Classical and Quantum Heuristics for the Binary Paint Shop Problem

V. Vijendran[1,2⋆], Dax Enshan Koh[1,3,4], Ping Koy Lam[1,2], and Syed M. Assad[1,2]

[1] Quantum Innovation Centre (Q.InC), A*STAR, Singapore
[2] Centre for Quantum Computation and Communication Technologies (CQC2T), Australian National University, Australia
[3] Institute of High Performance Computing (IHPC), A*STAR, Singapore
[4] Science, Mathematics and Technology Cluster, Singapore University of Technology and Design, Singapore

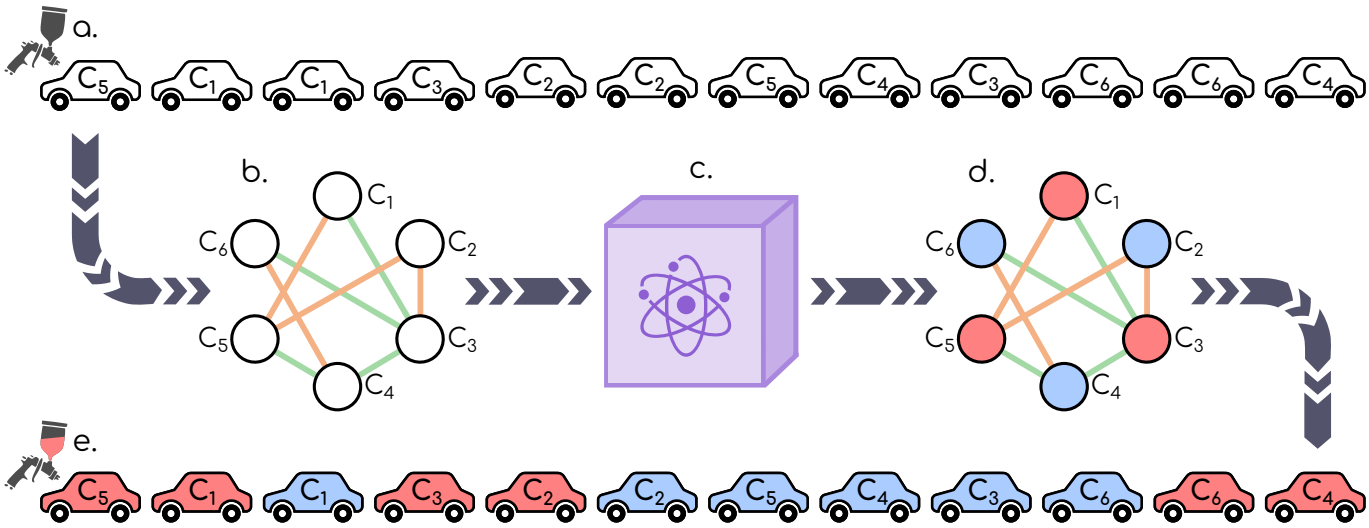


Fig. 1: **Solving the BPSP on a Quantum Computer Using the ICC Encoding.** This figure illustrates the process of solving the BPSP with the Initial-Car-Colour (ICC) encoding using QAOA and its variants. a) The process starts with a BPSP instance, which in this example consists of 6 cars of length 12 forming the sequence $C_5C_1C_1C_3C_2C_2C_5C_4C_3C_6C_6C_4$. b) Using the ICC encoding the BPSP instance with 6 cars is mapped to an Ising model with 6 spins. In the depicted graphs, orange edges symbolise an edge weight of +1, while green edges indicate an edge weight of -1. c) The Ising model is then solved on quantum computer using QAOA or its variants. d) This is the ground state solution of the Ising model, where the colour of node (based on the spin) determines the colour of the first occurrence of the car; using this colour of the second occurrence of the car is painted the opposite colour. e) This is the solved BPSP instance with a total of 4 paint swaps.

The Binary Paint Shop Problem (BPSP) [5, 4, 6], a well-known APX-hard optimisation problem with significant applications in automotive manufacturing, aims to colour a sequence of $2n$ cars, where each of n unique cars appears exactly twice, using two colours such that each car receives both colours while minimising consecutive colour swaps. The paint swap ratio, defined as the average number of colour changes per car, serves as a key performance metric, with lower ratios enhancing production efficiency, reducing material waste, and increasing profitability. Recently, Streif et al. [8] demonstrated that the Quantum Approximate Optimisation Algorithm at $p = 7$ (QAOA_7) achieves an expected paint swap ratio of 0.393, outperforming the classical Recursive Greedy (RG) [2, 1] heuristic's ratio of 0.4. A newer classical heuristic, Recursive Star Greedy (RSG) [7], was conjectured to achieve a ratio of 0.361. In this study, we investigate two state-of-the-art QAOA—eXpressive QAOA (XQAOA) [9] and Recursive QAOA (RQAOA) [3]—and compare their performance against best known classical heuristics. We first establish a rigorous reduction of the BPSP sequence of $2n$ cars to a graph colouring problem on n vertices, which we term the Initial-Car-Colour (ICC) encoding. This reduction not only offers a novel structural perspective on the problem but also reduces the Hamiltonian size by a factor of two. Benchmarking XQAOA_1 and RQAOA_1 on BPSP instances with $N = 2^7$ to 2^{12} cars, we find that XQAOA_1 achieves an average paint swap ratio of 0.357, consistently surpassing RQAOA_1 and classical heuristics across nearly all instances, surpassing even the conjectured performance of RSG. Contrary to previous claims, we empirically estimate RSG's true average ratio to be closer to 0.371. Notably, RQAOA_1 exhibits diminishing performance as the problem size increases. Despite employing provably optimal QAOA_1 parameters at each recursive step, RQAOA_1 is outperformed by RSG on a majority of instances at $N = 2^{11}$ and on all instances at $N = 2^{12}$. To our knowledge, this work is the first to highlight RQAOA_1's performance degradation at scale. In contrast, XQAOA_1 maintains robust performance with a paint swap ratio of 0.357, demonstrating strong potential to outperform all known heuristics in the asymptotic limit and offering a promising quantum approach to the BPSP.

⋆ Corresponding author: vjqntm@gmail.com

References

1. Andres, S.D.: Greedy versus recursive greedy: Uncorrelated heuristics for the binary paint shop problem. Discrete Applied Mathematics **303**, 4–7 (2021). https://doi.org/10.1016/j.dam.2020.05.028
2. Andres, S.D., Hochstättler, W.: Some heuristics for the binary paint shop problem and their expected number of colour changes. Journal of Discrete Algorithms **9**(2), 203–211 (2011). https://doi.org/10.1016/j.jda.2010.12.003
3. Bravyi, S., Kliesch, A., Koenig, R., Tang, E.: Obstacles to variational quantum optimization from symmetry protection. Physical review letters **125**(26), 260505 (2020)
4. Epping, T., Hochstättler, W., Oertel, P.: Complexity results on a paint shop problem. Discrete Applied Mathematics **136**(2-3), 217–226 (2004). https://doi.org/10.1016/S0166-218X(03)00442-6
5. Epping, T., Hochstattler, W., Oertel, P.: Some results on a paint shop problem for words. Electronic Notes in Discrete Mathematics **8**(31-33), 3 (2001). https://doi.org/10.1016/S1571-0653(05)80072-1
6. Gupta, A., Kale, S., Nagarajan, V., Saket, R., Schieber, B.: The approximability of the binary paintshop problem. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings. pp. 205–217. Springer (2013). https://doi.org/10.1007/978-3-642-40328-6_15
7. Hančl, J., Kabela, A., Opler, M., Sosnovec, J., Šámal, R., Valtr, P.: Improved bounds for the binary paint shop problem. In: International Computing and Combinatorics Conference. pp. 210–221. Springer (2023)
8. Streif, M., Yarkoni, S., Skolik, A., Neukart, F., Leib, M.: Beating classical heuristics for the binary paint shop problem with the quantum approximate optimization algorithm. Physical Review A **104**(1), 012403 (2021)
9. Vijendran, V., Das, A., Koh, D.E., Assad, S.M., Lam, P.K.: An expressive ansatz for low-depth quantum approximate optimisation. Quantum Science and Technology **9**(2), 025010 (2024)

Classical and Quantum Heuristics for the Binary Paint Shop Problem

V Vijendran [1,2,3,*] Dax Enshan Koh [2,4,5] Ping Koy Lam [2,1,3] and Syed M Assad [2,3]

[1] Centre for Quantum Technologies (CQT), National University of Singapore, Singapore 117543, Republic of Singapore
[2] Quantum Innovation Centre (Q.InC), Agency for Science, Technology and Research (A*STAR), 2 Fusionopolis Way, Innovis #08-03, Singapore 138634, Republic of Singapore
[3] Centre for Quantum Computation and Communication Technologies (CQC2T), Department of Quantum Science and Technology, Research School of Physics, Australian National University, Acton 2601, Australia
[4] Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR), 1 Fusionopolis Way, #16-16 Connexis, Singapore 138632, Republic of Singapore
[5] Science, Mathematics and Technology Cluster, Singapore University of Technology and Design, 8 Somapah Road, Singapore 487372, Republic of Singapore

arXiv:2509.15294v2 [quant-ph] 2 Jan 2026

The Binary Paint Shop Problem (BPSP) is an APX-hard optimisation problem in automotive manufacturing: given a sequence of $2n$ cars, comprising n distinct models each appearing twice, the task is to decide which of two colours to paint each car so that the two occurrences of each model are painted differently, while minimising consecutive colour swaps. The key performance metric is the paint swap ratio, the average number of colour changes per car, which directly impacts production efficiency and cost. Prior work showed that the Quantum Approximate Optimisation Algorithm (QAOA) at depth $p = 7$ achieves a paint swap ratio of 0.393, outperforming the classical Recursive Greedy (RG) heuristic with an expected ratio of 0.4 [Phys. Rev. A 104, 012403 (2021)]. More recently, the classical Recursive Star Greedy (RSG) heuristic was conjectured to achieve an expected ratio of 0.361. In this study, we develop the theoretical foundations for applying QAOA to BPSP through a formal reduction of BPSP to weighted MaxCut, and use this framework to benchmark two state-of-the-art low-depth QAOA variants—eXpressive QAOA (XQAOA) and Recursive QAOA (RQAOA)—at depth $p = 1$ (denoted XQAOA_1 and RQAOA_1) against the strongest classical heuristics known to date. Across instances ranging from 2^7 to 2^{12} cars, XQAOA_1 achieves an average ratio of 0.357, surpassing RQAOA_1 and all known classical heuristics, including the conjectured performance of RSG. Surprisingly, RQAOA_1 shows diminishing performance as the problem size increases: despite employing provably optimal QAOA_1 parameters at each recursive step, it is outperformed by RSG on most 2^{11}-car instances and on all 2^{12}-car instances. To our knowledge, this is the first study to report RQAOA_1's performance degradation at scale. In contrast, XQAOA_1 stays robust across all sizes, indicating strong potential to asymptotically surpass all known heuristics.

I. INTRODUCTION

The advent of noisy intermediate-scale quantum (NISQ) devices [1–3] has opened new opportunities toward achieving near-term quantum advantage [4]. Although constrained by limited qubit connectivity, finite coherence times, and the absence of fault-tolerant quantum error correction, such devices can nevertheless execute relatively shallow quantum circuits with reasonable fidelity [5, 6]. These limitations have spurred the development of algorithms designed to maximise the utility of scarce quantum resources, most notably variational quantum algorithms (VQAs) [7–10], which combine quantum state preparation with classical parameter optimisation to adapt to device imperfections while tackling classically hard problems. A prominent example is the Quantum Approximate Optimisation Algorithm (QAOA) [9], which has been extensively studied for combinatorial optimisation [9, 11–21] through numerical benchmarks [22–24], theoretical analyses [25–29], and numerous algorithmic variants [30–39]. Beyond optimisation, QAOA is also computationally universal [40, 41] and capable of generating distributions that are classically hard to sample from [42, 43].

One application of particular interest is the Binary Paint Shop Problem (BPSP) [44–46], an APX-hard optimisation problem arising in automotive manufacturing. The input is a sequence of $2n$ cars comprising n distinct models, each appearing twice. The objective is to paint each car in one of two colours such that the two occurrences of every model receive different colours, while minimising the number of colour transitions between consecutive cars. Performance is quantified by the paint swap ratio—defined as the average number of adjacent pairs of cars painted in different colours—which directly reflects production efficiency and material usage. Streif et al. [47] showed that QAOA at depth $p = 7$ (QAOA_7) achieves an expected paint swap ratio of 0.393, surpassing the Recursive Greedy (RG) algorithm [48, 49], which has a proven expected ratio of 0.4 and was, at the time, the best known classical heuristic for the BPSP. More recently, a new classical heuristic—the Recursive Star Greedy (RSG) heuristic [50]—was proposed, with a conjectured expected ratio of 0.361. While QAOA_7's ability to outperform RG highlights the potential of quantum algorithms for solving complex real-world problems, matching or exceeding the conjectured performance of RSG would require quantum circuits far deeper than

* vjqntm@gmail.com

current devices can sustain. Indeed, noise accumulates rapidly with circuit depth [51–58], and while error mitigation can help [22, 59–65], its overhead often outweighs the modest performance gains [66, 67].

Motivated by the limitations of deep QAOA circuits, we investigate two state-of-the-art low-depth variants—eXpressive QAOA (XQAOA) [34] and Recursive QAOA (RQAOA) [30]—both of which have outperformed the best known classical algorithms for the maximum cut (MaxCut) problem at $p = 1$. We begin by establishing a formal reduction of BPSP to the weighted MaxCut problem and subsequently to an Ising spin model. Central to this reduction is the Initial Car Colour (ICC) encoding, which we formalise based on a technique introduced by Streif et al. [47], that halves the number of decision variables. We further derive exact interaction strengths from BPSP constraints and prove new structural properties of the resulting graphs, including a maximum degree bound and an explicit relation between total graph weight and paint swap cost. Together, these results provide a rigorous foundation for applying QAOA and its variants to BPSP and may be of independent interest for related optimisation problems.

We benchmark RQAOA_1 and XQAOA_1 on random BPSP instances with up to 2^{12} cars, comparing against the best known classical heuristics. Our simulations show that XQAOA_1 achieves an average swap ratio of 0.357, outperforming all tested classical heuristics—including the conjectured 0.361 performance of RSG—as well as RQAOA_1 on most instances. In contrast, RQAOA_1, despite employing provably optimal QAOA_1 parameters at each recursive step, exhibits diminishing performance with problem size, falling behind RSG for 2^{11}-car instances and larger. To our knowledge, this is the first report of RQAOA_1's performance degradation at scale. The consistent robustness of XQAOA_1 across all tested sizes suggests strong potential to surpass all known heuristics in the asymptotic limit.

We note that, during the preparation of this manuscript, a related study by Mooney et al. [68] appeared on the arXiv, also applying RQAOA to the BPSP. Their approach differs from ours in that they avoid direct optimisation of QAOA parameters within each recursion. Instead, they leverage parameter concentration [69] and parameter transfer [70], setting QAOA parameters based on values optimised for other instances. Benchmarking RQAOA for $1 \leq p \leq 3$ on instances of up to 20 cars, they report no noticeable performance loss under parameter transfer. While these results indicate that transferred parameters can sustain RQAOA performance on small instances, our findings reveal that even with fully optimised parameters, RQAOA suffers diminishing returns as problem size increases. Beyond the shared application of RQAOA to the BPSP, the focus and conclusions of our works are distinct.

The remainder of this paper is organised as follows. Section II reviews the BPSP, classical heuristics, and relevant quantum optimisation algorithms. Section III describes the reduction of BPSP to MaxCut and its equivalent Ising formulation. Section IV reports our numerical benchmarking results, while section V analyses their implications and limitations. Finally, section VI concludes with a summary of insights and directions for future work.

II. PRELIMINARIES

A. The Binary Paint Shop Problem

The binary paint shop problem (BPSP) [44–46] is a combinatorial optimisation problem that arises in the paint shops of the automobile industry. A production line receives a fixed sequence of n distinct car models, each appearing exactly twice in the sequence. The order is predetermined by upstream and downstream factory stages—such as the body shop before the paint shop, and trim/chassis/assembly after it—and cannot be rearranged. Each car must be painted in two different colours (e.g., red and blue), but the order in which each car receives its two colours is left to be decided. This assignment, called a *colouring*, determines how colours are applied across the sequence. The objective is to choose a colouring that minimises the number of colour changes between consecutive cars, since every change requires additional setup time, increases paint waste, and reduces throughput.

Formally, given a set of n distinct cars $C = \{c_1, \ldots, c_n\}$, and a sequence of car instances $w = (w_1, \ldots, w_{2n})$ with $w_i \in C$ where each $c \in C$ appears exactly twice, the BPSP asks for a colouring $f = (f_1, \ldots, f_{2n}) \in \{r, b\}^{2n}$ that minimises the total number of colour changes between consecutive cars, subject to the validity constraint that the two occurrences of each car receive opposite colours (i.e., if $w_i = w_j$ and $i \neq j$, then $f_i \neq f_j$). A convenient formulation is:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{2n-1} \left[f_i \neq f_{i+1} \right] \\ \text{subject to} \quad & f_i \in \{r, b\}, \quad \forall i \in \{1, \ldots, 2n\} \\ & f_i \neq f_j, \quad \forall i, j : \; w_i = w_j, \; i \neq j \end{aligned} \tag{1}$$

where $[\cdot]$ denotes the Iverson bracket which evaluates to 1 if its condition holds and 0 otherwise.

The BPSP is NP-complete in its decision form [45] and APX-hard in its optimisation form [44, 46]. In other words, unless P = NP, no polynomial-time classical algorithm can solve all instances exactly, nor approximate them arbitrarily well via a polynomial-time approximation scheme. Moreover, assuming the Unique Games Conjecture, BPSP is NP-hard to approximate within any constant factor [46]. This is in contrast to problems such as MaxCut, which admit polynomial-time classical algorithms that achieve constant-factor approximations [71].

Given these complexities, several heuristic approaches have been proposed for the BPSP, each producing solu-

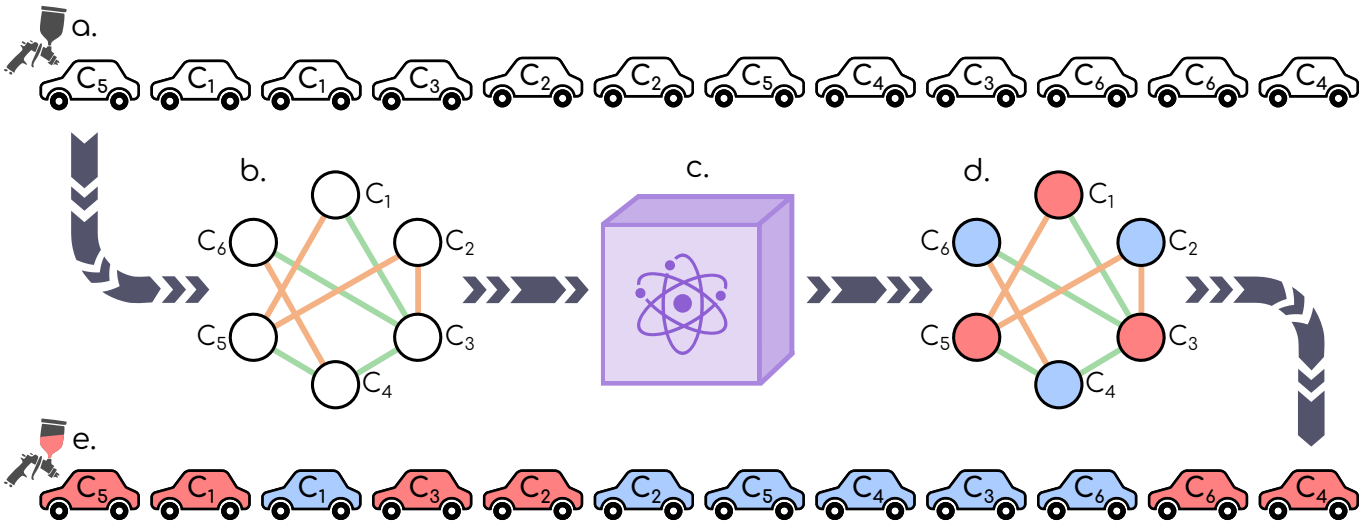


FIG. 1: **Solving the BPSP on a Quantum Computer Using the ICC Encoding.** This figure illustrates the process of solving the BPSP with the Initial-Car-Colour (ICC) encoding using QAOA and its variants. a) The process starts with a BPSP instance, which in this example consists of 6 cars of length 12 forming the sequence $C_5C_1C_1C_3C_2C_2C_5C_4C_3C_6C_6C_4$. b) Using the ICC encoding the BPSP instance with 6 cars is mapped to an Ising model with 6 spins. In the depicted graphs, orange edges symbolise an edge weight of +1, while green edges indicate an edge weight of -1. c) The Ising model is then solved on quantum computer using QAOA or its variants. d) This is the ground state solution of the Ising model, where the colour of node (based on the spin) determines the colour of the first occurrence of the car; using this colour the second occurrence of the car is painted the opposite colour. e) This is the solved BPSP instance with a total of 4 paint swaps.

tions with colour changes linear in n [48, 49, 72]. The simplest is the red-first algorithm [48], which colours the first occurrence of every car red and the second blue. The greedy algorithm [49, 72] improves on this by selecting at each step the colour that minimises immediate paint changes, switching only when necessary to satisfy the requirement that each car be painted in both colours. The recursive greedy (RG) algorithm [48, 49] offers a further improvement by recursively removing the last-occurring car from the sequence, optimally colouring the reduced instance, and then reinserting the removed pair in the best possible way. More recently, Hančl et al. [50] proposed the recursive star greedy (RSG) algorithm, a refinement of RG that introduces a temporary "undecided" colour (star) for car pairs that can be coloured either way without affecting the current swap count; during recursion, if placing a car next to a starred car would otherwise increase the swap count, both occurrences of that starred car are recoloured with binary colours to avoid the increase, and the final colouring is obtained by replacing any remaining stars with arbitrary legal binary assignments. Detailed explanations of these heuristics are provided in appendix B 2. The asymptotic performances of red-first, greedy, and RG are proven, with $\mathbb{E}_{\mathrm{RF}}(\Delta_C) = 2n/3$, $\mathbb{E}_{\mathrm{G}}(\Delta_C) = n/2$, and $\mathbb{E}_{\mathrm{RG}}(\Delta_C) = 2n/5$; in contrast, RSG's performance is unproven but conjectured to be $\mathbb{E}_{\mathrm{RSG}}(\Delta_C) = 0.361n$, an improvement over RG. Despite these advances, numerical results indicate that the optimal number of colour changes grows linearly in n, but with a much smaller constant factor than current heuristics achieve. A recent lower bound of $\mathbb{E}_{\mathrm{OPT}}(\Delta_C) \geq 0.214n - o(n)$ by Hančl et al. [50] confirms that existing heuristics remain far from optimal in the asymptotic limit, leaving room for quantum algorithms to yield improvements.

Heuristic	**Solution**	**# Swaps**
Red-First	$C_5C_1C_1C_3C_2C_2C_5C_4C_3C_6C_6C_4$	7
Greedy	$C_5C_1C_1C_3C_2C_2C_5C_4C_3C_6C_6C_4$	6
RG	$C_5C_1C_1C_3C_2C_2C_5C_4C_3C_6C_6C_4$	5
RSG	$C_5C_1C_1C_3C_2C_2C_5C_4C_3C_6C_6C_4$	4

TABLE I: This table compares the paint sequences and corresponding number of colour swaps for a 6-car BPSP instance, showing the solutions obtained by the classical heuristics together with their swap counts. In this example, red-first produces the highest number of swaps, while RSG achieves the fewest and matches the optimal solution.

We conclude this subsection with an illustrative example of the BPSP, shown in fig. 1 with the corresponding results summarised in table I. The example compares the four classical heuristics described earlier on the same instance. As expected from their asymptotic behaviour, red-first produces the highest number of swaps, while RSG achieves the fewest and, in this case, matches the

optimal solution. It is worth noting that although RG generally outperforms the standard greedy algorithm—as it does in this example—there are instances where the greedy algorithm produces fewer paint swaps [49].

B. Quantum Approximate Optimisation Algorithm

The Quantum Approximate Optimisation Algorithm (QAOA) [9] is a variational quantum algorithm for approximating solutions to combinatorial optimisation problems. The problem is encoded into a problem Hamiltonian H_P, whose ground state represents the optimal solution. QAOA seeks to approximate this ground state by preparing a depth-p variational quantum circuit, parameterised by $2p$ real angles, and optimising them to minimise the expectation value $\langle H_P \rangle$. Measuring the resulting quantum state in the computational basis produces candidate bit strings, from which a low-cost solution is selected.

QAOA alternates between two parameterised unitaries: one generated by the problem Hamiltonian H_P, which encodes the cost function and biases the state toward low-energy configurations, and one by the mixing Hamiltonian H_M, which drives transitions between computational basis states. For an n-qubit instance, we take

$$H_P = \sum_{j=1}^{n} \sum_{k=1}^{n} J_{jk} Z_j Z_k, \quad H_M = \sum_{j=1}^{n} X_j, \tag{2}$$

where Z_j and X_j are Pauli operators acting on qubit j, and J_{jk} are problem-specific couplings.

The corresponding parameterised unitaries are

$$U(H_P, \gamma) = e^{-i\gamma H_P} = \prod_{j,k}^{n} e^{-i\gamma J_{jk} Z_j Z_k}, \tag{3}$$

$$U(H_M, \beta) = e^{-i\beta H_M} = \prod_{j=i}^{n} e^{-i\beta X_j}, \tag{4}$$

with $\gamma \in \mathbb{R}$ and $\beta \in [0, \pi]$.

Starting from the uniform superposition $|+\rangle^{\otimes n}$, the level-p QAOA (QAOA$_p$) is

$$|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle = \prod_{l=1}^{p} U\left(H_M, \beta_l\right) U\left(H_P, \gamma_l\right) |+\rangle^{\otimes n}. \tag{5}$$

The parameters $(\boldsymbol{\beta}, \boldsymbol{\gamma})$ are optimised classically to obtain

$$(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*) = \underset{\boldsymbol{\beta}, \boldsymbol{\gamma}}{\operatorname{argmin}} \langle \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) | H_P | \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) \rangle \ . \tag{6}$$

The optimised state $|\psi\left(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*\right)\rangle$ is then prepared on a quantum computer, measured repeatedly in the computational basis, and the lowest-cost bit string obtained is reported as the solution.

C. Expressive QAOA

The Expressive QAOA (XQAOA) [34] is an overparameterised extension of standard QAOA, designed to increase ansatz expressiveness at shallow circuit depths. While standard QAOA uses a single γ for all terms in H_P and a single β for all terms in H_M, XQAOA assigns independent parameters to each term of both Hamiltonians. This allows different qubits and couplings to be rotated by distinct angles, enabling a richer exploration of the solution space.

XQAOA further augments the standard mixer by introducing an additional $\boldsymbol{\alpha}$-dependent unitary acting on Pauli-Y operators. Let $\boldsymbol{A} = (A_j)_{j=1}^{n}$ denote the collection of single-qubit Pauli-Y operators with $A_j = Y_j$, and let $\boldsymbol{B} = (B_j)_{j=1}^{n}$ denote the corresponding Pauli-X operators with $B_j = X_j$. For the problem Hamiltonian, we write $\boldsymbol{C} = \left(H_{P_{jk}}\right)_{(j,k)\in E}$, where each $H_{P_{jk}} = J_{jk} Z_j Z_k$ corresponds to the Ising interaction on edge (j, k).

The mixing unitaries with parameter vectors $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$ are defined as

$$U(\boldsymbol{A}, \boldsymbol{\alpha}) = e^{-i\sum_{j=1}^{n} \alpha_j A_j} = \prod_{j=1}^{n} e^{-i\alpha_j Y_j}, \tag{7}$$

$$U(\boldsymbol{B}, \boldsymbol{\beta}) = e^{-i\sum_{j=1}^{n} \beta_j B_j} = \prod_{j=1}^{n} e^{-i\beta_j X_j}, \tag{8}$$

where $\alpha_j, \beta_j \in [0, \pi]$ are independent angles acting locally on each qubit. The problem unitary is likewise parameterised per interaction term as

$$U(\boldsymbol{C}, \boldsymbol{\gamma}) = \prod_{j,k}^{n} e^{-i\gamma_{jk} J_{jk} Z_j Z_k}, \tag{9}$$

where $\gamma_{jk} \in \mathbb{R}$ are the parameters associated with each $Z_j Z_k$ term in the problem Hamiltonian H_P.

At $p = 1$, XQAOA prepares the state

$$\begin{aligned} |\psi(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})\rangle &= U(\boldsymbol{A}, \boldsymbol{\alpha}) U(\boldsymbol{B}, \boldsymbol{\beta}) U(\boldsymbol{C}, \boldsymbol{\gamma}) |+\rangle^{\otimes n} \\ &= \prod_{j}^{n} e^{-i\alpha_j Y_j} e^{-i\beta_j X_j} \prod_{j,k}^{n} e^{-i\gamma_{jk} J_{jk} Z_j Z_k} |+\rangle^{\otimes n}. \end{aligned} \tag{10}$$

The XQAOA ansatz admits four notable configurations, distinguished by the constraints placed on the mixing unitary angles. The Multi-Angle QAOA (MA-QAOA) [33] is obtained by setting all $\alpha_j = 0$, yielding a multi-angle extension of the standard X-only QAOA mixer. The XY mixer is the most general form, allowing independent α_j and β_j for each qubit. The Y mixer restricts the ansatz to Y-rotations by setting all $\beta_j = 0$, while the X=Y mixer applies both X and Y rotations with a shared angle on each qubit, enforcing $\alpha_j = \beta_j$. Numerical results indicate that the X=Y mixer outperforms the other three configurations; therefore, all references to XQAOA in this work pertain to the X=Y mixer.

D. Recursive QAOA

The Recursive QAOA (RQAOA) [30] is a non-local variant of the QAOA designed to enhance QAOA's performance, especially at shallow circuit depths. RQAOA leverages QAOA as a subroutine to recursively reduce the problem size until it becomes tractable for brute-force methods. Specifically, RQAOA_1—RQAOA that uses QAOA_1 as its subroutine—has been shown to outperform state-of-the-art classical algorithms for MaxCut and similar problems [30, 73, 74]. Additionally, RQAOA_1 provides performance guarantees for particular graph structures [30, 75, 76] and has been successfully applied to small-scale, real-world problems [77, 78].

Algorithm 1: RQAOA for Ising Models

Input: Problem Hamiltonian C_0 with n variables $V_0 = \{1, \ldots, n\}$, recursion depth η
Output: Approximate ground state of the Hamiltonian C_0

for $t \leftarrow 0$ **to** $n - \eta$ **do**
  Prepare QAOA state $|\psi_t(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle$ for C_t
  Optimise parameters:
$$(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*) \leftarrow \underset{\boldsymbol{\gamma}, \boldsymbol{\beta}}{\operatorname{argmin}} \langle \psi_t(\boldsymbol{\gamma}, \boldsymbol{\beta}) | C_t | \psi_t(\boldsymbol{\gamma}, \boldsymbol{\beta}) \rangle$$
  Compute M_{ij} for all $\{i, j\} \in V_t$:
$$M_{ij} \leftarrow \langle \psi_t(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*) | Z_i Z_j | \psi_t(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*) \rangle$$
  Identify the pair $\{u, v\} \leftarrow \underset{\{i,j\} \in V_t}{\operatorname{argmax}} |M_{ij}|$
  Impose Constraint: $Z_v = \operatorname{sign}(M_{uv}) Z_u$
  Update Variable Set: $V_{t+1} \leftarrow V_t \setminus \{v\}$
  Update Hamiltonian:
$$C_{t+1} \leftarrow \operatorname{sign}(M_{uv}) \sum_{i \in V_{t+1}} J_{iv} Z_i Z_u + \sum_{\{i,j\} \in V_{t+1}} J_{ij} Z_i Z_j$$
end
Solve C_t via brute-force when $|V_t| \leq \eta$
Solve C_0 by backtracking through constraints
return Approximate ground state of C_0

At each recursive step, RQAOA executes QAOA on the current problem Hamiltonian H_P, calculating the correlation metrics $M_{ij} = \langle Z_i Z_j \rangle$ for all pairs of qubits $\{i, j\}$. The pair with the highest absolute correlation, identified as $\{u, v\} = \operatorname{argmax}_{\{i,j\}} |M_{ij}|$, is selected. If $M_{uv} > 0$, the qubits u and v are positively correlated, and the constraint $Z_v = Z_u$ is imposed. Conversely, if $M_{uv} < 0$, the qubits are anti-correlated, leading to the constraint $Z_v = -Z_u$. These constraints effectively eliminate one qubit by expressing its state in terms of the other, thereby reducing the problem size. This process is iteratively applied to the updated Hamiltonian until the problem size is sufficiently small to be solved exactly using brute-force methods. The solution to the reduced instance is then propagated back through the imposed constraints to derive the solution to the original problem. The detailed steps of RQAOA are outlined in algorithm 1.

III. RECASTING BPSP AS AN ISING MODEL

We now present a complete and provably equivalent mapping of the BPSP to an Ising model suitable for quantum optimisation. Our approach formalises the Initial-Car-Colour (ICC) encoding, inspired by Streif et al. [47], which reduces the number of decision variables by half. Using this encoding we construct, for any BPSP instance, a weighted graph whose MaxCut value directly corresponds to the optimal paint swap cost. In particular, we establish a polynomial-time reduction from BPSP to weighted MaxCut, which in turn, provides an exact Ising Hamiltonian whose ground state encodes the optimal solution. The full technical development, with proofs and derivations, is given in the appendix; here we present only the essential ingredients needed for the remainder of the paper.

A. Initial Car Colour Encoding

A BPSP instance is a sequence $x \in (1, \ldots, n)^{2n}$ in which every symbol occurs exactly twice. A colouring of x is given by a string $f \in \{r, b\}^{2n}$, where r (red) and b (blue) denote the two paint colours. The colouring f is valid if, whenever two positions i and j in the sequence correspond to the same car (i.e. $x_i = x_j$), the assigned colours differ ($f_i \neq f_j$). This direct encoding, however, contains redundant information: once the colour of the first occurrence of each car is fixed, the second occurrence must necessarily take the opposite colour. In other words, if the first appearance of car j is assigned colour f_i, then its second appearance is automatically $\neg f_i$, where $\neg$ flips red to blue and vice versa. This motivates the Initial-Car-Colour (ICC) encoding, which records only the colours of the first occurrences. Thus, every valid colouring can be described succinctly by an n-bit string instead of a $2n$-bit one. For a rigorous treatment of this encoding, we refer the reader to appendix C 1.

To formalise this compression, we define the expansion map $\mathcal{E}_n$. Given a BPSP instance x and an ICC colouring $z \in \{r, b\}^n$, the corresponding full colouring is defined as $f = \mathcal{E}_n(x, z) \in \{r, b\}^{2n}$, with entries

$$\mathcal{E}_n(x, z)_i = \neg^{[x_i \in \{x_1, \ldots, x_{i-1}\}]} z_{x_i} \quad \forall i \in [2n], \tag{11}$$

where $[\cdot]$ denotes the Iverson bracket which evaluates to 1 if its condition holds and 0 otherwise. In words, $\mathcal{E}_n$ assigns to each position the colour of that car's first occurrence, flipped if the car has already appeared earlier in the sequence. For example, for the BPSP instance $x = (1, 2, 1, 3, 3, 2)$ and an ICC string $z = rrb$, the expansion $\mathcal{E}_3(x, z)$ yields the valid BPSP colouring $rrbrbb$. The mapping from ICC strings to valid colourings is bijective (see proposition 4), ensuring no information is lost.

With the ICC encoding and the expansion map $\mathcal{E}_n$, the BPSP objective can be expressed directly in terms of ICC variables $z \in \{r, b\}^n$ (see proposition 5). Given a

BPSP instance x and an ICC colouring z, the number of paint swaps is

$$\tilde{\xi}_n(x,z) = \sum_{i=1}^{2n-1} \left[\neg^{\eta(x,i)} z_{x_i} \neq z_{x_{i+1}}\right], \tag{12}$$

where the auxiliary function $\eta(x,i)$ determines whether the pair (x_i, x_{i+1}) contributes a "same-colour" or "opposite-colour" preference. It is defined by

$$\eta(x,i) = [x_{i+1} \in \{x_1, \ldots, x_i\}] \oplus [x_i \in \{x_1, \ldots, x_{i-1}\}], \tag{13}$$

with $\oplus$ denoting the exclusive-or operation. In words, $\eta(x,i) = 1$ when the boundary (x_i, x_{i+1}) consists of one first occurrence and one second occurrence, and $\eta(x,i) = 0$ when both are of the same type. Thus, η encodes the local parity between neighbours, enabling the swap count to be expressed as a quadratic form in the ICC variables and setting the stage for the graph and Ising formulations.

B. BPSP Graph Construction

We associate each BPSP instance with a corresponding BPSP graph whose vertices correspond to cars, edges connect cars that appear consecutively in the sequence, and weights reflect the cost preference for colour assignments.

Given a BPSP instance $x = (x_1, x_2, \ldots, x_{2n})$, we associate to it a weighted graph $G_x = (V_x, E_x, W_x)$. The vertex set is $V_x = \{1, \ldots, n\}$, corresponding to the cars in the alphabet $[n]$. An edge $\{u, v\}$ is included in E_x whenever the symbols u and v appear consecutively somewhere in the word x, provided that $u \neq v$ and the resulting weight is nonzero. The weight of an edge $e \in E_x$ aggregates the contributions of all such adjacencies in the word and is defined as

$$W_x(e) = -\sum_{k=1}^{2n-1} (-1)^{\eta(x,k)} \delta_{e,\{x_k, x_{k+1}\}}, \tag{14}$$

where $\delta_{e,\{x_k,x_{k+1}\}}$ equals 1 if the boundary (x_k, x_{k+1}) corresponds to the unordered pair e and 0 otherwise. In this way, each edge weight records how many times the pair of cars appears as neighbours in the sequence, corrected by the parity term $(-1)^{\eta(x,k)}$ that enforces whether equal or opposite colours are favoured.

The BPSP graph has two particularly useful structural properties. First, we show in proposition 8 that its maximum degree is bounded by four, since each car can be adjacent to at most two distinct neighbours across its two occurrences. Second, as shown by Streif et al. [47], the edge weights take values in $\{\pm 1, -2\}$. As $n \to \infty$, edges of weight -2 vanish in probability, while the remaining couplings occur with limiting probabilities: -1 with probability 2/3 and +1 with probability 1/3.

Beyond local structure, we also identify global properties of the BPSP graph. In proposition 9 we show that its total weight can be expressed compactly in terms of the η-function and the number of double letters. Corollary 10 then relates this quantity directly to the cost of the red-first colouring, revealing that even this simple heuristic is governed by a global invariant of the graph.

Together, these local and global properties establish the BPSP graph as the central combinatorial object underlying the problem. In the next subsection, we exploit this structure to obtain an exact reduction of BPSP to weighted MaxCut.

C. Reduction to Weighted MaxCut

Once the BPSP graph is defined, the connection to MaxCut becomes clear. MaxCut asks for a partition of the vertices of a weighted graph into two sets such that the total weight of the edges between them is maximised. For a given instance x, each ICC assignment $z \in \{r, b\}^n$ induces such a partition of the vertices of G_x : two vertices lie on opposite sides of the cut whenever their ICC colours differ. The weight of this cut is obtained by summing the weights of the edges it separates, making the cut value a direct function of the ICC assignment.

In corollary 11 we show that for a BPSP instance x with graph G_x, the cut weight induced by an ICC assignment z can be expressed as

$$\mathrm{wt}_{G_x}(z) = -\sum_{k=1}^{2n-1} (-1)^{\eta(x,k)} \left[z_{x_k} \neq z_{x_{k+1}}\right]. \tag{15}$$

Building on this, corollary 13 establishes a direct link between cut weight and paint swaps:

$$\tilde{\xi}_n(x,z) = \xi_n\left(\sigma_{\mathrm{RF}}(x)\right) - \mathrm{wt}_{G_x}(z), \tag{16}$$

where σ_{RF} denotes the red-first colouring $f \in \{r, b\}^{2n}$ and ξ_n counts the number of swaps in such colouring. Thus, the BPSP cost can be written as the red-first baseline minus the cut weight achieved by z. Using corollary 10, this relation can also be expressed in terms of global graph invariants such as the number of double letters and the total weight, though we omit the details here for brevity.

This reformulation shows that minimising $\tilde{\xi}_n(x,z)$ is exactly equivalent to maximising $\mathrm{wt}_{G_x}(z)$, i.e. solving weighted MaxCut on the BPSP graph. In corollary 15 we rigorously prove this equivalence and, in algorithm 2, present the polynomial-time reduction from BPSP to weighted MaxCut problem.

D. Ising Model Formulation

The weighted MaxCut objective can be naturally reformulated as the ground-state energy of an Ising Hamiltonian. Using spin variables $z \in \{-1, 1\}^n$, the cut weight

of the BPSP graph can be written as

$$\mathrm{wt}_{G_x}(z) = \frac{-1}{2} \sum_{k=1}^{2n-1} (-1)^{\eta(x,k)} \left(1 - z_{x_k} z_{x_{k+1}}\right). \quad (17)$$

Substituting this expression into eq. (16) and replacing the spin variables by Pauli-Z operators yields a quantum Ising Hamiltonian whose ground state encodes the optimal BPSP colouring.

While this already provides a valid mapping, it is possible to obtain a more direct formulation that avoids referencing the red-first cost. By working directly with the ICC objective defined in eq. (12), proposition 17 shows that for a BPSP instance x and ICC assignment $z \in \{-1, 1\}^n$, the number of paint swaps can be expressed as

$$\tilde{\xi}_n(x, z) = n - \frac{1}{2} - \frac{1}{2} \sum_{i=1}^{2n-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}}. \quad (18)$$

In corollary 18 we further show that the ground state of this Hamiltonian corresponds exactly to the optimal BPSP colouring, i.e., the colouring with the minimal number of paint swaps.

Hence, solving BPSP with quantum heuristics such as QAOA amounts to preparing the ground state of the quantum Ising Hamiltonian

$$H_P(x) = n - \frac{1}{2} - \frac{1}{2} \sum_{i=1}^{2n-1} (-1)^{\eta(x,i)} Z_{x_i} Z_{x_{i+1}}. \quad (19)$$

where the spin variables have been promoted to Pauli-Z operators.

IV. NUMERICAL RESULTS

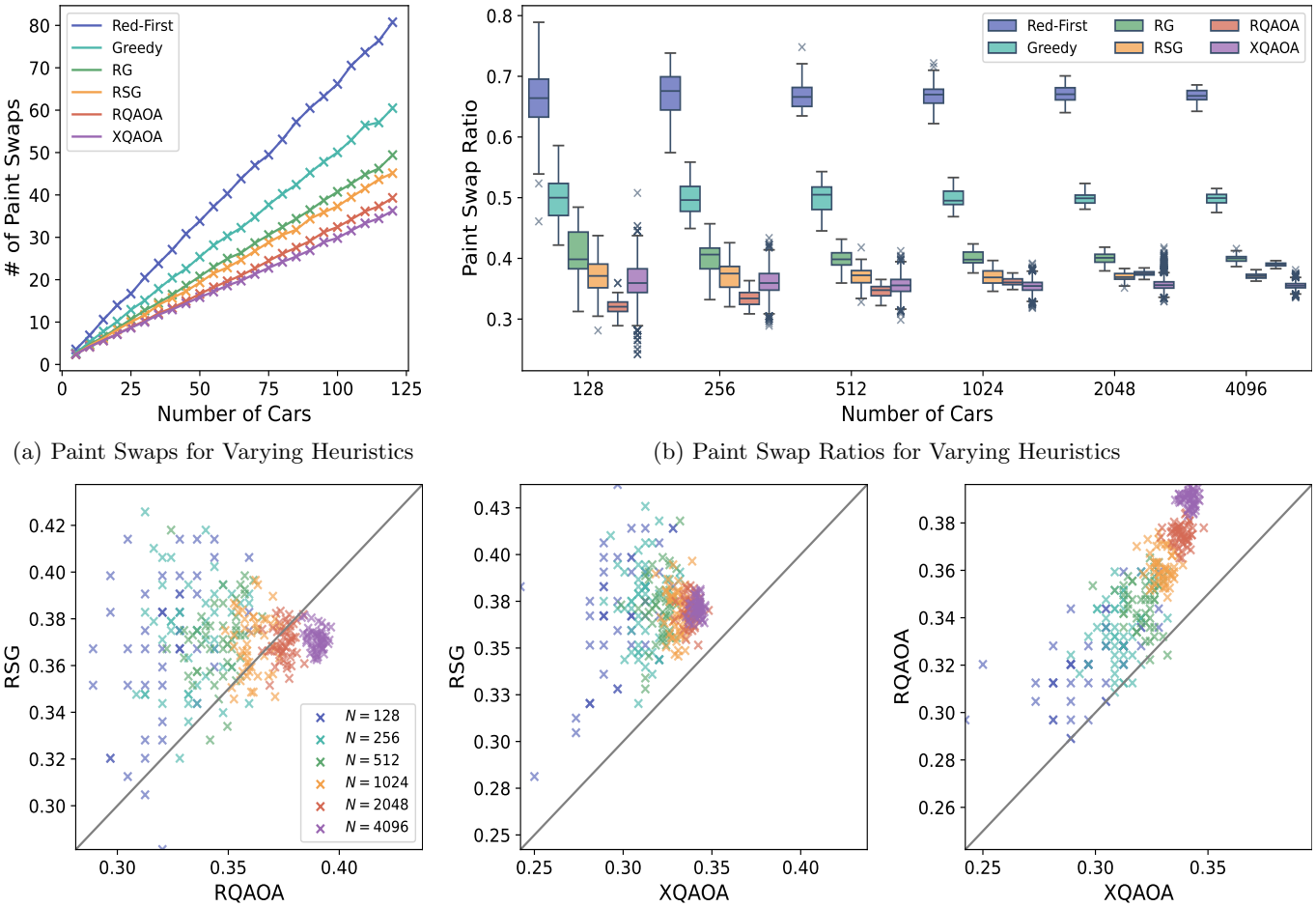
In this section, we benchmark the performance of XQAOA_1 and RQAOA_1 on the Binary Paint Shop Problem, comparing them against the known classical heuristics: Red-First, Greedy, Recursive Greedy, and Recursive-Star Greedy. All BPSP instances were generated randomly using a quantum random number generator [79, 80]. Since both XQAOA_1 and QAOA_1 admit closed-form expressions for their expectation values—given respectively by Theorem 3, Eq. (22), and Theorem 1, Eq. (13) of [34]—all benchmarks were performed entirely on a classical computer, without the use of quantum hardware. For XQAOA_1, the large number of parameters motivated the use of the JAX framework for efficient gradient computation via backpropagation, with optimisation carried out using the Limited-Memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) algorithm [81]. To address the highly non-convex optimisation landscape, each instance was run with 100 random initialisations. For RQAOA_1, the pairwise expectation values required at each recursion step were computed analytically as functions of (β, γ), and the variational parameters were set using the provably near-optimal values obtained by the method of Vijendran et al. [82][1], eliminating the need for multiple restarts; hence, each instance was simulated only once. The deterministic classical heuristics were likewise run once per instance.

We consider two benchmark sets. The first comprises 200 random instances for each $n \in \{5, 10, \ldots, 120\}$, probing how the number of paint swaps scales with problem size. The second consists of 50 random instances for each $n \in \{128, 256, 512, 1024, 2048, 4096\}$, used to compare expected swap ratios $\mathbb{E}(\Delta_C/n)$ on larger instances. Results are summarised in fig. 2 and fig. 3. In reporting results for XQAOA_1, we show the best solution from 100 restarts, except in fig. 2b and fig. 3b, where the distribution across all restarts is presented.

Figure 2a shows the average number of paint swaps on smaller instances. Among classical heuristics, Red-First scales the worst, while Recursive-Star Greedy is the most effective. Both quantum heuristics outperform the classical heuristics, with XQAOA_1 consistently below RQAOA_1 and showing the most favourable scaling. The widening separation between methods as n increases highlights how even small improvements in swap ratio compound into large advantages in the asymptotic regime.

Figure 2b compares paint-swap ratios for larger instances. The classical heuristics behave in line with theoretical expectations: Red-First, Greedy, and Recursive Greedy converge to asymptotic averages near 0.66, 0.5, and 0.4 respectively, while the Recursive-Star Greedy achieves an average of 0.37, slightly higher than its conjectured value of 0.361. XQAOA_1 attains an average ratio of about 0.357, outperforming all classical heuristics. In contrast, RQAOA_1 exhibits size-dependent performance: it is competitive on smaller instances, but its average ratio rises steadily with n, eventually falling behind both XQAOA_1 and Recursive-Star Greedy. This deterioration is evident in fig. 3, where the mean of the RQAOA_1 distributions drifts steadily rightward, increasing from 0.322 at $n = 128$ to 0.335 at $n = 256$, 0.346 at $n = 512$, and 0.362 at $n = 1024$, already worse than the stable XQAOA_1 average of 0.357; by $n = 2048$ the mean reaches 0.375, exceeding the Recursive-Star Greedy benchmark of 0.37, and at $n = 4096$ it rises further to 0.390, with the right tail extending to 0.4, matching the performance of Recursive Greedy. Despite relying on provably near-optimal parameters for QAOA_1 at every recursive step, RQAOA_1 deteriorates consistently with system size. In contrast, XQAOA_1 remains strikingly stable, with its distributions sharply peaked around 0.357

[1] In brief, this method applies to Ising models with arbitrary rational weights, reducing the two-dimensional optimisation over (β, γ) to a one-dimensional search in γ, using the Nyquist–Shannon theorem to set the sampling resolution, and a subdivision algorithm to locate the optimal angle in linear time.



(a) Paint Swaps for Varying Heuristics

(b) Paint Swap Ratios for Varying Heuristics

(c) Swap Ratios for RSG vs RQAOA

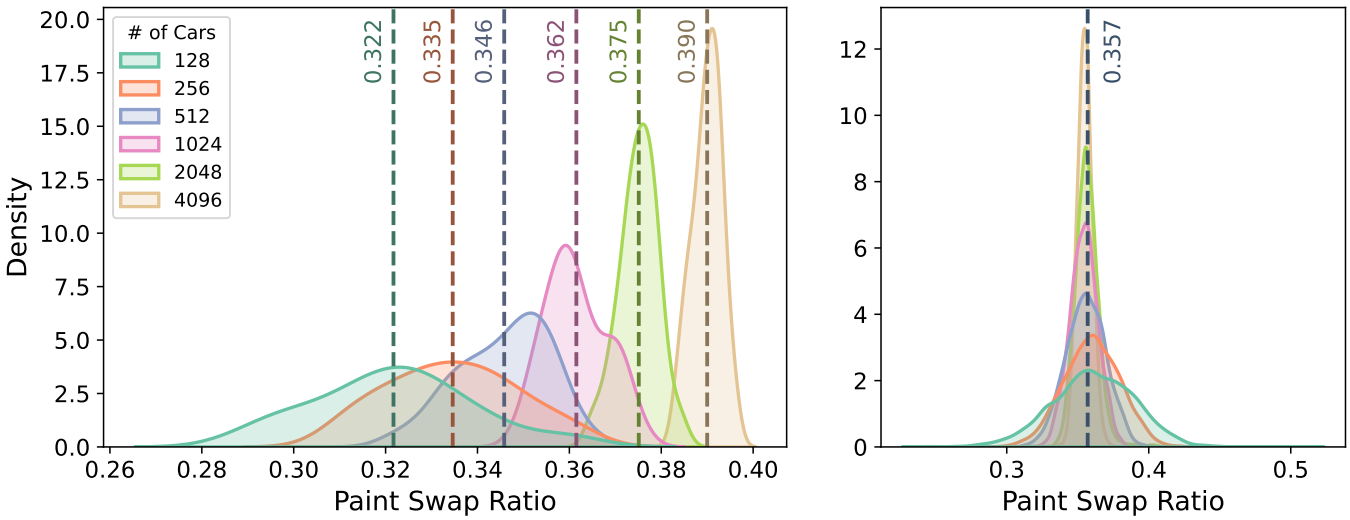
(d) Swap Ratios for RSG vs XQAOA

(e) Swap Ratios for RQAOA vs XQAOA

FIG. 2: **Benchmarking Classical and Quantum Heuristics on BPSP Instances.** (a) Line plot showing the number of paint swaps produced by different heuristics on 200 random problems for each $n \in \{5, 10, \ldots, 120\}$. Each point represents the average over all instances, with XQAOA_1 evaluated from the best of 100 random restarts per problem. Among the classical heuristics, Red-First performs the worst, while RSG performs the best. Both quantum heuristics achieve lower swap counts, with XQAOA_1 consistently outperforming RQAOA_1. (b) Boxplot showing the distribution of paint swap ratios for larger instances comprising 50 random problems for each $n \in \{128, 256, 512, 1024, 2048, 4096\}$. For XQAOA_1, the boxplots include all solutions from 100 restarts per instance. RSG achieves the best classical performance with an average ratio of about 0.37, yet it is consistently surpassed by XQAOA_1, which attains an average ratio of roughly 0.357. RQAOA_1 performs well for smaller n, but its advantage diminishes after $n = 512$. The best solutions from XQAOA_1, visible as outliers below the whiskers, consistently outperform those of RQAOA_1. (c–e) Scatterplots comparing the relative performance of RSG, RQAOA_1, and XQAOA_1. Each plot places one algorithm on each axis, with the $y = x$ diagonal as reference: points above the diagonal indicate superior performance of the x-axis algorithm, while points below indicate the y-axis algorithm performs better. Panel (c) shows RQAOA_1 outperforming RSG on most instances up to $n = 1024$, after which RSG dominates, particularly for $n \geq 2048$. Panel (d) shows XQAOA_1 consistently outperforming RSG by a wide margin across all sizes. Panel (e) shows XQAOA_1 outperforming RQAOA_1 in nearly all cases, with only a few exceptions.

across all tested sizes, strongly supporting convergence to this value in the large-n limit.

Returning to fig. 2, the boxplots in fig. 2b compress as variance shrinks with increasing size, making it difficult to distinguish the relative performance of Recursive-Star Greedy, RQAOA_1, and XQAOA_1. To address this, figs. 2c to 2e present pairwise scatter plots, where each point compares the swap ratios of two algorithms and the diagonal $y = x$ line is shown for reference. Points lying above the diagonal indicate that the algorithm on the x-axis achieves a lower swap ratio, while points below the diagonal indicate that the y-axis algorithm performs better. In fig. 2c, RQAOA_1 outperforms Recursive-Star Greedy up to $n = 1024$, but its advantage disappears at



(a) Swap Ratios for RQAOA

(b) Swap Ratios for XQAOA

FIG. 3: **Distribution of Paint Swap Ratios for RQAOA$_1$ and XQAOA$_1$.** This figure shows KDE plots of paint-swap ratio distributions for larger instances comprising 50 random problems for each $n \in \{128, 256, 512, 1024, 2048, 4096\}$. Panel (a) presents RQAOA$_1$ and panel (b) XQAOA$_1$, with the latter including all solutions from 100 restarts per instance. For RQAOA$_1$, the mean of each distribution shifts steadily right as n increases, indicating deteriorating performance: rising from 0.322 at $n = 128$ to 0.362 at $n = 1024$, where it already falls behind XQAOA$_1$'s average of 0.357. At $n = 2048$, the mean increases further to 0.375, worse than RSG's 0.37, and by $n = 4096$ it reaches 0.39, with the right tail extending to 0.4, matching the performance of RG. In contrast, XQAOA$_1$ exhibits stable behaviour across all sizes, with means consistently near 0.357. As n grows, its distributions sharpen and concentrate around this value, strongly suggesting that XQAOA$_1$ outputs solutions with an average swap ratio of about 0.357 in the large limit.

$n = 2048$ and reverses completely by $n = 4096$, where Recursive-Star Greedy dominates all instances. By contrast, fig. 2d shows that XQAOA$_1$ consistently surpasses Recursive-Star Greedy at every size, typically by a substantial margin. Finally, fig. 2e reveals that XQAOA$_1$ also dominates RQAOA$_1$ across nearly all instances, with only a handful of small-n cases lying close to the diagonal. Taken together, these comparisons reinforce the trend that XQAOA$_1$ maintains robust performance at scale, while RQAOA$_1$ progressively deteriorates and is ultimately overtaken by classical heuristics.

Across all simulations, Recursive-Star Greedy averages about 0.37, above its conjectured 0.361; RQAOA$_1$ shows diminishing returns with size, drifting toward Recursive Greedy despite near-optimal QAOA$_1$ parameters; and XQAOA$_1$ remains stable around 0.357, consistently outperforming both classical heuristics and RQAOA$_1$. Together, these results indicate that XQAOA$_1$ sustains competitiveness at scale, whereas RQAOA$_1$ does not.

V. DISCUSSIONS

In this section we discuss the key patterns that emerge from our benchmarks. We highlight the diminishing returns observed for RQAOA as problem size increases, contrast this with the stable performance of XQAOA around a paint-swap ratio of roughly 0.357, and consider what these findings imply for scalability and the choice of algorithms.

A. Diminishing Returns of Level-1 RQAOA

Our large-scale benchmarks reveal a surprising and previously unreported trend: the performance of RQAOA$_1$ deteriorates with problem size even when the QAOA$_1$ subroutine is run with provably near-optimal parameters [82]. This stands in sharp contrast to prior guarantees showing that RQAOA$_1$ can exactly solve unweighted cycles [30], complete graphs [75], and weighted bipartite graphs [76], as well as large-scale simulations on MaxCut and Ising models without external fields, which reported stable behaviour up to hundreds of qubits with no indication of degradation [30, 82]. Empirically, however, our kernel density estimates in fig. 3a shift monotonically toward higher paint-swap ratios as n grows, providing the first clear evidence of RQAOA$_1$ deteriorating at scale. Related work [82] also noted instability on Ising models with external fields, where RQAOA$_1$ could perform extremely well on some instances but poorly on others.

This degradation could be detected largely because of our choice of metric. The paint-swap ratio $\mathbb{E}[\Delta_C/n]$ is scale-normalised and requires no oracle access to the optimum, enabling consistent evaluation at arbitrarily large n. In contrast, the approximation ratio for MaxCut requires the exact optimum, which is infeasible at scale and can mask size-dependent pathologies. While one might object that these are distinct metrics, our formal reduction of BPSP to MaxCut, together with the expression of BPSP cost in terms of MaxCut cost, suggests that degradation in BPSP implies analogous deterioration in MaxCut, and consequently in general Ising models as system size increases.

B. Performance of Level-1 XQAOA as $n \to \infty$

In contrast to the deteriorating performance of RQAOA_1, XQAOA_1 remains stable and consistent across all tested sizes. As shown in figs. 2b and 3b, its distributions are sharply peaked and essentially stationary, with empirical means concentrated near ≈ 0.357 throughout the full range of n. This behaviour parallels that of the Red-First, Greedy, and Recursive Greedy heuristics, whose distributions are similarly stationary around their respective proven constants. Motivated by this consistency, we state the following conjecture on the asymptotic performance of XQAOA_1.

Conjecture 1. *For Binary Paint Shop Problem instances encoded as Ising models, the level-1 XQAOA ansatz with parameters optimised by gradient-based methods achieves a limiting paint-swap ratio of approximately* 0.357*, i.e.*

$$\lim_{n\to\infty} \mathbb{E}\left[\frac{\Delta_C}{n}\right] \approx 0.357.$$

Although a formal proof remains out of reach—owing to the high dimensionality of the parameter space and the non convexity of the optimisation landscape—the conjectured value is strongly supported by numerical evidence and, being lower than the empirical 0.37 ratio of Recursive-Star Greedy, constitutes a meaningful improvement in the BPSP literature.

C. Beyond the Benchmarked Algorithms

In this work we restricted our benchmarks to classical heuristics with known guarantees for BPSP and shallow-depth QAOA variants previously shown to outperform the best classical algorithms on MaxCut. Other algorithms, while lacking proven performance for BPSP, could in principle achieve lower empirical paint-swap ratios. A natural candidate is the Goemans–Williamson (GW) algorithm [71]: since BPSP reduces to a weighted MaxCut instance, GW can be applied directly. However, two caveats arise. First, the presence of negative edge weights prevents the constant-factor approximation guarantee from carrying over. Second, GW requires solving a large semidefinite program whose memory demands grow rapidly with system size, often exceeding the capacity of standard hardware. For this reason we did not include GW in our large-scale comparisons; preliminary tests on small instances suggest that it can perform marginally better than XQAOA_1, though its scalability is limited.

Beyond this, there exist many untested heuristics that could achieve lower empirical ratios on specific instances, including simple meta-heuristics such as randomised local search, albeit without provable guarantees. Finally, we emphasise that there is still considerable room between the conjectured asymptotic performance of XQAOA_1, $\mathbb{E}_{\mathrm{XQAOA}}[\Delta_C] \approx 0.357n$, and the recently established lower bound on the optimal solution, $\mathbb{E}_{\mathrm{OPT}}[\Delta_C] \geq 0.214n - o(n)$ [50]. This gap suggests that further algorithmic improvements—classical, quantum, or hybrid—remain possible.

VI. CONCLUSION

We studied the Binary Paint Shop Problem through a formal reduction to a weighted MaxCut instance and then to an Ising model, enabled by an Initial Car Colour encoding that halves the number of decision variables and yields exact interaction strengths together with structural graph properties. These reductions provide a rigorous foundation for applying QAOA-type quantum heuristics to BPSP.

Building on this foundation, we benchmarked two state-of-the-art QAOA variants, XQAOA_1 and RQAOA_1, against classical heuristics with proven or conjectured performance guarantees, including Red-First, Greedy, Recursive-Greedy, and Recursive-Star Greedy. Our large-scale simulations reveal three key findings. First, XQAOA_1 consistently outperforms all classical heuristics and RQAOA_1, with outcomes tightly concentrated around a stable paint-swap ratio of ≈ 0.357. Second, RQAOA_1 exhibits clear diminishing returns: its average ratio drifts steadily upward from 0.322 at $n = 128$ to 0.390 at $n = 4096$, eventually being surpassed by Recursive-Star Greedy and approaching Recursive-Greedy. Third, our benchmarks revise the empirical performance of Recursive-Star Greedy on upward to ≈ 0.370, above its conjectured value of 0.361.

The remarkable stationarity of XQAOA_1's distributions motivates the conjecture that it converges to a limiting swap ratio of ≈ 0.357 under our Ising encoding with gradient-based parameter optimisation. Taken together, these results establish XQAOA_1 as the most robust method we tested at scale, while providing the first clear numerical evidence that RQAOA_1 deteriorates on large BPSP instances. More broadly, these findings hint that expressiveness may be a more influential factor than non-locality in shaping the performance of QAOA-type heuristics for general Ising models, though further study

is required to confirm this. Extending this line of work to higher-depth variants, multi-colour paint shop problems, and other industrially motivated combinatorial tasks remains an important direction for future research.

ACKNOWLEDGEMENTS

This research is supported by the Agency for Science, Technology and Research (A*STAR) under the Central Research Fund (CRF) Award for Use-Inspired Basic Research (UIBR), A*STAR C230917003, and the Quantum Innovation Centre (Q.InC) Strategic Research and Translational Thrust.

CODE AND DATA AVAILABILITY

All code used to generate the numerical results, together with the datasets (including the random BPSP instances used in our benchmarks), are publicly available at `github.com/vijeycreative/XQAOA-BPSP`.

[1] J. Preskill, Quantum Computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[2] J. Preskill, Quantum Computing 40 Years Later, in *Feynman Lectures on Computation* (CRC Press, Boca Raton, FL, USA, 2023) pp. 193–244.

[3] B. Cheng, X.-H. Deng, X. Gu, Y. He, G. Hu, P. Huang, J. Li, B.-C. Lin, D. Lu, Y. Lu, C. Qiu, H. Wang, T. Xin, S. Yu, M.-H. Yung, J. Zeng, S. Zhang, Y. Zhong, X. Peng, F. Nori, and D. Yu, Noisy intermediate-scale quantum computers, Frontiers of Physics **18**, 21308 (2023).

[4] H.-Y. Huang, S. Choi, J. R. McClean, and J. Preskill, The vast world of quantum advantage, arXiv preprint arXiv:2508.05720 (2025).

[5] M. E. Beverland, P. Murali, M. Troyer, K. M. Svore, T. Hoeffler, V. Kliuchnikov, G. H. Low, M. Soeken, A. Sundaram, and A. Vaschillo, Assessing requirements to scale to practical quantum advantage, arXiv preprint arXiv:2211.07629 (2022).

[6] J. W. Z. Lau, K. H. Lim, H. Shrotriya, and L. C. Kwek, NISQ computing: where are we and where do we go?, AAPPS Bulletin **32**, 27 (2022).

[7] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, Variational quantum algorithms, Nature Reviews Physics **3**, 625 (2021).

[8] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, *et al.*, Noisy intermediate-scale quantum algorithms, Reviews of Modern Physics **94**, 015004 (2022).

[9] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv preprint arXiv:1411.4028 (2014).

[10] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, A variational eigenvalue solver on a photonic quantum processor, Nature communications **5**, 4213 (2014).

[11] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem, arXiv preprint arXiv:1412.6062 (2014).

[12] P. Vikstål, M. Grönkvist, M. Svensson, M. Andersson, G. Johansson, and G. Ferrini, Applying the quantum approximate optimization algorithm to the tail-assignment problem, Physical Review Applied **14**, 034009 (2020).

[13] M. P. Harrigan, K. J. Sung, M. Neeley, K. J. Satzinger, F. Arute, K. Arya, J. Atalaya, J. C. Bardin, R. Barends, S. Boixo, *et al.*, Quantum approximate optimization of non-planar graph problems on a planar superconducting processor, Nature Physics **17**, 332 (2021).

[14] E. Farhi, J. Goldstone, S. Gutmann, and L. Zhou, The Quantum Approximate Optimization Algorithm and the Sherrington-Kirkpatrick Model at Infinite Size, Quantum **6**, 759 (2022).

[15] E. Anschuetz, J. Olson, A. Aspuru-Guzik, and Y. Cao, Variational quantum factoring, in *Quantum Technology and Optimization Problems: First International Workshop, QTOP 2019, Munich, Germany, March 18, 2019, Proceedings 1* (Springer, 2019) pp. 74–85.

[16] S. Harwood, C. Gambella, D. Trenev, A. Simonetto, D. Bernal, and D. Greenberg, Formulating and solving routing problems on quantum computers, IEEE Transactions on Quantum Engineering **2**, 3100118 (2021).

[17] D. Amaro, M. Rosenkranz, N. Fitzpatrick, K. Hirano, and M. Fiorentini, A case study of variational quantum algorithms for a job shop scheduling problem, EPJ Quantum Technology **9**, 5 (2022).

[18] U. Azad, B. K. Behera, E. A. Ahmed, P. K. Panigrahi, and A. Farouk, Solving vehicle routing problem using quantum approximate optimization algorithm, IEEE Transactions on Intelligent Transportation Systems **24**, 7564 (2023).

[19] D. Fitzek, *Applying quantum approximate optimization to the heterogeneous vehicle routing problem*, Ph.D. thesis, Chalmers Tekniska Hogskola (Sweden) (2022).

[20] D. Fitzek, T. Ghandriz, L. Laine, M. Granath, and A. F. Kockum, Applying quantum approximate optimization to the heterogeneous vehicle routing problem, Scientific Reports **14**, 25415 (2024).

[21] Y. Ruan, Z. Yuan, X. Xue, and Z. Liu, Quantum approximate optimization for combinatorial problems with constraints, Information Sciences **619**, 98 (2023).

[22] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices, Physical Review X **10**, 021067 (2020).

[23] P. C. Lotshaw, T. S. Humble, R. Herrman, J. Ostrowski, and G. Siopsis, Empirical performance bounds for quantum approximate optimization, Quantum Information Processing **20**, 403 (2021).

[24] J. Golden, A. Bärtschi, D. O'Malley, and S. Eidenbenz, Numerical evidence for exponential speed-up of qaoa over unstructured search for approximate constrained optimization, in *2023 IEEE International Conference on*

Quantum Computing and Engineering (QCE), Vol. 01 (2023) pp. 496–505.

[25] L. Bittel and M. Kliesch, Training variational quantum algorithms is NP-hard, Phys. Rev. Lett. **127**, 120502 (2021).

[26] S. Hadfield, T. Hogg, and E. G. Rieffel, Analytical framework for quantum alternating operator ansätze, Quantum Science and Technology **8**, 015017 (2022).

[27] J. Wurtz and P. Love, MaxCut quantum approximate optimization algorithm performance guarantees for $p > 1$, Phys. Rev. A **103**, 042612 (2021).

[28] T. Y. Ng, J. M. Koh, and D. E. Koh, Analytical expressions for the quantum approximate optimization algorithm and its variants, arXiv preprint arXiv:2411.09745 (2024).

[29] N. Benchasattabuse, A. Bärtschi, L. P. García-Pintos, J. Golden, N. Lemons, and S. Eidenbenz, Lower bounds on the number of rounds of the quantum approximate optimization algorithm required for guaranteed approximation ratios, Phys. Rev. A **111**, 062411 (2025).

[30] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang, Obstacles to variational quantum optimization from symmetry protection, Physical Review Letters **125**, 260505 (2020).

[31] Y. Yu, C. Cao, C. Dewey, X.-B. Wang, N. Shannon, and R. Joynt, Quantum approximate optimization algorithm with adaptive bias fields, Phys. Rev. Res. **4**, 023249 (2022).

[32] P. Chandarana, N. N. Hegade, K. Paul, F. Albarrán-Arriagada, E. Solano, A. del Campo, and X. Chen, Digitized-counterdiabatic quantum approximate optimization algorithm, Phys. Rev. Res. **4**, 013141 (2022).

[33] R. Herrman, P. C. Lotshaw, J. Ostrowski, T. S. Humble, and G. Siopsis, Multi-angle quantum approximate optimization algorithm, Scientific Reports **12**, 6781 (2022).

[34] V. Vijendran, A. Das, D. E. Koh, S. M. Assad, and P. K. Lam, An expressive ansatz for low-depth quantum approximate optimisation, Quantum Science and Technology **9**, 025010 (2024).

[35] I. Gaidai and R. Herrman, Performance analysis of multi-angle QAOA for $p > 1$, Scientific Reports **14**, 18911 (2024).

[36] T. Yoshioka, K. Sasada, Y. Nakano, and K. Fujii, Fermionic quantum approximate optimization algorithm, Phys. Rev. Res. **5**, 023071 (2023).

[37] A. Bärtschi and S. Eidenbenz, Grover mixers for QAOA: Shifting complexity from mixer design to state preparation, in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, 2020) pp. 72–82.

[38] Y. Zhu, Z. Zhang, B. Sundar, A. M. Green, C. H. Alderete, N. H. Nguyen, K. R. Hazzard, and N. M. Linke, Multi-round QAOA and advanced mixers on a trapped-ion quantum computer, Quantum Sci. Technol. **8**, 015007 (2022).

[39] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, A review on quantum approximate optimization algorithm and its variants, Physics Reports **1068**, 1 (2024).

[40] S. Lloyd, Quantum approximate optimization is computationally universal, arXiv preprint arXiv:1812.11075 (2018).

[41] M. E. S. Morales, J. D. Biamonte, and Z. Zimborás, On the universality of the quantum approximate optimization algorithm, Quantum Information Processing **19**, 291 (2020).

[42] E. Farhi and A. W. Harrow, Quantum supremacy through the quantum approximate optimization algorithm, arXiv preprint arXiv:1602.07674 (2016).

[43] A. M. Dalzell, A. W. Harrow, D. E. Koh, and R. L. La Placa, How many qubits are needed for quantum computational supremacy?, Quantum **4**, 264 (2020).

[44] T. Epping, W. Hochstattler, and P. Oertel, Some results on a paint shop problem for words, Electronic Notes in Discrete Mathematics **8**, 3 (2001).

[45] T. Epping, W. Hochstättler, and P. Oertel, Complexity results on a paint shop problem, Discrete Applied Mathematics **136**, 217 (2004).

[46] A. Gupta, S. Kale, V. Nagarajan, R. Saket, and B. Schieber, The approximability of the binary paintshop problem, in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings* (Springer, 2013) pp. 205–217.

[47] M. Streif, S. Yarkoni, A. Skolik, F. Neukart, and M. Leib, Beating classical heuristics for the binary paint shop problem with the quantum approximate optimization algorithm, Physical Review A **104**, 012403 (2021).

[48] S. D. Andres and W. Hochstättler, Some heuristics for the binary paint shop problem and their expected number of colour changes, Journal of Discrete Algorithms **9**, 203 (2011).

[49] S. D. Andres, Greedy versus recursive greedy: Uncorrelated heuristics for the binary paint shop problem, Discrete Applied Mathematics **303**, 4 (2021).

[50] J. Hančl, A. Kabela, M. Opler, J. Sosnovec, R. Šámal, and P. Valtr, Improved bounds for the binary paint shop problem, in *Computing and Combinatorics: 29th International Conference, COCOON 2023, Hawaii, HI, USA, December 15–17, 2023, Proceedings, Part II* (Springer-Verlag, Berlin, Heidelberg, 2023) p. 210–221.

[51] C. Xue, Z.-Y. Chen, Y.-C. Wu, and G.-P. Guo, Effects of quantum noise on quantum approximate optimization algorithm, Chinese Physics Letters **38**, 030302 (2021).

[52] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles, Noise-induced barren plateaus in variational quantum algorithms, Nature communications **12**, 6961 (2021).

[53] J. Marshall, F. Wudarski, S. Hadfield, and T. Hogg, Characterizing local noise in QAOA circuits, IOP SciNotes **1**, 025208 (2020).

[54] M. Alam, A. Ash-Saki, and S. Ghosh, Analysis of quantum approximate optimization algorithm under realistic noise in superconducting qubits, arXiv preprint arXiv:1907.09631 (2019).

[55] M. Alam, A. Ash-Saki, and S. Ghosh, Design-space exploration of quantum approximate optimization algorithm under noise, in *2020 IEEE Custom Integrated Circuits Conference (CICC)* (IEEE, 2020) pp. 1–4.

[56] M. Streif, M. Leib, F. Wudarski, E. Rieffel, and Z. Wang, Quantum algorithms with local particle-number conservation: Noise effects and error correction, Physical Review A **103**, 042412 (2021).

[57] E. R. Anschuetz and B. T. Kiani, Quantum variational algorithms are swamped with traps, Nature Communications **13**, 7760 (2022).

[58] D. Stilck França and R. Garcia-Patron, Limitations of optimization algorithms on noisy quantum devices, Nature Physics **17**, 1221 (2021).

[59] A. Weidinger, G. B. Mbeng, and W. Lechner, Error mitigation for quantum approximate optimization, Physical Review A **108**, 032408 (2023).

[60] R. Shaydulin and A. Galda, Error mitigation for deep quantum optimization circuits by leveraging problem symmetries, in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, 2021) pp. 291–300.

[61] L. Botelho, A. Glos, A. Kundu, J. A. Miszczak, Ö. Salehi, and Z. Zimborás, Error mitigation for variational quantum algorithms through mid-circuit measurements, Physical Review A **105**, 022441 (2022).

[62] J. M. Koh, D. E. Koh, and J. Thompson, Readout error mitigation for mid-circuit measurements and feedforward, arXiv preprint arXiv:2406.07611 (2024).

[63] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, Error mitigation extends the computational reach of a noisy quantum processor, Nature **567**, 491 (2019).

[64] B. Koczor, Exponential error suppression for near-term quantum devices, Physical Review X **11**, 031057 (2021).

[65] F. Leymann and J. Barzen, The bitter truth about gate-based quantum algorithms in the NISQ era, Quantum Science and Technology **5**, 044007 (2020).

[66] R. Takagi, S. Endo, S. Minagawa, and M. Gu, Fundamental limits of quantum error mitigation, npj Quantum Information **8**, 114 (2022).

[67] Y. Quek, D. Stilck França, S. Khatri, J. J. Meyer, and J. Eisert, Exponentially tighter bounds on limitations of quantum error mitigation, Nature Physics **20**, 1648 (2024).

[68] G. J. Mooney, J. Villanueva, B. R. Radhan, J. Ghosh, C. D. Hill, and L. C. Hollenberg, An optimization-free recursive QAOA for the binary paint shop problem, arXiv preprint arXiv:2507.10908 (2025).

[69] V. Akshay, D. Rabinovich, E. Campos, and J. Biamonte, Parameter concentrations in quantum approximate optimization, Phys. Rev. A **104**, L010401 (2021).

[70] R. Shaydulin, P. C. Lotshaw, J. Larson, J. Ostrowski, and T. S. Humble, Parameter transfer for quantum approximate optimization of weighted MaxCut, ACM Transactions on Quantum Computing **4(3)**, 19 (2023).

[71] M. X. Goemans and D. P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, Journal of the ACM (JACM) **42**, 1115 (1995).

[72] H. Amini, F. Meunier, H. Michel, and A. Mohajeri, Greedy colorings for the binary paintshop problem, Journal of Discrete Algorithms **8**, 8 (2010).

[73] S. Bravyi, D. Gosset, and R. Movassagh, Classical algorithms for quantum mean values, Nature Physics **17**, 337 (2021).

[74] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang, Hybrid quantum-classical algorithms for approximate graph coloring, Quantum **6**, 678 (2022).

[75] E. Bae and S. Lee, Recursive QAOA outperforms the original QAOA for the MAX-CUT problem on complete graphs, Quantum Information Processing **23**, 78 (2024).

[76] E. Bae, H. Kwon, V. Vijendran, and S. Lee, Modified recursive qaoa for exact max-cut solutions on bipartite graphs: closing the gap beyond qaoa limit, Journal of Physics A: Mathematical and Theoretical **58**, 485304 (2025).

[77] B. Gülbahar, Majority voting with recursive QAOA and cost-restricted uniform sampling for maximum-likelihood detection in massive MIMO, IEEE Transactions on Wireless Communications **24**, 2620 (2025).

[78] L. Palackal, B. Poggel, M. Wulff, H. Ehm, J. M. Lorenz, and C. B. Mendl, Quantum-assisted solution paths for the capacitated vehicle routing problem, in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Vol. 01 (2023) pp. 648–658.

[79] T. Symul, S. M. Assad, and P. K. Lam, Real time demonstration of high bitrate quantum random number generation with coherent laser light, Applied Physics Letters **98**, 231103 (2011).

[80] J. Y. Haw, S. M. Assad, A. M. Lance, N. H. Y. Ng, V. Sharma, P. K. Lam, and T. Symul, Maximization of extractable randomness in a quantum random-number generator, Phys. Rev. Appl. **3**, 054004 (2015).

[81] D. C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, Mathematical programming **45**, 503 (1989).

[82] V. Vijendran, D. E. Koh, E. Bae, H. Kwon, P. K. Lam, and S. M. Assad, Near-optimal parameter tuning of level-1 QAOA for Ising models, arXiv preprint arXiv:2501.16419 (2025).

Appendices

In the following appendices, we delve into the necessary mathematical ingredients needed to formulate BPSP in a framework compatible with implementation using QAOA and its variants. Furthermore, we elucidate the relationship between BPSP and the weighted maximum cut (MaxCut) problem, examining key properties exhibited by the associated MaxCut graph.

Appendix A: BPSP Formulation

We begin by introducing some notation. Throughout the appendix, we fix $n \in \mathbb{Z}^+$ to be a positive integer that denotes the number of distinct symbols. Additionally, we let r and b be formal symbols representing the colours *red* and *blue*, respectively.

BPSP instances: The instances of BPSP are *double-occurrence* words, defined as even-length words over an alphabet $[n] = \{1, \ldots, n\}$ where each symbol $i \in [n]$ occurs exactly twice in each word. Each BPSP word of length $2n$ can be thought of as representing a sequence comprising n cars, with each car appearing precisely twice within the sequence. Formally, we denote the set of such words of length $2n$ by

$$\Gamma_n = \left\{(x_1, x_2, \ldots, x_{2n}) \in [n]^{2n} : \forall i \in [n], |\{j \in [2n] : x_j = i\}| = 2\right\}. \tag{A1}$$

Note that the length of any word $x \in \Gamma_n$ is $|x| = 2n$. The set of double-occurrence words, regardless of their length, is then the union of the sets in (A1):

$$\Gamma = \bigcup_{n \in \mathbb{Z}^+} \Gamma_n. \tag{A2}$$

An example of an $n = 3$ BPSP instance is

$$t = (1, 2, 1, 3, 3, 2) \in \Gamma_3. \tag{A3}$$

BPSP colourings: A valid BPSP colouring of a BPSP instance $x \in \Gamma$ is one that colours the symbols of x using two different colours in such a way that pairs of identical symbols are coloured differently. We let Ξ denote the function that maps an instance $x \in \Gamma$ to the set of its valid colourings, i.e.,

$$\Xi(x) = \left\{f \in \{r, b\}^{|x|} : \forall i \neq j \in [|x|], x_i = x_j \implies f_i \neq f_j\right\}. \tag{A4}$$

We denote the restriction of Ξ to the domain Γ_n by $\Xi_n = \Xi|_{\Gamma_n}$, i.e. if $x \in \Gamma_n$, we write $\Xi(x) = \Xi_n(x)$. For example,

$$rrbrbb, rbbbrr \in \Xi_3(t) \tag{A5}$$

are valid BPSP colourings of the BPSP instance in (A3).

BPSP cost function: The BPSP cost function ξ measures the number of colour swaps in a given colouring of a BPSP instance. More formally, let $f \in \Xi(x)$ be a valid BPSP colouring of a BPSP instance $x \in \Gamma_n$. The BPSP cost ξ of f is given by

$$\xi(f) = \sum_{i=1}^{2n-1} [f_i \neq f_{i+1}], \tag{A6}$$

where $[\,\cdot\,]$ denotes the Iverson bracket which evaluates to 1 if its argument is true and 0 otherwise. When restricted to inputs of length $2n$, we write $\xi = \xi_n$, i.e. $\xi(f) = \xi_n(f)$ if $f \in \{r, b\}^{2n}$. For the valid BPSP colourings in our example (A5), we have

$$\xi(rrbrbb) = 3, \quad \xi(rbbbrr) = 2. \tag{A7}$$

BPSP problem: With the above definitions, the BPSP problem (1) can formally be expressed as follows:

Problem 1. Binary Paint Shop Problem (BPSP)

Input: a BPSP instance $x \in \Gamma$.
Output: minimise $\xi(f)$, as defined in eq. (A6)
subject to $f \in \Xi(x)$, where $\Xi(x)$ is defined in eq. (A4).

For $x \in \Gamma$, we denote the optimal cost and the set of optimal solutions of Problem 1, respectively, as follows:

$$\mathrm{BPSP}(x) = \min_{f \in \Xi(x)} \xi(f), \tag{A8}$$

$$\mathrm{BPSP}^*(x) = \arg\min_{f \in \Xi(x)} \xi(f). \tag{A9}$$

In the context of our example presented in eq. (A3), with some effort, one could demonstrate that $\mathrm{BPSP}(t) = 2$ and $\mathrm{BPSP}^*(t) \ni rbbbrr$.

Appendix B: Fundamentals

1. Eta Function

We begin this section by introducing the *eta function*, a useful function that recurs throughout many subsequent expressions. The *eta function* $\eta : \Gamma_n \times [2n-1] \to \{0,1\}$ is defined as follows:

$$\eta(x,i) = [x_{i+1} \in \{x_1, \ldots, x_i\}] \oplus [x_i \in \{x_1, \ldots, x_{i-1}\}], \tag{B1}$$

where $\oplus$ is the *exclusive or* operation defined by $a \oplus b = 0$ if $(a,b) = (0,0)$ or $(1,1)$, and $a \oplus b = 1$ otherwise. Equivalently,

$$\eta(x,i) = \begin{cases} 0, & \text{if } (x_{i+1} \in \{x_1, \ldots, x_i\} \wedge x_i \in \{x_1, \ldots, x_{i-1}\}) \text{ or } (x_{i+1} \notin \{x_1, \ldots, x_i\} \wedge x_i \notin \{x_1, \ldots, x_{i-1}\}) \\ 1, & \text{if } (x_{i+1} \in \{x_1, \ldots, x_i\} \wedge x_i \notin \{x_1, \ldots, x_{i-1}\}) \text{ or } (x_{i+1} \notin \{x_1, \ldots, x_i\} \wedge x_i \in \{x_1, \ldots, x_{i-1}\}). \end{cases} \tag{B2}$$

For our example word $t = (1,2,1,3,3,2)$ in eq. (A3), we have

$$\eta(t,1) = \eta(t,5) = 0 \text{ and } \eta(t,2) = \eta(t,3) = \eta(t,4) = 1. \tag{B3}$$

We next establish a useful property of the eta function: if a BPSP instance denoted by $x \in \Gamma$ exhibits consecutive occurrences of the same symbol, with the initial occurrence positioned at index i, it follows that $\eta(x,i) = 1$.

Lemma 2. *Let $x \in \Gamma_n$ and $i \in [2n-1]$. If $x_i = x_{i+1}$, then $\eta(x,i) = 1$.*

Proof. If $x \in \Gamma_n$ and $x_i = x_{i+1}$, then $x_{i+1} \in \{x_1, \ldots, x_i\}$ and $x_i \notin \{x_1, \ldots, x_{i-1}\}$ since the symbol x_i occurs at most twice. Hence, $\eta(x,i) = 1$. □

2. Colouring Schemes

A *colouring scheme* refers to an algorithm designed to assign colours to a BPSP word in accordance with a valid BPSP colouring. These schemes can be categorised as either deterministic or probabilistic. In this section, we focus on deterministic colouring schemes, which can be formally characterised by functions $\sigma : \Gamma_n \to \{r,b\}^{2n}$ satisfying $\sigma(x) \in \Xi_n(x)$. To exemplify our definition, we discuss two deterministic colouring schemes:

1. The *red-first colouring scheme* is the function $\sigma_{\mathrm{RF}} : \Gamma_n \to \{r,b\}^{2n}$, where $\forall x \in \Gamma_n$ and $i \in [2n]$,

$$\sigma_{\mathrm{RF}}(x)_i = \begin{cases} r, & x_i \notin \{x_1, \ldots, x_{i-1}\} \\ b, & \text{otherwise.} \end{cases} \tag{B4}$$

   In words, this scheme colours the initial instance of each symbol red and the subsequent occurrence of each symbol blue.

2. The *greedy colouring scheme* is the function $\sigma_{\mathrm{G}} : \Gamma_n \to \{r,b\}^{2n}$, where $\forall x \in \Gamma_n$, $g = \sigma_{\mathrm{G}}(x) \in \{r,b\}^{2n}$ is defined recursively by

$$g_1 = r \tag{B5}$$

$$g_i = \begin{cases} g_{i-1}, & x_i \notin \{x_1, \ldots, x_{i-1}\}, \\ \neg g_{\min\{j : x_j = x_i\}}, & \text{otherwise,} \end{cases} \tag{B6}$$

   for $i \geq 2$. In essence, this scheme employs a greedy approach to colour symbols in x from left to right, switching colours only when maintaining the current colour would violate the validity of the BPSP colouring.

3. The *recursive greedy colouring scheme* is a map $\sigma_{\mathrm{RG}} : \Gamma_n \to \{r, b\}^{2n}$ that assigns to every BPSP instance x a feasible colouring $\sigma_{\mathrm{RG}}(x) \in \Xi_n(x)$ of its $2n$ car occurrences. Write the input as $x^{(n)} = (x_1^{(n)}, \ldots, x_{2n}^{(n)})$, where the superscript records the number of distinct cars. The construction proceeds in two stages. *Reduction* iteratively removes, from right to left, the last car in the current sequence together with its earlier partner, recording the partner's index and thereby producing a chain of shorter instances $x^{(n)} \to x^{(n-1)} \to \cdots \to x^{(1)}$. *Reconstruction* then rebuilds the colouring from length 2 up to $2n$: at each step it copies all already fixed colours into the longer sequence and assigns colours to the two reinserted cars using only local information. We now formalise the two stages, beginning with the reduction step.

   Reduction: Starting from $x^{(n)}$, peel off pairs from right to left; at each step note where the partner of the last car sits so the pair can be reinserted in the correct gap later. Formally, for $j = n-1, n-2, \ldots, 1$:

   (a) Let $m_{j+1} \in \{1, 2, \ldots, 2j+1\}$ be the position of the earlier occurrence of the last car of $x^{(j+1)}$, i.e. $x_{m_{j+1}}^{(j+1)} = x_{2j+2}^{(j+1)}$.

   (b) Delete both occurrences to obtain $x^{(j)} \in \Gamma_j$:

$$x_k^{(j)} = \begin{cases} x_k^{(j+1)}, & k = 1, 2, \ldots, m_{j+1} - 1, \\ x_{k+1}^{(j+1)}, & k = m_{j+1}, m_{j+1} + 1, \ldots, 2j. \end{cases}$$

   Reconstruction: Begin from the unique length-two colouring and, at each extension, reinsert the pair at the recorded gap, copy all previously fixed colours unchanged, and choose the two new colours so as not to introduce avoidable swaps. Concretely:

   (a) *Base Case.* Set $g^{(1)} = (r, b) \in \Xi_1(x^{(1)})$.

   (b) *Recursive Step.* For $j = 2, 3, \ldots, n$:

      i. *Embed the previously assigned colours* by copying the colouring from the shorter instance into $x^{(j)}$ at all positions except the two reinserted occurrences, which occupy indices m_j and $2j$:

$$g_k^{(j)} = \begin{cases} g_k^{(j-1)}, & k = 1, 2, \ldots, m_j - 1, \\ g_{k-1}^{(j-1)}, & k = m_j + 1, m_j + 2, \ldots, 2j - 1. \end{cases}$$

      ii. *Colour the two reinserted cars*—the first occurrence placed at index m_j and the second occurrence placed at index $2j$. Let $L := g_{m_j - 1}^{(j)}$ and $R := g_{m_j + 1}^{(j)}$ denote the colours of the cars immediately to the left and right of the gap (when they exist), and let $T := g_{2j-1}^{(j)}$ be the colour of the last pre-existing car. Set

$$\begin{aligned} &\text{if } m_j = 1: && g_{m_j}^{(j)} = R, \quad g_{2j}^{(j)} = \neg R; \\ &\text{if } m_j = 2j - 1: && g_{m_j}^{(j)} = L, \quad g_{2j}^{(j)} = \neg L; \\ &\text{if } 1 < m_j < 2j - 1 \text{ and } L = R: && g_{m_j}^{(j)} = L, \quad g_{2j}^{(j)} = \neg L; \\ &\text{if } 1 < m_j < 2j - 1 \text{ and } L \neq R: && g_{2j}^{(j)} = T, \quad g_{m_j}^{(j)} = \neg T. \end{aligned}$$

   This recursion yields a unique colouring $g^{(n)}$, ensuring that the two occurrences of each car receive opposite colours (so $g^{(n)} \in \Xi_n(x^{(n)})$), and does not introduce additional colour changes beyond those that are unavoidable. We set $\sigma_{\mathrm{RG}}(x^{(n)}) := g^{(n)}$.

4. The *recursive star greedy* colouring is a map $\sigma_{\mathrm{RSG}} : \Gamma_n \to \{r, b\}^{2n}$ that returns, for every BPSP instance x, a feasible colouring $\sigma_{\mathrm{RSG}}(x) \in \Xi_n(x)$. It augments the recursive greedy scheme by allowing a temporary symbol $*$ ("star") during the construction to mark a car whose colour choice does not affect the current number of paint swaps. Stars are used only as placeholders: a finalisation step replaces each $*$ by r or b without changing the swap count and while preserving the opposite-colour constraint for the two occurrences of every car. Write the input as $x^{(n)} = (x_1^{(n)}, \ldots, x_{2n}^{(n)})$. The algorithm has two stages. *Reduction* removes, from left to right, the first car of the current sequence together with its later partner, records the partner's index, and thereby produces a chain of shorter instances $x^{(n)} \to x^{(n-1)} \to \cdots \to x^{(1)}$. *Reconstruction* then rebuilds the colouring from length 2 up to $2n$ by copying previously fixed colours and assigning colours—or, when locally indifferent, stars—to the

previously reinserted cars using only local information. We now formalise these two stages, beginning with the reduction step.

Reduction: Starting from $x^{(n)}$, peel off pairs from left to right; at each step note where the partner of the first car sits so the pair can be reinserted in the correct gap later. Formally, for $j = n-1, n-2, \ldots, 1$:

(a) Let $m_j \in \{2, 3, \ldots, 2j+2\}$ be the position of the partner of the first car of $x^{(j+1)}$, i.e. $x_1^{(j+1)} = x_{m_j}^{(j+1)}$.

(b) Delete positions 1 and m_j to obtain $x^{(j)} \in \Gamma_j$:

$$x_k^{(j)} = \begin{cases} x_{k+1}^{(j+1)}, & k = 1, 2, \ldots, m_j - 2, \\ x_{k+2}^{(j+1)}, & k = m_j - 1, m_j, \ldots, 2j, \end{cases}$$

and record m_j (the future insertion position of the second occurrence).

Reconstruction: Start from the unique length-two colouring; at each extension, reinsert the recorded pair, copy all previously fixed colours unchanged, and assign colours to the two new cars so as to avoid any unnecessary swaps. When a local tie occurs—i.e., both assignments give the same swap count—mark the indifferent choice with a star $*$. After the final step, replace every remaining $*$ arbitrarily by r or b, choosing the replacements so that the two occurrences of each car have opposite colours and the swap count is unchanged. Concretely:

(a) *Base Case.* Set $h^{(1)} = (r, b) \in \Xi_1(x^{(1)})$.

(b) *Recursive Step.* For $j = 2, 3, \ldots, n$:

i. *Embed the previously assigned symbols* by copying the colouring from the shorter instance into $x^{(j)}$ at all positions except the two reinserted occurrences, which occupy indices 1 and m_{j-1}:

$$h_k^{(j)} = \begin{cases} h_{k-1}^{(j-1)}, & k = 2, 3, \ldots, m_{j-1} - 1, \\ h_{k-2}^{(j-1)}, & k = m_{j-1} + 1, \ldots, 2j. \end{cases}$$

ii. *Colour the two reinserted cars*—the first occurrence at index 1 and the second at index m_{j-1}. Let $L := h_{m_{j-1}-1}^{(j)}$ and $R := h_{m_{j-1}+1}^{(j)}$ be the colours immediately to the left and right of the car at m_{j-1} (when they exist), and let $N := h_2^{(j)}$ be the colour of the second car. If exactly one of L or R is a star, write $s \in \{-1, +1\}$ for the side of that star (so $h_{m_{j-1}+s}^{(j)} = *$) and set $U := h_{m_{j-1}-s}^{(j)}$ for its second (non-starred) neighbour. Apply the following local rules:

$$\begin{array}{lll}
\text{if } m_{j-1} = 2: & h_{m_{j-1}}^{(j)} = R, & h_1^{(j)} = \neg R; \\
\text{if } m_{j-1} = 2j: & h_{m_{j-1}}^{(j)} = b, & h_1^{(j)} = r; \\
\text{if } 2 < m_{j-1} < 2j \text{ and } L = R: & h_{m_{j-1}}^{(j)} = L, & h_1^{(j)} = \neg L; \\
\text{if } 2 < m_{j-1} < 2j \text{ and } L \neq R: & h_1^{(j)} = N, & h_{m_{j-1}}^{(j)} = \neg N; \\
\text{if } 2 < m_{j-1} < 2j \text{ and exactly one of } \{L, R\} \text{ is } * \text{ and } U \neq N: & h_1^{(j)} = N, & h_{m_{j-1}}^{(j)} = \neg N; \\
\text{if } 2 < m_{j-1} < 2j \text{ and exactly one of } \{L, R\} \text{ is } * \text{ and } U = N: & h_1^{(j)} = N, & h_{m_{j-1}}^{(j)} = h_{m_{j-1}+s}^{(j)} = \neg N.
\end{array}$$

iii. *Star Creation.* Let b be the index of the second occurrence of the second car in the sequence $x^{(j)}$. Define the adjacent colours as follows:

$$L_1 := h_1^{(j)}, \quad R_1 := h_3^{(j)}, \quad L_2 := h_{b-1}^{(j)}, \quad R_2 := h_{b+1}^{(j)}.$$

Apply the following rules:

$$\begin{array}{ll}
\text{if } L_1 = R_1 \text{ and } L_2 = R_2: & h_2^{(j)} = h_b^{(j)} = *; \\
\text{if } L_1 \neq R_1 \text{ and } L_2 \neq R_2: & h_2^{(j)} = h_b^{(j)} = *.
\end{array}$$

(c) *Breaking Ties.* After all pairs have been reinserted, replace each remaining $*$ arbitrarily with r or b, subject to the opposite-colour constraint: the two occurrences of any car must receive different colours. This choice does not affect the total number of paint swaps.

The recursion together with the star finalisation step determines a colouring $h^{(n)}$. It is feasible, so $h^{(n)} \in \Xi_n(x^{(n)})$, and by construction no additional paint swaps are created unless they are forced by the local neighbourhood. We set $\sigma_{\mathrm{RSG}}(x^{(n)}) := h^{(n)}$. The recursive star greedy method is a local strategy: resolving ties with stars and committing based only on current information can block better future choices. Consequently, there are instances where recursive star greedy performs worse than recursive greedy, and in some cases even worse than the basic greedy heuristic.

Note that in all colouring schemes, we consistently follow the convention of colouring the first symbol of x in red.

3. BPSP Costs of Red-First colouring

In this section, we focus on the red-first colouring scheme and show that the BPSP cost of this colouring can be expressed in terms of the eta function.

Proposition 3. *Let $x \in \Gamma_n$. Then the BPSP cost of the red-first colouring of x is*

$$\xi_n\left(\sigma_{\mathrm{RF}}(x)\right) = \sum_{i=1}^{2n-1} \eta(x,i) = n - \frac{1}{2} - \frac{1}{2}\sum_{i=1}^{2n-1} (-1)^{\eta(x,i)}. \tag{B7}$$

Proof. We start by proving the first equality. Let $f = \sigma_{\mathrm{RF}}(x)$. Then, we have that

$$f_i = \begin{cases} r, & x_i \notin \{x_1, \ldots, x_{i-1}\} \\ b, & x_i \in \{x_1, \ldots, x_{i-1}\} \end{cases} \tag{B8}$$

$$\text{and } f_{i+1} = \begin{cases} r, & x_{i+1} \notin \{x_1, \ldots, x_i\} \\ b, & x_{i+1} \in \{x_1, \ldots, x_i\}. \end{cases} \tag{B9}$$

Hence, it follows that:

$$\begin{aligned} f_i \neq f_{i+1} &\iff (f_i = r \wedge f_{i+1} = b) \vee (f_i = b \wedge f_{i+1} = r) \\ &\iff (x_i \notin \{x_1, \ldots, x_{i-1}\} \wedge x_{i+1} \in \{x_1, \ldots, x_i\}) \vee (x_i \in \{x_1, \ldots, x_{i-1}\} \wedge x_{i+1} \notin \{x_1, \ldots, x_i\}) \\ &\iff [x_{i+1} \in \{x_1, \ldots, x_i\}] \oplus [x_i \in \{x_1, \ldots, x_{i-1}\}] = 1 \\ &\iff \eta(x,i) = 1. \end{aligned} \tag{B10}$$

Therefore, the sum $\xi_n(f)$ can be calculated as:

$$\xi_n(f) = \sum_{i=1}^{2n-1} [f_i \neq f_{i+1}] \tag{B11}$$

$$= \sum_{i=1}^{2n-1} [\eta(x_i, i) = 1] \tag{B12}$$

$$= \sum_{i=1}^{2n-1} \eta(x_i, i), \quad \text{since } \eta(x_i) \in \{0,1\}. \tag{B13}$$

To prove the second equality, we use the fact that if $z \in \{0,1\}$, then $z = \dfrac{1-(-1)^z}{2}$. This allows us to rewrite the above expression as follows:

$$\xi_n(f) = \sum_{i=1}^{2n-1} \eta(x,i) = \sum_{i=1}^{2n-1} \frac{1-(-1)^{\eta(x,i)}}{2} = n - \frac{1}{2} - \frac{1}{2}\sum_{i=1}^{2n-1} (-1)^{\eta(x,i)}.$$

□

Appendix C: Initial-Car-Colour (ICC) Encoding

1. ICC Encoding Scheme

In eq. (A4), we defined a valid BPSP colouring of a BPSP instance $x \in \Gamma_n$ using a $2n$-bit string $f \in \{r,b\}^{2n}$. However, this representation carries excess information. Merely indicating the colouring of the initial occurrence of each symbol in x is adequate for reconstructing the valid colouring. In other words, a succinct description of the colouring requires only n bits. To formally express the relationship between these two representations, we introduce the function $\mathcal{E}_n \colon \Gamma_n \times \{r,b\}^n \to \{r,b\}^{2n}$ defined as:

$$\mathcal{E}_n(x,z)_i = \neg^{[x_i \in \{x_1,\ldots,x_{i-1}\}]} z_{x_i} \tag{C1}$$

$$= \begin{cases} z_{x_i}, & \text{if } x_i \notin \{x_1,\ldots,x_{i-1}\}, \\ \bar{z}_{x_i}, & \text{if } x_i \in \{x_1,\ldots,x_{i-1}\}, \end{cases} \quad \text{for } i \in [2n]. \tag{C2}$$

In words, $\mathcal{E}_n$ takes as input a BPSP instance and the succinct n-bit description of a BPSP colouring and outputs the $2n$-bit description used in (A4). Also, we define $\mathcal{E} = \bigcup_{n\in\mathbb{N}} \mathcal{E}_n$ to be the union of the functions $\mathcal{E}_n$ (viewed as a binary relation), i.e., $\mathcal{E}(x,z) = \mathcal{E}_n(x,z)$ if $x \in \Gamma_n$ and $z \in \{r,b\}^n$. For convenience, we shall refer to the elements $f \in \Xi(x)$ defined in eq. (A4) as BPSP colourings in the *standard encoding* and refer to the succinct colouring $\mathcal{E}(x,z)$ as BPSP colourings in the *ICC encoding*.

Now, note that if we fix the first argument of $\mathcal{E}_n$, we get a function $\{r,b\}^n \to \{r,b\}^{2n}$ that maps the succinct n-bit description to the original $2n$-bit description. Formally, this may be described by the *currying* function, which gives:

$$\mathrm{curry}(\mathcal{E}_n) \colon \Gamma_n \to \left(\{r,b\}^n \to \{r,b\}^{2n}\right), \tag{C3}$$

where

$$\mathrm{curry}(\mathcal{E}_n)(x)(z) = \mathcal{E}_n(x,z). \tag{C4}$$

Note that for $x \in \Gamma_n$, the function $\mathrm{curry}(\mathcal{E}_n)(x) : \{r,b\}^n \to \{r,b\}^{2n}$ is not surjective, since not all $2n$-bit colourings are valid BPSP colourings. For simplicity, we will henceforth restrict the codomain of $\mathrm{curry}(\mathcal{E}_n)(x)$ to its range. Our next proposition establishes this notion and confirms that the range of $\mathrm{curry}(\mathcal{E}_n)(x)$ is $\Xi_n(x)$.

Proposition 4. *Let $x \in \Gamma_n$. Then, the function*

$$\mathrm{curry}(\mathcal{E}_n)(x) \colon \{r,b\}^n \to \Xi_n(x) \tag{C5}$$

is invertible, with the inverse given by

$$\mathrm{curry}(\mathcal{E}_n)(x)^{-1} \colon \Xi_n(x) \to \{r,b\}^n, \tag{C6}$$

where

$$\mathrm{curry}(\mathcal{E}_n)(x)^{-1}(f)_t = f_{\min\{k|x_k=t\}}, \qquad \text{for } t \in [n]. \tag{C7}$$

In other words, for all $x \in \Gamma_n$, we have the following:

(i) For all $z \in \{r,b\}^n$, $\mathcal{E}_n(x,z) = \mathrm{curry}(\mathcal{E}_n)(x)(z) \in \Xi_n(x)$,

(ii) Let $f \in \Xi_n(x)$ and $z = \left(f_{\min\{k|x_k=t\}}\right)_{t=1}^n$. Then

$$\mathcal{E}_n(x,z) := \mathrm{curry}(\mathcal{E}_n)(x)(z) = f, \tag{C8}$$

That is, one could write

$$\Xi_n(x) = \{\mathcal{E}_n(x,z) \mid z \in \{r,b\}^n\}. \tag{C9}$$

Proof. We prove the two statements (i) and (ii) of the proposition separately:

(i) Let $x \in \Gamma_n$ and $z \in \{r,b\}^n$. Choose distinct indices $i < j \in [2n]$ such that $x_i = x_j$. In order to demonstrate that $\mathcal{E}_n(x,z) \in \Xi_n(x)$, we employ a proof by contradiction. Suppose that $\mathcal{E}_n(x,z)_i = \mathcal{E}_n(x,z)_j$. Then, the following implications hold

$$\begin{aligned}
&\neg^{[x_i \in \{x_1,\ldots,x_{i-1}\}]} z_{x_i} = \neg^{[x_j \in \{x_1,\ldots,x_{j-1}\}]} z_{x_j} \\
&\implies \neg^{[t \in \{x_1,\ldots,x_{i-1}\}]} z_t = \neg^{[t \in \{x_1,\ldots,x_{j-1}\}]} z_t \\
&\implies [t \in \{x_1,\ldots,x_{i-1}\}] = [t \in \{x_1,\ldots,x_{j-1}\}] \\
&\implies \underbrace{t \in \{x_1,\ldots,x_{i-1}\} \wedge t \in \{x_1,\ldots,x_{j-1}\}}_{①} \quad \text{or} \quad \underbrace{t \notin \{x_1,\ldots,x_{i-1}\} \wedge t \notin \{x_1,\ldots,x_{j-1}\}}_{②}.
\end{aligned}$$

Now, case ① implies that there exists $k \in \{1,\ldots,i-1\}$ such that $t = x_k$. But this would then mean that $x_k = x_i = x_j = t$ for distinct $k < i < j$, which then implies that $|\{j : x_j = t\}| \geqslant 3$, which contradicts our assumption that $x \in \Gamma_n$. On the other hand, case ② implies that $t \notin \{x_1,\ldots,x_{j-1}\}$. But since $i < j$, we have $\{x_1,\ldots,x_{j-1}\} \ni x_i = t$, which contradicts ②.

Since both cases lead to a contradiction, $\mathcal{E}_n(x,z)_i = \mathcal{E}_n(x,z)_j$, which contradicts our supposition. Therefore, $\mathcal{E}_n(x,z) \in \Xi_n(x)$.

(ii) Let $x \in \Gamma_n$ represent a BPSP instance, and let $f \in \Xi_n(x)$ denote a valid BPSP colouring. For $t \in [n]$, define $z_t = f_{\min\{k : x_k = t\}}$.

We aim to show that $\mathcal{E}_n(x,z) = f$. To establish this, we note that for $i \in [2n]$,

$$\mathcal{E}_n(x,z)_i = \neg^{[x_i \in \{x_1,\ldots,x_{i-1}\}]} z_{x_i} \tag{C10}$$

$$= \neg^{[x_i \in \{x_1,\ldots,x_{i-1}\}]} f_{\min\{k : x_k = x_i\}}. \tag{C11}$$

Since $x \in \Gamma_n$ we have that $|\{k \colon x_k = x_i\}| = 2$. Hence, let l be the unique element in $\{k \colon x_k = x_i\}$ that is not i. In other words, $\{k \colon x_k = x_i\} = \{i, l\}$, where $l \neq i$. We now consider the following two cases.

<u>Case 1: $i < l$</u>. In this case, we have $x_i \notin \{x_1,\ldots,x_{i-1}\}$ (otherwise $\exists k < i \colon x_k = x_i$), and $\min\{k \colon x_k = x_i\} = i$. Hence, by eq. (C11) it follows that

$$\mathcal{E}_n(x,z)_i = \neg^0 f_i = f_i. \tag{C12}$$

<u>Case 2: $i > l$</u>: In this case, $x_i \in \{x_1,\ldots,x_l,\ldots,x_{i-1}\}$ since $l \in \{1,\ldots,i-1\}$. Furthermore, $\min\{k \colon x_k = x_i\} = l$. Hence, by eq. (C11) we have that

$$\mathcal{E}_n(x,z)_i = \neg f_l = f_i. \tag{C13}$$

Since $\mathcal{E}_n(x,z)_i = f_i$ for all $i \in [2n]$, it follows that $\mathcal{E}_n(x,z) = f$.

This completes our proof of the proposition. □

2. BPSP Cost in ICC Encoding

In eq. (A6), we expressed the BPSP cost function ξ as a function of the BPSP colouring $f \in \Xi(x)$. With the introduction of the ICC encoding scheme in appendix C 1, our objective in this section is to express the BPSP cost function directly in terms of the ICC-encoded colouring $\mathcal{E}(x,z)$. To this end, we define the function $\tilde{\xi}_n : \Gamma_n \times \{r,b\}^n \to \mathbb{N}$ as

$$\tilde{\xi}_n = \xi_n \circ \mathcal{E}_n \tag{C14}$$

and $\tilde{\xi} = \bigcup_{n \in \mathbb{N}} \tilde{\xi}_n$. Expressed in words, $\tilde{\xi}_n(x,z)$ denotes the tally of colour swaps when the BPSP instance x is coloured using the colouring specified by z. The proof of the following proposition derives an explicit expression for $\tilde{\xi}_n(x,z)$.

Proposition 5. *Let $z \in \{r,b\}^n$ and $x \in \Gamma_n$. Then,*

$$\tilde{\xi}_n(x,z) = \sum_{i=1}^{2n-1} \left[\neg^{\eta(x,i)} z_{x_i} \neq z_{x_{i+1}} \right]. \tag{C15}$$

Proof. Through a straightforward calculation, we determine:

$$\begin{aligned}
\tilde{\xi}_n(x,z) &= \xi_n(\mathcal{E}_n(x,z)) \\
&= \sum_{i=1}^{2n-1} [\mathcal{E}_n(x,z)_i \neq \mathcal{E}_n(x,z)_{i+1}] \\
&= \sum_{i=1}^{2n-1} \left[\neg^{[x_i \in \{x_1,\ldots,x_{i-1}\}]} z_{x_i} \neq \neg^{[x_{i+1} \in \{x_1,\ldots,x_i\}]} z_{x_{i+1}}\right] \\
&= \sum_{i=1}^{2n-1} \left[\neg^{[x_{i+1} \in \{x_1,\ldots,x_i\}] \oplus [x_i \in \{x_1,\ldots,x_{i-1}\}]} z_{x_i} \neq z_{x_{i+1}}\right] \\
&= \sum_{i=1}^{2n-1} \left[\neg^{\eta(x,i)} z_{x_i} \neq z_{x_{i+1}}\right],
\end{aligned}$$

where the first line is derived from eq. (C14), the second line from eq. (A6), the third line from eq. (C1), and the fifth line from the definition of the eta function as specified in eq. (B1). The fourth line follows from the properties $\neg^a \neg^b = \neg^{a \oplus b}$ and $\neg^0 = I$. This completes the proof of the proposition. □

3. BPSP in ICC Encoding

We now formulate the BPSP problem, as defined in Problem 1, using the ICC encoding. We begin by observing the equivalence $\{\tilde{\xi}_n(x,z) : z \in \{r,b\}^n\} = \{\xi_n(f) : f \in \Xi_n(x)\}$, which implies that for any BPSP instance $x \in \Gamma_n$,

$$\min_{f \in \Xi_n(x)} \xi_n(f) = \min_{z \in \{r,b\}^n} \tilde{\xi}_n(x,z). \tag{C16}$$

Hence, in parallel to the formulation presented in Problem 1, the *ICC formulation* of the BPSP problem can be expressed as

$$\begin{aligned}
\text{Input:}\ & x \in \Gamma \\
\text{Output:}\ & \text{Minimize} \quad \widetilde{\xi}(x,z) \\
& \text{subject to} \quad z \in \{r,b\}^{|x|/2}.
\end{aligned} \tag{C17}$$

Also, eq. (A8) can be written as

$$\mathrm{BPSP}(x) = \min_{f \in \Xi(\omega)} \xi(f) = \min_{z \in \{r,b\}^{|x|/2}} \widetilde{\xi}(x,z). \tag{C18}$$

Similarly to eq. (A9), we define $\widetilde{\mathrm{BPSP}}^*(x)$ as

$$\widetilde{\mathrm{BPSP}}^*(x) := \operatorname*{arg\,min}_{z \in \{r,b\}^{|x|/2}} \tilde{\xi}(x,z) \quad \subseteq \{r,b\}^{|x|/2}. \tag{C19}$$

4. ICC Encoding of Colouring Schemes

Our next objective is to reformulate the results of appendix B 2 for colouring schemes using the ICC encoding. This can be achieved as follows: Let $\sigma : \Gamma_n \to \{r,b\}^{2n}$ be a BPSP colouring scheme. We define the *ICC encoding* of σ as $\mathrm{ICC}(\sigma) : \Gamma_n \to \{r,b\}^n$, where

$$\mathrm{ICC}(\sigma)(x) = (\mathrm{curry}(\mathcal{E}_n)(x))^{-1}(\sigma(x)), \qquad \text{for } x \in \Gamma_n. \tag{C20}$$

Equivalently,

$$\sigma(x) = \mathrm{curry}(\mathcal{E}_n)(x)\,(\mathrm{ICC}(\sigma)(x)) = \mathcal{E}_n\,(x, \mathrm{ICC}(\sigma)(x)). \tag{C21}$$

Therefore,

$$\xi(\sigma(x)) = \widetilde{\xi}(x, \mathrm{ICC}(\sigma)(x)). \tag{C22}$$

Our next proposition demonstrates that the ICC encoding of the red-first colouring scheme maps any length-n BPSP instance to r^n, aligning with the prescribed rule of colouring the initial occurrence of each symbol in red.

Proposition 6. *Let $x \in \Gamma_n$. The ICC encoding of the red-first colouring* $\mathrm{RF}(x)$ *of x is*

$$\mathrm{ICC}\left(\sigma_{\mathrm{RF}}\right)(x) = r^n. \tag{C23}$$

Proof. For $t \in [n]$, we have

$$\begin{aligned} \mathrm{ICC}(\sigma_{\mathrm{RF}})(x)_t &= \mathrm{curry}(\mathcal{E}_n)(x)^{-1}\left(\sigma_{\mathrm{RF}}(x)\right)_t \\ &= \sigma_{\mathrm{RF}}(x)_{\min\{k : x_k = t\}}. \end{aligned} \tag{C24}$$

Now, let $i = \min\{k : x_k = t\}$. Then, $x_j = x_i \Rightarrow j > i$, which implies that

$$\begin{aligned} & x_i \notin \{x_1, \ldots, x_{i-1}\} \\ \implies \quad & \sigma_{\mathrm{RF}}(x_i) = r \\ \implies \quad & \mathrm{ICC}(\sigma_{\mathrm{RF}})(x)_t = r \quad \forall t \in [n] \\ \implies \quad & \mathrm{ICC}\left(\sigma_{\mathrm{RF}}\right)(x) = r^n, \end{aligned}$$

completing the proof of the proposition. □

Appendix D: BPSP Graph

In this appendix, we will demonstrate how every BPSP instance x can be associated with a graph G_x, referred to as the BPSP graph. Through this association, we will establish that computing $\mathrm{BPSP}(x)$ from x can be efficiently reduced to finding the maximum cut of G_x.

1. BPSP Graph: Definition

We begin by establishing some notation. For the BPSP instance $x = (x_1, x_2, \ldots, x_{2n}) \in \Gamma_n$, we define the function $\theta_x : 2^{[n]} \to \mathbb{R}$, which maps subsets of $[n]$ to real numbers as follows:

$$\theta_x(e) = -\sum_{i=1}^{2n-1} (-1)^{\eta(x,i)} \delta_{e,\{x_i, x_{i+1}\}}, \tag{D1}$$

where $2^{[n]}$ denotes the power set of $[n]$ and δ denotes the Kronecker delta. Now, we define the BPSP graph.

Definition 7 (BPSP graph)**.** Let $x = (x_1, x_2, \ldots, x_{2n}) \in \Gamma_n \subseteq [n]^{2n}$ be a BPSP instance. The *BPSP graph* of x is the weighted graph $G_x = (V_x, E_x, W_x)$, where

- $V_x = [n]$ are the vertices,
- $E_x = \left\{ \{x_i, x_{i+1}\} : i \in [2n-1],\ x_i \neq x_{i+1},\ \theta_x\left(\{x_i, x_{i+1}\}\right) \neq 0 \right\}$ are the edges,
- $W_x = \theta_x|_{E_x} : E_x \to \mathbb{R}$ is the weight function assigning edges to real numbers.

In essence, the BPSP graph associated with a length-$2n$ BPSP instance x is constructed with vertices representing the symbols of the alphabet $[n]$. In this graph, edges connect symbols that are adjacent in x, with the exception of consecutive identical symbols and the zeroes of the θ_x function. According to Definition 7 and eq. (D1), the weight of an edge $e \in E_x \subset [n]$ in the graph is given by

$$W_x(e) = -\sum_{i=1}^{2n-1} (-1)^{\eta(x,i)} \delta_{e,\{x_i, x_{i+1}\}}. \tag{D2}$$

2. Properties of BPSP Graph

a. Possible degrees of BPSP graph

We begin by establishing some properties of BPSP graphs. To initiate, we show that the vertex degrees of a BPSP graph cannot exceed 4. Denoting the degree of a vertex v in G by

$$\deg_G(v) = |\{u \in V : \{u, v\} \in E\}|, \tag{D3}$$

we prove the following proposition.

Proposition 8. *Let $x \in \Gamma_n$ be a BPSP instance. Then, for all vertices $v \in V_x$ in the BPSP graph of x,*

$$\deg_{G_x}(v) \leq 4. \tag{D4}$$

Proof. Consider $v \in V_x = [2n]$. Given that $x \in \Gamma_n$, it follows that the set of indices $\{j \in [2n] : x_j = v\}$ contains precisely two elements, which we denote by α and β. By construction, $\alpha \neq \beta$, $x_\alpha = x_\beta = v$, and for any γ such that $x_\gamma = v$, we have $\gamma \in \{\alpha, \beta\}$.

Let $u \in V$ satisfy $\{u, v\} \in E$, i.e. u and v are neighbours in the BPSP graph. Then we have that

$$\begin{aligned} & \{u, x_\alpha\} \in E \quad \text{or} \quad \{u, x_\beta\} \in E \\ \implies \quad & u = x_{\alpha-1} \quad \text{or} \quad u = x_{\alpha+1} \quad \text{or} \quad u = x_{\beta-1} \quad \text{or} \quad u = x_{\beta+1} \\ \implies \quad & u \in \{x_{\alpha-1}, x_{\alpha+1}, x_{\beta-1}, x_{\beta+1}\}. \end{aligned} \tag{D5}$$

Hence, $\{u \in V : \{u, v\} \in E\} \subseteq \{x_{\alpha-1}, x_{\alpha+1}, x_{\beta-1}, x_{\beta+1}\}$, which implies that

$$\begin{aligned} \deg_{G_x}(v) &= |\{u \in V : \{u, v\} \in E\}| \\ &\leq |\{x_{\alpha-1}, x_{\alpha+1}, x_{\beta-1}, x_{\beta+1}\}| \\ &\leq 4, \end{aligned} \tag{D6}$$

which completes the proof of the proposition. □

b. Total weight of BPSP graph

Next, for any weighted graph $G = (V, E, W)$ (where V is the set of vertices, E is the set of edges, and $W : E \to \mathbb{R}$ is a weight function), we define the total weight of G as the sum of the weights of all its edges:

$$W_{\text{tot}}(G) = \sum_{\{i,j\} \in E} W(\{i, j\}). \tag{D7}$$

Additionally, we introduce the concept of the *double letter count* of a BPSP instance $x = (x_1, x_2, \ldots, x_{2n}) \in \Gamma_n \subseteq [n]^{2n}$, which is defined as the number of consecutive identical symbols in x, denoted as:

$$\#_{\text{dl}}(x) = |\{i \in [2n-1] : x_i = x_{i+1}\}|. \tag{D8}$$

Our next proposition expresses the total weight of a BPSP graph G_x using eq. (D8) and the eta function defined in eq. (B1).

Proposition 9. *Given a BPSP instance $x \in \Gamma_n$, the total weight of the BPSP graph $G_x = (V_x, E_x, W_x)$ associated with x is given by*

$$W_{\text{tot}}(G_x) = -\sum_{\substack{k=1 \\ x_k \neq x_{k+1}}}^{2n-1} (-1)^{\eta(x,k)} = -\#_{\text{dl}}(x) - \sum_{k=1}^{2n-1} (-1)^{\eta(x,k)}. \tag{D9}$$

Proof. To establish the first equality presented in eq. (D9), we proceed with the following calculation:

$$
\begin{aligned}
W_{\text{tot}}\left(G_x\right) &= \sum_{\{a,b\}\in E_x} W_x(\{a,b\}) \\
&= \sum_{\{a,b\}\in E_x} \theta_x(\{a,b\}) \\
&= \sum_{\substack{i\in\{2n-1]\\ x_i\neq x_{i+1}\\ \theta_x(\{x_i,x_{i+1}\})\neq 0}} \frac{\theta_x\left(\{x_i,x_{i+1}\}\right)}{\mid \{j\in[2n-1]:\{x_j,x_{j+1}\}=\{x_i,x_{i+1}\}\mid} \\
&= \sum_{\substack{i\in[2n-1]\\ x_i\neq x_{i+1}}} \frac{\theta_x\left(\{x_i,x_{i+1}\}\right)}{\mid \{j\in[2n-1]:\{x_j,x_{j+1}\}=\{x_i,x_{i+1}\}\mid} \\
&= \sum_{\substack{i\in[2n-1]\\ x_i\neq x_{i+1}}} \frac{-\sum_{k=1}^{2n-1}(-1)^{\eta(x,k)}\delta_{\{x_i,x_{i+1}\},\{x_k,x_{k+1}\}}}{\mid \{j\in\{2n-1]:\{x_j,x_{j+1}\}=\{x_i,x_{i+1}\}\mid} \\
&= -\sum_{k=1}^{2n-1}(-1)^{\eta(x,k)} \sum_{\substack{i\in[2n-1]\\ x_i\neq x_{i+1}}} \frac{\delta_{\{x_i,x_{i+1}\},\{x_k,x_{k+1}\}}}{\mid \{j\in[2n-1]:\{x_j,x_{j+1}\}=\{x_i,x_{i+1}\}\mid} \\
&= -\sum_{k=1}^{2n-1}(-1)^{\eta(x,k)} \sum_{i\in[2n-1]} [x_i\neq x_{i+1}] \frac{\delta_{\{x_i,x_{i+1}\},\{x_k,x_{k+1}\}}}{\mid \{j\in[2n-1]:\{x_j,x_{j+1}\}=\{x_i,x_{i+1}\}\mid} \\
&= -\sum_{k=1}^{2n-1}(-1)^{\eta(x,k)} \frac{[x_k\neq x_{k+1}]}{\mid \{j\in[2n-1]:\{x_j,x_{j+1}\}=\{x_k,x_{k+1}\}\mid} \sum_{i\in[2n-1]} \delta_{\{x_i,x_{i+1}\},\{x_k,x_{k+1}\}} \\
&= -\sum_{k=1}^{2n-1}(-1)^{\eta(x,k)} \frac{[x_k\neq x_{k+1}]}{\mid \{j\in[2n-1]:\{x_j,x_{j+1}\}=\{x_k,x_{k+1}\}\mid} |\{i\in[2n+1]:\{x_{i,},x_{i+1}\}=\{x_k=x_{k+1}\}| \\
&= -\sum_{\substack{k=1\\ x_k\neq x_{k+1}}}^{2n-1}(-1)^{\eta(x,k)}.
\end{aligned}
\tag{D10}
$$

The first line above follows directly from the definition of total weight given in eq. (D7). The second line stems from the relationship $W_x = \theta_x|_{E_x}$, as defined in Definition 7. The third line follows from the observation that if a finite set $T = \{a_i : i \in M\}$ (where M is also a finite set) contains elements a_i, with the possibility of some elements being identical, then for any function $f : T \to \mathbb{R}$:

$$
\sum_{t\in T} f(t) = \sum_{i\in M} \frac{f(a_i)}{|\{j : a_i = a_j\}|}. \tag{D11}
$$

This indicates that when summing over a set with potentially repeated elements, each distinct element contributes its value divided by the number of times it appears in the set. The fourth line arises from the observation that the summand in the expression vanishes whenever $\theta_x(\{x_i, x_{i+1}\}) = 0$. The fifth line follows from the definition of θ_x provided in eq. (D1). The sixth line is attained by rearranging the terms presented on the fifth line. The seventh line ensues from the property that if $S \subseteq T$ are any two indexing sets, then the following sums are equivalent:

$$
\sum_{i\in S}(\ \cdot\ ) = \sum_{i\in T}[i\in S](\ \cdot\ ). \tag{D12}
$$

This enables us to represent the sum over the potentially smaller set S as the sum over its superset T by incorporating an indicator function. The eighth line follows from the observation that the sum in the expression vanishes whenever the sets $\{x_i, x_{i+1}\}$ and $\{x_k, x_{k+1}\}$ are not equal. The ninth line stems from the property that

$$
\sum_i \delta_{a_i a_j} f(a_j) = f(a_j)|\{i : a_i = a_j\}|.
$$

Finally, the tenth line is derived by applying eq. (D12) once more. This concludes the proof of the first equality.

To show the second equality of eq. (D9), consider:

$$\begin{aligned}\sum_{\substack{k=1\\x_k\neq x_{k+1}}}^{2n-1}(-1)^{\eta(x,k)} &= \left(\sum_{k=1}^{2n-1}-\sum_{\substack{k=1\\x_k=x_{k+1}}}^{2n-1}\right)(-1)^{\eta(x,k)}\\ &= \sum_{k=1}^{2n-1}(-1)^{\eta(x,k)}-\sum_{\substack{k=1\\x_k=x_{k+1}}}^{2n-1}(-1)^{\eta(x,k)}\\ &= \sum_{k=1}^{2n-1}(-1)^{\eta(x,k)}+|\{i\in[2n-1]:x_i=x_{i+1}\}|\\ &= \sum_{k=1}^{2n-1}(-1)^{\eta(x,k)}+\#_{\mathrm{dl}}(x). \end{aligned} \tag{D13}$$

Negating the equalities in eq. (D13) yields the second equality in eq. (D9).

The first line above arises from the observation that the indices k satisfying $x_k = x_{k+1}$ and those satisfying $x_k \neq x_{k+1}$ form a partition of the index set $[2n-1]$. The second line follows from the distributive property. The third line is deduced from the observation that the second summation is exclusively over terms where $x_k = x_{k+1}$, thereby implying $\eta(x,k) = 1$, as shown by Lemma 2. Finally, the fourth line follows from the definition of the double letter count as defined by eq. (D8). □

c. *Relating total weight of BPSP graph to BPSP cost of red-first colouring*

Our next corollary expresses the BPSP cost of the red-first colouring, calculated in eq. (B7), in terms of the double letter count as defined in eq. (D8) and the total weight of the BPSP graph defined in eq. (D7).

Corollary 10. *Let $x \in \Gamma_n$. Then the BPSP cost of the red-first colouring of x is*

$$\xi_n\left(\sigma_{\mathrm{RF}}(x)\right) = n-\frac{1}{2}+\frac{1}{2}\#_{\mathrm{dl}}(x)+\frac{1}{2}W_{\mathrm{tot}}\left(G_x\right). \tag{D14}$$

Proof. From eq. (B7), we derive the BPSP cost of the red-first colouring of x as follows:

$$\xi_n\left(\sigma_{\mathrm{RF}}(x)\right) = n-\frac{1}{2}-\frac{1}{2}\underbrace{\sum_{i=1}^{2n-1}(-1)^{\eta(x,i)}}_{=-\#_{\mathrm{dl}}(x)-W_{\mathrm{tot}}(G_x)} = n-\frac{1}{2}+\frac{1}{2}\#_{\mathrm{dl}}(x)+\frac{1}{2}W_{\mathrm{tot}}\left(G_x\right),$$

where the equality within the underbraces above is obtained from eq. (D9). □

3. Reducing BPSP to Weighted MaxCut

In this section, we present a polynomial-time reduction from the BPSP problem to the weighted maximum cut (MaxCut) problem. We begin by introducing the necessary terminology for defining the MaxCut problem: Consider a weighted graph $G = (V, E, W)$, where $V = [n]$ for simplicity and $W : E \to \mathbb{R}$ represents the edge weights. A *cut* of G is defined as a binary string $z \in \{r, b\}^n$. The *weight* of a cut $z \in \{r, b\}^n$ in G is defined as

$$\mathrm{wt}_G(z) = \sum_{\{i,j\}\in E} W(\{i,j\})[z_i \neq z_j]. \tag{D15}$$

The MaxCut problem, whose goal is to identify the cut that maximises the total weight of the cut edges in the graph, can be expressed as follows:

Problem 2. Weighted Maximum Cut (MaxCut)

Input: a weighted graph $G = (V, E, W)$, where $V = [n]$ and $W : E \to \mathbb{R}$.
Output: maximize $\mathrm{wt}_G(z)$, as defined in eq. (D15)
subject to $z \in \{r, b\}^n$.

Also, we denote the optimal cost and the set of optimal solutions of Problem 2, respectively, as follows:

$$\textsc{MaxCut}(G) = \max_{z \in \{r,b\}^n} \mathrm{wt}_G(z) \tag{D16}$$

$$\textsc{MaxCut}^*(G) = \arg\max_{z \in \{r,b\}^n} \mathrm{wt}_G(z). \tag{D17}$$

Our next proposition provides a formula for computing the weight of a cut of a BPSP graph G_x for a given BPSP instance x, expressed in terms of the eta function.

Corollary 11. *Let $x \in \Gamma_n$ be a BPSP instance, and denote its BPSP graph as $G_x = (V_x, E_x, W_x)$. For a cut $z \in \{r, b\}^n$ of G_x, the weight of z in G_x is*

$$\mathrm{wt}_{G_x}(z) = -\sum_{k=1}^{2n-1} (-1)^{\eta(x,k)} \left[z_{x_k} \neq z_{x_{k+1}}\right]. \tag{D18}$$

Proof. To begin, we note the similarity between the formulae for the weight of a cut specified in eq. (D15) and the total weight of a graph given in eq. (D7), where the sole difference is the presence of the indicator function $[z_i \neq z_j]$ in each term of the sum in the former formula. Accordingly, our proof parallels that of Proposition 9:

$$\begin{aligned}
\mathrm{wt}_{G_x}(z) &= \sum_{\{a,b\} \in E_x} W_x(\{a,b\}) \left[z_a \neq z_b\right] \\
&= \sum_{\{a,b\} \in E_x} \theta_x(\{a,b\}) \left[z_a \neq z_b\right] \\
&= \sum_{\substack{i \in \{2n-1] \\ x_i \neq x_{i+1} \\ \theta_x(\{x_i, x_{i+1}\}) \neq 0}} \frac{\theta_x\left(\{x_i, x_{i+1}\}\right) \left[z_{x_i} \neq z_{x_{i+1}}\right]}{\mid \{j \in [2n-1] : \{x_j, x_{j+1}\} = \{x_i, x_{i+1}\} \mid} \\
&= \sum_{\substack{i \in [2n-1] \\ x_i \neq x_{i+1}}} \frac{\theta_x\left(\{x_i, x_{i+1}\}\right) \left[z_{x_i} \neq z_{x_{i+1}}\right]}{\mid \{j \in [2n-1] : \{x_j, x_{j+1}\} = \{x_i, x_{i+1}\} \mid} \\
&= \sum_{\substack{i \in \{2n-1] \\ x_i \neq x_{i+1}}} \frac{1}{\mid \{j \in \{2n-1] : \{x_j, x_{j+1}\} = \{x_i, x_{i+1}\} \mid} \left\{ -\sum_{k=1}^{2n-1} (-1)^{\eta(x,k)} \delta_{\{x_i, x_{i+1}\}, \{x_k, x_{k+1}\}} \right\} \left[z_{x_i} \neq z_{x_{i+1}}\right] \\
&= -\sum_{k=1}^{2n-1} (-1)^{\eta(x,k)} \sum_{\substack{i \in [2n-1] \\ x_i \neq x_{i+1}}} \frac{\delta_{\{x_i, x_{i+1}\}, \{x_k, x_{k+1}\}}}{\mid \{j \in [2n-1] : \{x_j, x_{j+1}\} = \{x_i, x_{i+1}\} \mid} \left[z_{x_i} \neq z_{x_{i+1}}\right] \\
&= -\sum_{k=1}^{2n-1} (-1)^{\eta(x,k)} \sum_{i \in [2n-1]} [x_i \neq x_{i+1}] \frac{\delta_{\{x_i, x_{i+1}\}, \{x_k, x_{k+1}\}}}{\mid \{j \in [2n-1] : \{x_j, x_{j+1}\} = \{x_i, x_{i+1}\} \mid} \left[z_{x_i} \neq z_{x_{i+1}}\right] \\
&= -\sum_{k=1}^{2n-1} (-1)^{\eta(x,k)} \frac{[x_k \neq x_{k+1}] \left[z_{x_k} \neq z_{x_{k+1}}\right]}{\mid \{j \in [2n-1] : \{x_j, x_{j+1}\} = \{x_k, x_{k+1}\} \mid} \sum_{i \in [2n-1]} \delta_{\{x_i, x_{i+1}\}, \{x_k, x_{k+1}\}} \\
&= -\sum_{k=1}^{2n-1} (-1)^{\eta(x,k)} \frac{[x_k \neq x_{k+1}] \left[z_{x_k} \neq z_{x_{k+1}}\right]}{\mid \{j \in [2n-1] : \{x_j, x_{j+1}\} = \{x_k, x_{k+1}\} \mid} \mid \{i \in [2n+1] : \{x_{i,}, x_{i+1}\} = \{x_k, x_{k+1}\} \mid \\
&= -\sum_{\substack{k=1 \\ x_k \neq x_{k+1}}}^{2n-1} (-1)^{\eta(x,k)} \left[z_{x_k} \neq z_{x_{k+1}}\right]
\end{aligned}$$

$$= -\sum_{k=1}^{2n-1} (-1)^{\eta(x,k)} \left[z_{x_k} \neq z_{x_{k+1}}\right].$$

The first line above stems from the definition of the weight of a cut, as provided in eq. (D15). Subsequent lines 2 through 7, and 9 through 10, are derived from the same line of reasoning as the chain of equalities presented in eq. (D10).

To show that the eighth line follows from the seventh, it suffices to prove that if

$$\{x_i, x_{i+1}\} = \{x_k, x_{k+1}\}, \tag{D19}$$

then

$$[x_i \neq x_{i+1}][z_{x_i} \neq z_{x_{i+1}}] = [x_k \neq x_{k+1}][z_{x_k} \neq z_{x_{k+1}}]. \tag{D20}$$

To prove this, let us assume that eq. (D19) holds. Then the ordered pair (x_i, x_{i+1}) is equal to either (x_k, x_{k+1}) or (x_{k+1}, x_k). In either case, $x_i = x_{i+1}$ holds if and only if $x_k = x_{k+1}$ holds. Hence,

$$[x_i \neq x_{i+1}] = [x_k \neq x_{k+1}]. \tag{D21}$$

Similarly, eq. (D19) implies that the ordered pair $(z_{x_i}, z_{x_{i+1}})$ is equal to either $(z_{x_k}, z_{x_{k+1}})$ or $(z_{x_{k+1}}, z_{x_k})$. In either case, $z_{x_i} \neq z_{x_{i+1}}$ holds if and only if $z_{x_k} \neq z_{x_{k+1}}$ holds. Hence,

$$[z_{x_i} \neq z_{x_{i+1}}] = [z_{x_k} \neq z_{x_{k+1}}]. \tag{D22}$$

Combining eq. (D21) and eq. (D22) yields eq. (D20). This concludes our proof that eq. (D19) implies eq. (D20), thereby completing the derivation of the eighth line.

Finally, the derivation of the eleventh line follows from the following observation: when $x_k = x_{k+1}$, it follows that $z_{x_k} = z_{x_{k+1}}$, yielding $[z_{x_k} \neq z_{x_{k+1}}] = 0$. This implies that the terms where $x_k \neq x_{k+1}$ make no contribution to the sum. Consequently, whether these terms are included or not has no bearing on the sum, affirming the equivalence of the tenth and eleventh lines. □

Our next objective is to establish a formula that connects $\tilde{\xi}_n(x, z)$—the measure of colour swaps required in colouring the BPSP instance x with the colouring z, defined in eq. (C14)—to the weight of the cut z in the BPSP graph G_x. To achieve this, we note that the expression for $\tilde{\xi}_n(x, z)$ that was derived in eq. (C15) contains terms of the form $[\neg^a u \neq v]$. To simplify these terms, our subsequent lemma establishes an identity that re-expresses them in a more useful form:

Lemma 12. *For any $u, v \in \{r, b\}$ and $a \in \{0, 1\}$, the following equality holds:*

$$[\neg^a u \neq v] = a + (-1)^a [u \neq v]. \tag{D23}$$

Proof. We analyse the cases $a = 0$ and $a = 1$ separately, establishing the equivalence between the expressions on the left-hand side (LHS) and the right-hand side (RHS) of eq. (D23): When $a = 0$,

$$\text{RHS} = 0 + (-1)^0 [u \neq v] = [u \neq v] = \text{LHS}. \tag{D24}$$

When $a = 1$,

$$\text{RHS} = 1 + (-1)[u \neq v] = 1 - [u \neq v] = [u = v] = [\neg u \neq v] = \text{LHS}. \tag{D25}$$

In both cases, we observe that the RHS equals the LHS, thus establishing the validity of the lemma. □

We are now poised to leverage Lemma 12 to express the number of colour swaps $\tilde{\xi}_n(x, z)$ in terms of the cut weight $\text{wt}_{G_x}(z)$. Strikingly, we find that $\tilde{\xi}_n(z, x)$ is precisely the difference between $\xi_n(\sigma_{\text{RF}}(x))$—the number of colour swaps in the red-first encoding—and the cut weight $\text{wt}_{G_x}(z)$:

Proposition 13. *For any $z \in \{r, b\}^n$ and $x \in \Gamma_n$,*

$$\tilde{\xi}_n(x, z) = \xi_n(\sigma_{\text{RF}}(x)) - \text{wt}_{G_x}(z). \tag{D26}$$

Proof. To prove the proposition, we carry out the following calculation:

$$\begin{aligned}\tilde{\xi}_n(x,z) &= \sum_{i=1}^{2n-1}\left[\neg^{\eta(x,i)} z_{x_i} \neq z_{x_{i+1}}\right] \\ &= \sum_{i=1}^{2n-1}\left\{\eta(x,i) + (-1)^{\eta(x,i)}\left[z_{x_i} \neq z_{x_{i+1}}\right]\right\} \\ &= \sum_{i=1}^{2n-1}\eta(x,i) + \sum_{i=1}^{2n-1}(-1)^{\eta(x,i)}\left[z_{x_i} \neq z_{x_{i+1}}\right] \\ &= \xi_n\left(\sigma_{\mathrm{RF}}(x)\right) - \mathrm{wt}_{G_x}(z).\end{aligned}$$

The first line above corresponds to eq. (C15). The second line follows from utilising Lemma 12 to rewrite the indicator function as a sum of two terms. The third line rearranges the second line into a sum of two terms, ensuring that the first term equals $\xi_n\left(\sigma_{\mathrm{RF}}(x)\right)$ by eq. (B7), and the second term equals $-\mathrm{wt}_{G_x}(z)$ by eq. (D18). From this rearrangement, we observe that the fourth line ensues. □

Next, note that eq. (D14) provides an equivalent expression for the number of colour swaps $\xi_n\left(\sigma_{\mathrm{RF}}(x)\right)$ in the red-first encoding. By substituting eq. (D14) into eq. (D26), we obtain the following corollary:

Corollary 14. *For any $z \in \{r,b\}^n$ and $x \in \Gamma_n$,*

$$\tilde{\xi}_n(x,z) = n - \frac{1}{2} + \frac{1}{2}\#_{\mathrm{dl}}(x) + \frac{1}{2}W_{\mathrm{tot}}\left(G_x\right) - \mathrm{wt}_{G_x}(z). \tag{D27}$$

The above equation expresses the number of colour swaps involved when x is coloured with z in terms of the double-letter count of x, the total weight of the BPSP graph G_x, and the weight of the cut z in G_x.

Next, by optimising eq. (D27) over the cuts $z \in \{r,b\}^n$, we obtain the following corollary:

Corollary 15. *Let $x \in \Gamma_n$. Then, we have that*

$$\mathrm{BPSP}(x) = \xi_n\left(\sigma_{\mathrm{RF}}(x)\right) - \textsc{MaxCut}(G_x) = n - \frac{1}{2} + \frac{1}{2}\#_{\mathrm{dl}}(x) + \frac{1}{2}W_{\mathrm{tot}}\left(G_x\right) - \textsc{MaxCut}\left(G_x\right), \tag{D28}$$

$$\widetilde{\mathrm{BPSP}}^*(x) = \textsc{MaxCut}^*\left(G_x\right). \tag{D29}$$

Proof. Let $x \in \Gamma_n \subset \Gamma$. Then eq. (D28) holds because

$$\begin{aligned}\mathrm{BPSP}(x) &= \min_{z\in\{r,b\}^n}\tilde{\xi}_n(x,z) = \min_{z\in\{r,b\}^n}\left[\xi_n(\sigma_{\mathrm{RF}}(x)) - \mathrm{wt}_{G_x}(z)\right] \\ &= \xi_n(\sigma_{\mathrm{RF}}(x)) - \max_{z\in\{r,b\}^n}\mathrm{wt}_{G_x}(z) \\ &= \xi_n(\sigma_{\mathrm{RF}}(x)) - \textsc{MaxCut}(G_x) = n - \frac{1}{2} + \frac{1}{2}\#_{dl}(x) + \frac{1}{2}W_{\mathrm{tot}}\left(G_x\right) - \textsc{MaxCut}\left(G_x\right),\end{aligned}$$

where the last equality follows from eq. (D14). Similarly, eq. (D29) holds because

$$\begin{aligned}\widetilde{\mathrm{BPSP}}^*(x) &= \arg\min_{z\in\{r,b\}^n}\tilde{\xi}_n(x,z) = \arg\min_{z\in\{r,b\}^n}\left[\xi_n(\sigma_{\mathrm{RF}}(x)) - \mathrm{wt}_{G_x}(z)\right] \\ &= \arg\min_{z\in\{r,b\}^n}\left(-\mathrm{wt}_{G_x}(z)\right) = \arg\max_{z\in\{r,b\}^n}\left(\mathrm{wt}_{G_x}(z)\right) = \textsc{MaxCut}^*(G_x).\end{aligned}$$

This completes the proof of the corollary. □

Finally, we note that Corollary 15 provides us with a polynomial-time reduction from the BPSP problem stated in Problem 1 to the weighted MaxCut problem presented in Problem 2. Specifically, one use of an oracle for Problem 2 enables us to solve Problem 1 in polynomial time, as we demonstrate in Algorithm 2.

Algorithm 2: Polynomial-time reduction from BPSP (Problem 1) to MaxCut (Problem 2)

Input: $x \in \Gamma$
Output: $\mathrm{BPSP}(x)$ and $\mathrm{BPSP}^*(x)$
Construct G_x, the BPSP graph of x, utilising Definition 7.
Execute the weighted MaxCut oracle for Problem 2 on G_x to obtain $\mathrm{MaxCut}(G_x)$ and $z^* \in \mathrm{MaxCut}^*(G_x)$.
Compute $\xi_n(\sigma_{\mathrm{RF}}(x))$ using eq. (D14).
Compute $f^* = \mathcal{E}(x, z^*)$ using eq. (C1).
return $\mathrm{BPSP}(x) = \xi_n(\sigma_{\mathrm{RF}}(x)) - \mathrm{MaxCut}(G_x)$ and $\mathrm{BPSP}^*(x) = f^*$.

Appendix E: Ising Formulation of BPSP

In the Ising formulation, we assign numerical values to the formal symbols r and b:

$$r \equiv 1, \qquad b \equiv -1. \tag{E1}$$

In the following lemma, we present several identities that will prove useful for translating formulas in appendix D into an algebraic form:

Lemma 16. *Let $y, z \in \{r, b\}$, where $r = 1$ and $b = -1$, and $a \in \{0, 1\}$. Then, we have:*

(i) $\neg z = -z$.

(ii) $\neg^a z = (-1)^a z$.

(iii) $[y = z] = \frac{1}{2}(1 + yz)$.

(iv) $[y \neq z] = \frac{1}{2}(1 - yz)$.

(v) $[\neg^a z \neq y] = \frac{1}{2} - \frac{1}{2}(-1)^a zy$.

Proof. To prove these identities, it suffices to check that equality holds for each of the possible discrete binary values of y, z, and a. Indeed, (i) holds since $\neg r = b = -1 = -r$ and $\neg b = r = 1 = -b$. (ii) holds because $a = 0$ implies that $\neg^a z = z = (-1)^0 z$ and $a = 1$ implies that $\neg^a z = \neg z = -z = (-1)^1 z$. (iii) holds because

$$yz = \begin{cases} 1, & y = z \\ -1, & y \neq z \end{cases} \implies \frac{1}{2}(1 + yz) = \begin{cases} 1, & y = z \\ 0, & y \neq z \end{cases} = [y = z]. \tag{E2}$$

(iv) holds since $[y \neq z] = 1 - [y = z] = 1 - \frac{1}{2}(1 + yz) = \frac{1}{2}(1 - yz)$. Finally, (v) holds because $[\neg^a z \neq y] = [(-1)^a z \neq y] = \frac{1}{2}\left(1 - (-1)^a zy\right) = \frac{1}{2} - \frac{1}{2}(-1)^a zy$. □

1. Ising Expressions for BPSP Cost Function

We now present a formula for the colour swap tally $\tilde{\xi}_n(x, z)$ from eq. (C15) in terms of the Ising formulation.

Proposition 17. *Consider $z \in \{-1, 1\}^n$ and $x \in \Gamma_n$. Then, we can express the number of colour swaps required when x is coloured with z as*

$$\tilde{\xi}_n(x, z) = n - \frac{1}{2} - \frac{1}{2} \sum_{i=1}^{2n-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}} \tag{E3}$$

$$= n - \frac{1}{2} + \frac{1}{2} \#_{\mathrm{dl}}(x) - \frac{1}{2} \sum_{\substack{i=1 \\ x_i \neq x_{i+1}}}^{2n-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}}. \tag{E4}$$

Proof. By applying Lemma 16(v) to eq. (C15) and simplifying, we obtain:

$$\tilde{\xi}_n(x, z) = \sum_{i=1}^{2n-1} \left[\neg^{\eta(x,i)} z_{x_i} \neq z_{x_{i+1}}\right]$$

$$
\begin{aligned}
&= \sum_{i=1}^{2n-1} \frac{1}{2}\left(1-(-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}}\right) \\
&= n - \frac{1}{2} - \frac{1}{2}\sum_{i=1}^{2n-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}} \\
&= n - \frac{1}{2} - \frac{1}{2}\left(\sum_{\substack{i=1\\ x_i=x_{i+1}}}^{2n-1} + \sum_{\substack{i=1\\ x_i\neq x_{i+1}}}^{2n-1}\right)(-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}} \\
&= n - \frac{1}{2} - \frac{1}{2}\sum_{\substack{i=1\\ x_i\neq x_{i+1}}}^{2n-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}} - \frac{1}{2}\sum_{\substack{i=1\\ x_i=x_{i+1}}}^{2n-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}} \\
&= n - \frac{1}{2} - \frac{1}{2}\sum_{\substack{i=1\\ x_i\neq x_{i+1}}}^{2n-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}} + \frac{1}{2}\left|\{i\in[2n-1] : x_i = x_{i+1}\}\right| \\
&= n - \frac{1}{2} - \frac{1}{2}\sum_{\substack{i=1\\ x_i\neq x_{i+1}}}^{2n-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}} + \frac{1}{2}\#_{\mathrm{dl}}(x),
\end{aligned}
$$

where the sixth line follows from the observation that when $x_i = x_{i+1}$, we have $\eta(x,i) = 1$ (by Lemma 2) and $z_{x_i} z_{x_{i+1}} = z_{x_i}^2 = 1$. The third and last lines provide the desired expressions eq. (E3) and eq. (E4), respectively. □

Finally, through optimisation of eq. (E3) and eq. (E4) over the cuts $z \in \{-1,1\}^n$, and noting that for $n = |x|/2$ for $x - \Gamma_n$, we arrive at the following corollary:

Corollary 18. *Let $x \in \Gamma$ be a BPSP instance. Then the optimal cost and the set of optimal solutions of the BPSP problem are, respectively,*

$$
\mathrm{BPSP}(x) = \frac{1}{2}\left\{|x| - 1 - \max_{z\in\{-1,1\}^{|x|/2}} \sum_{i=1}^{|x|-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}}\right\} \tag{E5}
$$

$$
= \frac{1}{2}\left\{|x| - 1 + \#_{\mathrm{dl}}(x) - \max_{z\in\{-1,1\}^{|x|/2}} \sum_{\substack{i=1\\ x_i\neq x_{i+1}}}^{|x|-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}}\right\} \tag{E6}
$$

$$
\widetilde{\mathrm{BPSP}}^*(x) = \operatorname*{arg\,max}_{z\in\{-1,1\}^{|x|/2}} \sum_{i=1}^{|x|-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}} \tag{E7}
$$

$$
= \operatorname*{arg\,max}_{z\in\{-1,1\}^{|x|/2}} \sum_{\substack{i=1\\ x_i\neq x_{i+1}}}^{|x|-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}}. \tag{E8}
$$

In essence, these expressions unveil the underlying connection between solving the BPSP and finding the ground state energy of the associated Ising Hamiltonian:

$$
H(z_1,\ldots,z_n) = -\sum_{\substack{i=1\\ x_i\neq x_{i+1}}}^{2n-1} (-1)^{\eta(x,i)} z_{x_i} z_{x_{i+1}}. \tag{E9}
$$