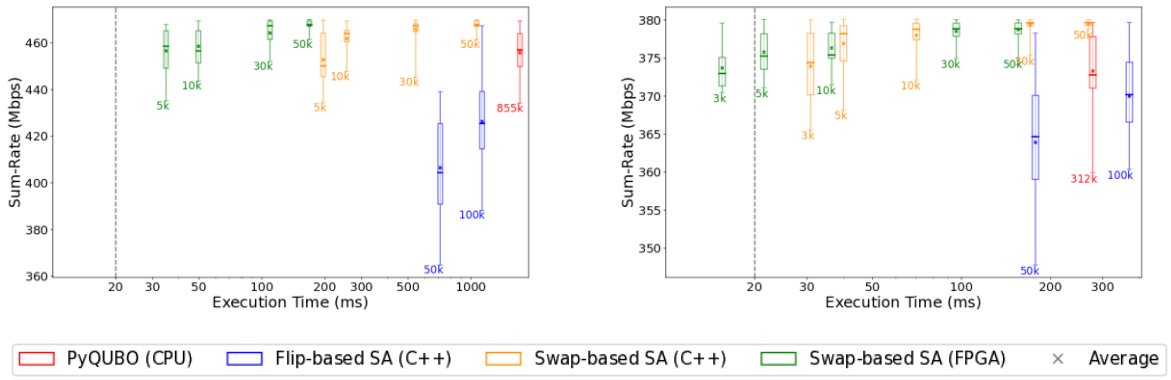

Figure 5 Execution time and solution quality trade-off (towards the left and higher is better, labels show the number of iterations). On the left, 312 node graph (12 UEs, 4 sub-carriers), on the right 855 node graph (18 UEs and 5 sub-carriers)

チャネルコヒーレンス時間は、実用的なスケジューリングにおいて約 20 ms というリアルタイム制約を課す本研究の結果から、提案する FPGA 実装のスワップ型 SA ソルバーは、設定によってはこのコヒーレンス時間内に処理を完了できることが分かる。具体的には、インスタンス A では 3000 イテレーションで 15.5 ms で終了し、379.6 Mbps のスループットを達成している。これは、1 つのコヒーレンス時間内に高品質な解を得られることを示している。

一方で、もう少し規模の大きい問題では、FPGA 上であっても実行時間が 20 ms を超えてしまう（例：5000 イテレーションで 34.8 ms）。ある程度高性能の Xilinx U50 FPGA を用いることでこの点を緩和できるが(Table 1)、問題の規模によって引き起こされる根本的な問題を解決するものではない。

したがって、提案手法は小規模～中規模の割り当て問題に対しては十分高速であるものの、大規模ネットワークでリアルタイム動作を実現するには、複数の FPGA に UE やサブキャリアを分散させるなどの並列化が必要となる。

Table 1 Computation time of the Swap-based Simulated Annealing solver depending on the FPGAS used

Iterations	ZCU104 (ms)	Alveo U50 (ms)
5,000	34.829	14.090
10,000	49.745	24.531
30,000	109.354	66.142
50,000	168.928	107.589

3　グラフ彩色問題への展開

3-1 グラフ彩色問題について

グラフ彩色問題は、グラフの「頂点（ノード）」に色をつけるとき、つながっている頂点同士が同じ色にならないように色を割り当てる問題である。たとえば、A と B という 2 つのノードが線でつながっている場合、A と B は同じ色を使ってはいけない。このように、隣り合うノードに異なる色を割り当てることがグラフ彩色の目的である(Figure 6)。

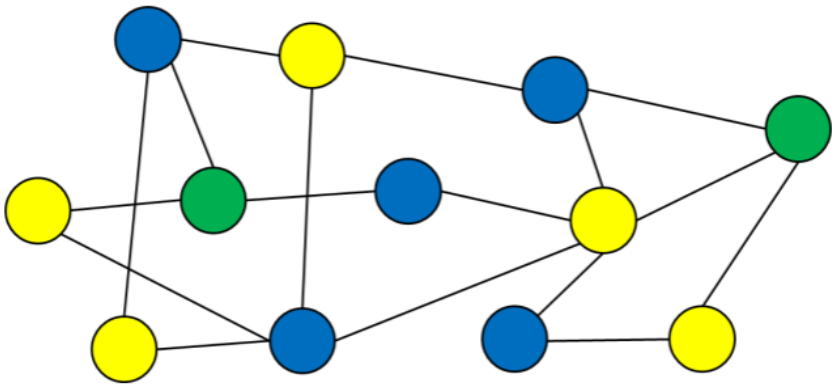

Figure 6 Example of a graph coloring problem. In this example, three colors are needed to solve the problem.

グラフ彩色は、現実のさまざ

まな「割り当て問題」を表すのに使うことができる。その理由は、ノードを「ある決定」や「ある選択肢」として表し、エッジを「同時に選んではいけない関係」として表せるからである。たとえば次のように考えられる：

- ノード：ある資源を使う候補（例：ある時間帯に計算や通信資源を使いたい端末）
- エッジ：同時に割り当てると問題が起きる組み合わせ（例：同じ資源を同じ時間に使いたい端末

このとき、ノードに割り当てる「色」は、その資源をどのグループ（時間帯・場所・周波数など）に割り当てるかを表すことができる。

つまり、同じ色を使うノードは、同じ資源を共有することになる。エッジでつながっているノードは同じ色を使えないので、同じ資源を同時に使えないという関係が自然に表現でき、最大独立集合と同様、情報通信における様々の問題に向いていると言える。

シミュレーテッド・アニーリングによるグラフ彩色では、まずノードに適当な色を割り当て、そこから少しずつ色を変えて衝突を減らしていく。ここでいう衝突とは、隣り合うノードが同じ色を使ってしまう状態のことを指す。アルゴリズムは、ランダムにノードを選び、そのノードの色を別の色に変更してみます。もし変更によって衝突が減れば、その変更を採用する。衝突が増える場合でも、一定の確率で採用し、より良い解にたどり着く可能性を残す。この確率は「温度」と呼ばれる値によって決まり、温度が高い間は悪い変更も受け入れやすく、温度が下がるにつれて安定した解に収束する。最終的に、隣接ノードが同じ色を使わない彩色に近づくように、反復的に改善を続ける。

3-2 グラフ分割による並列探索

大きなグラフをそのまま彩色しようとすると、ノード数やエッジ数が膨大になり、計算時間が非常に長くなる。そこで、グラフをいくつかの小さな部分グラフに分割し、部分グラフは元のグラフよりもずっと小さいため、彩色に必要な計算量が減り、処理が速くなる。これは、クラスタ計算やマルチコア CPU、GPU、FPGA など、並列処理が得意なハードウェアと相性が良い方法である。大規模なネットワークや通信システムのように、リアルタイム性が求められる場面では、この「分割して並列に処理する」戦略が大きなメリットになる。

しかし、グラフを分割するには「どこで切るか」を決める必要がある。分割の仕方が悪いと、部分グラフ同士の境界が多くなり、後で整合性を取るのが難しくなる。たとえば、境界に多くのエッジがあると、部分グラフごとに彩色した結果が互いに矛盾しやすくなる。また、部分グラフを独立に彩色した後、境界にあるノードの色の整合性が問題になる。元のグラフでは、境界にあるノード同士がエッジでつながっている場合がある。部分グラフごとに独立に彩色すると、境界のノードが同じ色を使ってはいけないのに同じ色になってしまうことがある (Figure 7)。この矛盾を解消するには、部分グラフ同士が境界ノードの色を互いに伝え合い、調整する仕組みが必要ではあるが、次のような問題が生じる：

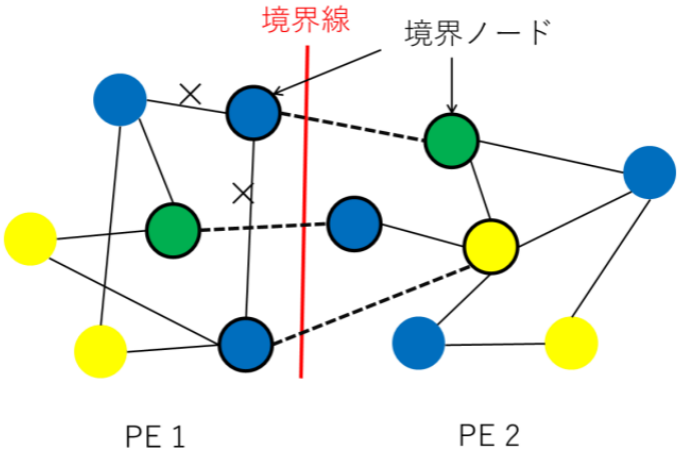

Figure 7 Graph Partitioning in a Graph Coloring Problem. The color of the nodes located at the border between the two subgraphs must be communicated to the neighboring Processing Element.

- どのタイミングで色を共有するか（途中か、最後か）
- どちらの部分グラフの色を優先するか
- 矛盾が起きた場合、どの部分を再計算するか
- 通信コストや同期の遅延が発生する

特に並列処理では、境界ノードの情報をやり取りするために通信が必要になり、オーバヘッドの要因になるため全体の速度を下げる原因になる。

3-3 FPGA によるグラフ彩色問題の並列シミュレーテッド・アニーリング

（１）グラフ分割について

グラフの分割処理に METIS ライブラリ[5]を利用している。METIS は、大規模グラフを効率よく複数の部分グラフに分割するための高性能なパーティショニングツールであり、境界エッジをできるだけ少なくしつつ、各部分のサイズを均等に保つよう最適化する。

METIS によって分割された部分グラフは、それぞれ独立した計算単位として扱うことができる。この特徴を活かし、同じ FPGA（Alveo U50）上に配置された複数の独立した処理要素（Processing Elements-PE）に、各部分グラフの彩色処理を割り当てている。これにより、複数の部分グラフを並列に彩色できる。

（２）並列探索

まず、グラフを 2 つに分割した場合には、各 PE が相手の PE と直接 FIFO キューでつながる方式を採用した（Figure 8）。この方式では、境界に位置するノードの色や状態を、FIFO を通じてメッセージとして直接送信する。構造が単純で、2 分割のような小規模な場合には高速かつ効率的に動作する。

しかし、この方式は PE の数が増えると急速にスケールしなくなるという問題がある。PE が増えるたびに、各 PE は他のすべての PE と通信するための FIFO を持つ必要があり、PE 数が N のとき FIFO の数は $N\times(N-1)$ に増えてしまう。このため、大規模な並列化には適していない。

この問題を解決するため、2 つ目の設計では 中央ルーター（Router）を用いた通信方式を採用した（Figure 9）。各 PE は、境界ノードの色や状態を中央ルーターに送信し、ルーターが必要な PE にだけその情報を転送する。この方式では、PE 同士が直接つながる必要がなく、通信経路が「PE → ルーター → 必要な PE」へと一本化されるため、PE の数が増えても通信構造が複雑にならない。結果として、より多くの部分グラフを同時に処理する大規模な構成にも対応できる。

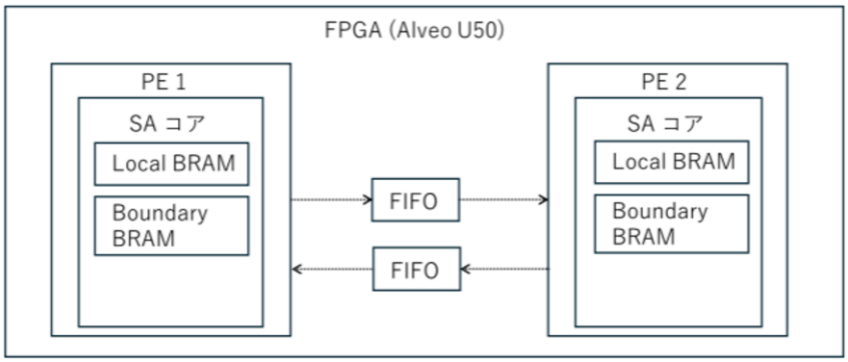

Figure 8 Direct FIFO Queue communication between PEs for parallel simulated annealing on FPGA

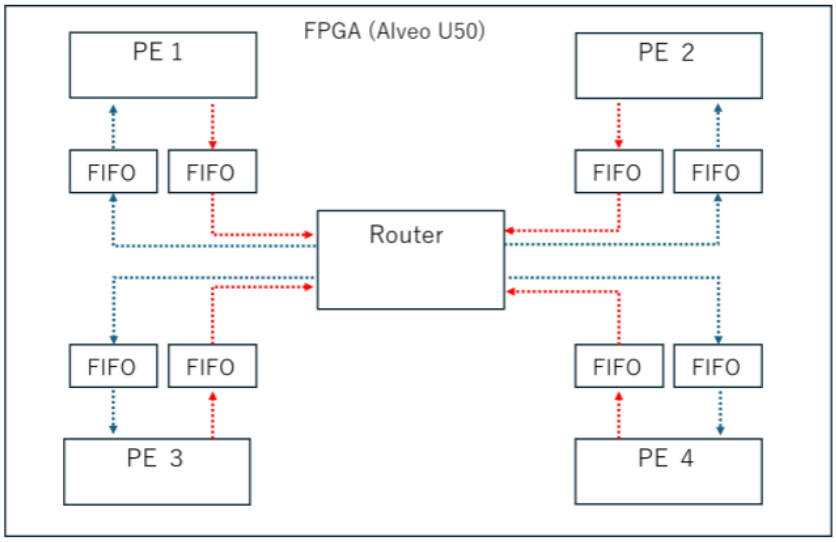

Figure 9 Central router design for communication between PEs

（３）最善解の保存について

その他の大きな課題の一つは、FPGA 上の複数の PE（Processing Element）が、それぞれ自分のローカルな衝突数しか把握できない点にある。各 PE は独立して動作しているため、全体としてのグローバル衝突数がいくつなのかを個別に知ることができない。その結果、どのタイミングで「最善解を更新すべきか」を判断できないという問題が生じる。さらに、最善解を保存する際にも別の問題が発生しる。ある PE が「今が最善」と判断して保存を開始しても、その保存処理の途中で別の PE がノードの色を書き換えてしまう可能性がある。このような状況では、保存された最善解が、実際には異なる時刻の断片が混ざった不整合な状態になってしまう。

これらの問題を解決するために、加算木（Adder Tree）を用いてグローバル衝突数を計算する仕組みを導入した。各 PE は毎クロック、自分のローカル衝突数を出力し、加算木がそれらを固定レイテンシで集約してグローバル衝突数を算出する。これにより、全体の衝突数を正確かつ高速に把握できるようになる。

また、最善解の保存には専用のモニタ回路を用いる。この回路は、グローバル衝突数が現在の最善値より小さくなった瞬間を検知し、Save 信号を全 PE にブロードキャストする。Save 信号を受け取った PE は、現在保持している色の情報を Shadow RAM に一斉に転送する。もし保存中に PE が色を書き換えそうな場合には、全 PE を一時的にストールさせて整合性を保つようにする。これらの処理は、SA のメインループを一切停止させることなく実行される。最善解の保存用の回路をFigure 10に示す。

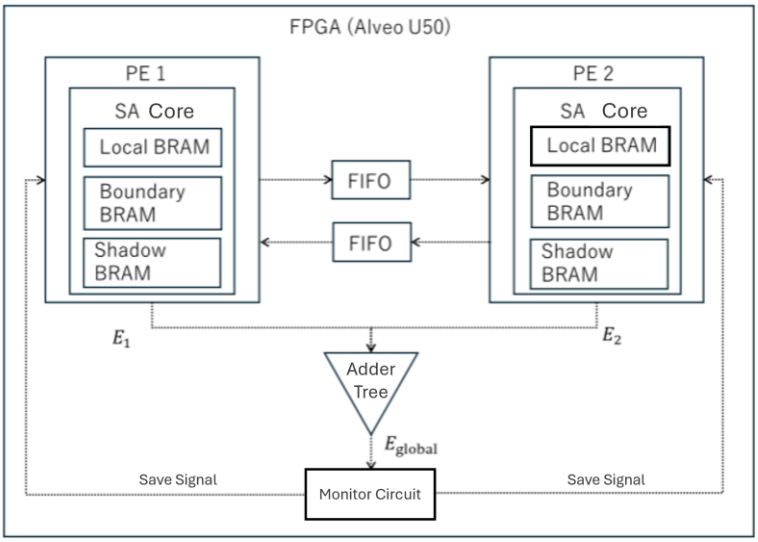

Figure 10 Detail of the FPGA circuit necessary for the global solution retention

3-4 性能評価

我々が提案したソルバーの性能評価にあたって、公開されているベンチマークを用いた[6]。結果一部を

Table 2 に示す。

Table 2 Avg number of conflicts and execution time (ms) of our FPGA-based solvers on various graph coloring benchmarks. The number of iterations is set identical for all solvers at 500,000 iterations and each data point is averaged on 10 executions.

Graph	Nodes	Edges	Colors	Iterations	Base (1PE)	2PE	4PE
fpsol2.i.1	496	11654	65	500000	6.1 / 159.07	5.5 / 204.93	※2
fpsol2.i.2	451	8691	30	500000	12.8 / 156.82	10.9 / 203.10	9.4 / 252.71
fpsol2.i.3	425	8688	30	500000	12.4 / 157.21	10.6 / 209.18	9.6 / 254.04
inithx.i.1	864	18707	54	500000	12.9 / 158.39	11.2 / 204.25	10.1 / 238.16
inithx.i.2	645	13979	31	500000	27.6 / 157.71	25.4 / 205.72	24.2 / 267.56

Graph	Nodes	Edges	Colors	Iterations	Base (1PE)	2PE	4PE
inithx.i.3	621	13969	31	500000	27.3 / 157.98	25.8 / 207.37	23.9 / 270.52
le450_15a	450	8168	15	500000	13.6 / 156.35	11.2 / 194.42	9.2 / 207.43
le450_15b	450	8169	15	500000	13.0 / 156.39	10.6 / 194.27	8.9 / 207.77
le450_15c	450	16680	15	500000	183.6 / 164.27	175.1 / 199.26	165.4 / 208.93
le450_15d	450	16750	15	500000	186.1 / 164.44	176.3 / 199.31	165.4 / 208.75
le450_25a	450	8260	25	100000	3.6 / 31.50	1.9 / 41.11	0.9 / 45.79
le450_25b	450	8263	25	50000	4.9 / 15.89	2.3 / 20.51	1.1 / 23.06
le450_25c	450	17343	25	500000	31.8 / 164.89	26.9 / 203.55	23.5 / 216.46
le450_25d	450	17425	25	500000	31.2 / 165.02	26.6 / 202.57	23.0 / 215.91
le450_5a	450	5714	5	500000	57.2 / 154.11	44.6 / 185.85	31.8 / 192.13
le450_5b	450	5734	5	500000	60.3 / 154.11	47.8 / 185.74	34.4 / 191.94
le450_5c	450	9803	5	500000	44.5 / 157.87	35.0 / 189.45	34.0 / 196.88
le450_5d	450	9757	5	500000	51.5 / 158.00	40.6 / 189.27	31.5 / 196.92
mulsol.i.1	197	3925	49	50000	1.4 / 15.93	1.0 / 20.39	1.8 / 23.00
mulsol.i.2	188	3885	31	100000	5.1 / 31.57	4.3 / 41.35	6.7 / 48.96
mulsol.i.3	184	3916	31	100000	5.1 / 31.62	4.5 / 41.59	6.8 / 49.57
mulsol.i.4	185	3946	31	100000	5.2 / 31.60	4.6 / 41.62	7.0 / 49.73
mulsol.i.5	186	3973	31	100000	5.3 / 31.60	4.5 / 41.63	7.1 / 49.59
zeroin.i.1	211	4100	49	100000	3.0 / 31.60	2.9 / 40.11	※2
zeroin.i.2	211	3541	30	100000	5.4 / 31.31	4.7 / 40.46	※2
zeroin.i.3	206	3540	30	100000	5.2 / 31.34	4.8 / 40.74	5.5 / 48.59
anna	138	493	11	5000	2.0 / 1.59 (1.68)	1.5 / 2.47	1.0 / 2.75
david	87	406	11	5000	0.8 / 1.28 (1.66)	0.5 / 2.50	0.1 / 2.97
homer	561	1629	13	10000	※1	19.5 / 4.33	16.1 / 4.66
huck	74	301	11	5000	0 / 0.44	0 / 2.44	0 / 2.75
jean	80	254	10	5000	0 / 0.38	0 / 2.43	0 / 2.65
games120	120	638	9	5000	0 / 0.72	0 / 2.43	0.1 / 2.68
miles1000	128	3216	42	40000	3.4 / 13.72	2.6 / 17.78	5.2 / 19.87
miles1500	128	5198	73	20000	2.8 / 7.39 (7.49)	2.3 / 10.07	6.2 / 12.83
miles250	128	387	8	5000	1.1 / 1.39 (1.65)	0.3 / 2.40	0.1 / 2.58
miles500	128	1170	20	10000	2.1 / 3.16 (3.29)	1.6 / 4.43	1.2 / 4.85
miles750	128	2113	31	10000	4.5 / 3.41	3.5 / 4.62	3.5 / 5.20
queen11_11	121	3960	11	500000	28.8 / 162.46	27.7 / 195.02	26.0 / 203.70
queen13_13	169	6656	13	500000	37.0 / 165.18	36.1 / 198.52	34.8 / 207.20
queen5_5	25	160	5	5000	1.2 / 0.58 (1.61)	0.3 / 2.39	0.3 / 2.57
queen6_6	36	290	7	10000	2.5 / 3.14 (3.22)	2.2 / 4.29	1.8 / 4.56
queen7_7	49	476	7	100000	11.4 / 28.69 (31.51)	10.5 / 38.26	8.3 / 39.87
queen8_12	96	1368	12	10000	6.0 / 3.37 (3.38)	3.0 / 4.45	1.4 / 4.70
queen8_8	64	728	9	50000	4.1 / 15.42 (15.90)	3.5 / 19.71	3.0 / 20.57
queen9_9	81	2112	10	50000	5.3 / 15.99 (16.15)	4.3 / 19.96	3.8 / 20.74
myciel3	11	20	4	5000	0 / 0.07	0 / 2.48	0.1 / 2.82
myciel4	23	71	5	5000	0 / 0.18	0.1 / 2.48	0 / 2.81
myciel5	47	236	6	5000	0 / 0.34	0.1 / 2.48	0.1 / 2.82
myciel6	95	755	7	5000	0 / 0.8	0 / 2.51	0 / 2.85
myciel7	191	2360	8	5000	1.0 / 1.66 (1.76)	0.4 / 2.54	0.2 / 2.91

※1 Inconsistency between CPU and FPGA number of conflicts
※2 Error

本実験では、PE（Processing Element）の数を増やした際に発生するオーバーヘッドと、解の質への影響を評価した。その結果、PE 数の増加に伴い実行時間が長くなることが確認された。これは、複数の PE が並

列に動作する際、境界ノードの情報共有や最善解保存のために同期処理が必要となり、PE 間のストールが発生するためである。同じ反復回数で比較すると、4PE 構成は 2PE や 1PE よりも実行時間が長くなっており、並列化に伴う制御コストが無視できないことが分かる。

一方で、解の質（残存衝突数）は、PE 数を増やすことで改善される傾向が強い。これは、複数の PE が独立した部分グラフを同時に探索することで、より多様な探索経路が生まれ、局所解に陥りにくくなるためである。多くのデータセットで、4PE 構成は 2PE や 1PE よりも衝突数が少なく、並列探索が解の質向上に寄与することが確認された。これは、部分グラフを独立に探索する並列 SA の利点が実際に発揮されていることを示している。

しかしながら、すべてのグラフでこの傾向が当てはまるわけではない。特に mulsol.i 系列、zeroin.i.3、miles 系列の一部では、4PE の最終衝突数が 2PE よりも悪化しているケースが見られた。これは、グラフ構造や分割結果によっては、部分グラフ間の依存関係が強く、境界ノードの同期コストや探索の偏りが逆効果になる場合があることを示唆している。総じて、本実験は次の 2 点を明確に示している:

- PE 数を増やすと同期オーバーヘッドが増大し、実行時間は長くなる
- 多くのケースで解の質は向上するが、特定のグラフでは逆効果となる場合もある

これらの結果は、並列 PE による部分グラフ探索が一般的には有効である一方、グラフの性質や分割方法によって性能が左右されることを示しており、今後の最適な分割戦略や同期方式の検討が重要であることを示している。

5 まとめ

本研究は、NOMA における組合せ最適化問題と、大規模グラフに対する高速探索技術を統合的に扱い、通信システムの実用的最適化と FPGA による高速化を両立する新しい枠組み 3 つの研究成果が上げられる：

- NOMA の公平性・消費電力を考慮した新しい QUBO 定式化の提案

既存の QUBO 定式化では通信容量最大化のみが目的であり、消費電力が過大となる問題があった。本研究では、QoS を満たしつつ総消費電力を最小化する Minimize Power 定式化を新たに導入し、都市部マクロセル環境のシミュレーションにおいて 約 50% の電力削減を達成した。また、電力削減にもかかわらず高いスループットが維持されることを示し、目的関数設計の重要性と QUBO の柔軟性を明確にした。

- 最大独立集合問題に対する高速探索手法の開発

QUBO の制約構造を分析し、one-hot 制約を探索機構側で常に満たす ハイブリッド制約処理戦略を提案した。さらに、非実行可能状態への遷移を排除する Swap-based Simulated Annealing を導入し、従来の Flip 操作より大幅に高速な収束を実現した。この手法により、探索空間が縮小され、PyQUBO に比べて桁違いの高速化が可能となった。

- FPGA による高速化と並列化の実現

提案した Swap-based SA を FPGA（ZCU104 および Alveo U50）に実装し、CPU 実装比で最大 49 倍の高速化を達成した。特に小規模〜中規模問題では、チャネルコヒーレンス時間（約 20 ms）以内での求解が可能であり、リアルタイムスケジューリングへの適用可能性を示した。

また、METIS によるグラフ分割と複数 PE による並列探索を導入し、FPGA 上での大規模グラフ彩色の高速化を実現した。さらに、境界ノードの整合性維持、最善解保存のための加算木・モニタ回路など、並列探索に必要なハードウェア機構を新規に設計した。

今後の展望としては、グラフ構造に応じた適応的分割、境界ノード通信の最適化により、並列探索の安定性と性能向上を図る。

【参考文献】

[1] P. Finnerty, H. Nakano, C. Ohta, and F. Ishizaki, “Efficient QUBO Formulation for Non-Orthogonal Multiple Access,” in *2025 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA: IEEE, Jan. 2025, pp. 1–4. doi: 10.1109/ICCE63647.2025.10930024.

[2] 山下峻平, 朱致儀, フィネルティパトリック, and 太田能, “非直交多元接続における消費電力を考慮したパケット送信スケジュール最適化,” presented at the 電子情報通信学会 総合大会, Mar. 2026.

[3] M. J. A. Schuetz, J. K. Brubaker, and H. G. Katzgraber, “Combinatorial optimization with

physics-inspired graph neural networks," *Nat. Mach. Intell.*, vol. 4, no. 4, pp. 367–377, Apr. 2022, doi: 10.1038/s42256-022-00468-6.
[4] K. Takai, S. Yamashita, J. Wang, Z. Zhu, P. Finnerty, and C. Ohta, "An FPGA-Accelerated Maximum Weight Independent Set Solver for Non-Orthogonal Multiple Access Resource Allocation," in *2026 IEEE International Conference on Consumer Electronics (ICCE)*, Dubai, United Arab Emirates: IEEE, Feb. 2026, pp. 1–6. doi: 10.1109/ICCE67443.2026.11449675.
[5] *METIS*. University of Minnesota. Accessed: Jun. 30, 2026. [Online]. Available: https://github.com/KarypisLab/METIS
[6] "Graph Coloring Benchmark Instances." Accessed: Jun. 30, 2026. [Online]. Available: https://roars.dev/npbench/graphcoloring.html

〈発　表　資　料〉

題　　名	掲載誌・学会名等	発表年月
An FPGA-Accelerated Maximum Weight Independent Set Solver for Non-Orthogonal Multiple Access Resource Allocation	IEEE International Conference on Consumer Electronics (ICCE)	2026 年 2 月
非直交多元接続における消費電力を考慮したパケット送信スケジュール最適化	電子情報通信学会 総合大会	2026 年 3 月