IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 26 July 2023, accepted 2 September 2023, date of publication 8 September 2023, date of current version 13 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3313102

APPLIED RESEARCH

Quadratic Unconstrained Binary Optimization for the Automotive Paint Shop Problem

PIETER DEBEVERE, MASAHIKO SUGIMURA, AND MATTHIEU PARIZY
Fujitsu Ltd., Kawasaki 211-8588, Japan
Corresponding author: Pieter Debevere (pieter.debevere@me.com)

ABSTRACT The Binary Paint Shop Problem (BPSP) is a combinatorial optimization problem which draws inspiration from the automotive paint shop. Its binary nature, making it a good fit for Quadratic Unconstrained Binary Optimization (QUBO) solvers, has been well studied but its industrial applications are limited. In this paper, in order to expand the industrial applications, QUBO formulations for two generalizations of the BPSP, which are the Multi-Car Paint Shop Problem (MCPSP) and the Multi-Car Multi-Color Paint Shop Problem (MCMCPSP), are proposed. Given the multiple colors, the MCMCPSP is no longer natively binary which increases the problem size and introduces additional constraint factors in the QUBO formulation. Resulting QUBOs are solved using Scatter Search (SS). Furthermore, extensions of the SS that can exploit k-hot constrained structures within the formulations are proposed to compensate the additional complexity introduced by formulating non-binary problems into QUBO. Since no public benchmark database currently exists, random problem instances are generated. Viability of the proposed QUBO solving methods for the MCPSP and MCMCPSP, is highlighted through comparison with an integer-based Random Parallel Multi-start Tabu Search (RPMTS) and a greedy heuristic for the problems. The greedy heuristic has negligible computational requirements and therefore serves as a lower bound on the desired performance. The results for both problems show that better results can be obtained than the greedy heuristic and integer-based RPMTS, by using the novel k-hot extensions of the SS to solve the problems as QUBO.

INDEX TERMS Automotive paint shop, evolutionary algorithms, k-hot constraints, quadratic unconstrained binary optimization, scatter search.

I. INTRODUCTION

The demand for mass-customization in the automotive industry has presented numerous operational planning challenges for car manufacturers. This paper will focus on two of these problems encountered within the automotive paint shop and attempt to solve them as Quadratic Unconstrained Binary Optimization (QUBO). The binary paint shop problem (BPSP) is the most extensively studied problem within this context [1], [2], [3], [4], [5], and given its binary nature, is well suited to be solved as QUBO. Its restrictive formulation however, which considers only two colors and two cars per model, limits its applicability in industry [5]. This paper, therefore, examines two generalizations of the BPSP: the Multi-Car Paint Shop Problem (MCPSP), which extends the considered cars per model, and the Multi-Car Multi-Color Paint Shop Problem (MCMCPSP), which further extends the considered colors. Both problems were only recently introduced by Yarkoni et al. in 2021 [6] and are thus not yet extensively studied. Their research introduced both generalizations but focused exclusively on the binary MCPSP which they solved using D-wave's Quantum Annealer. The modelling and solving of the non-binary MCMCPSP was left up to future research.

QUBO has been proposed as a potential unifying model to tackle many combinatorial optimization problems encountered in operations research [7]. Contrary to what the naming would suggest, QUBO can be applied to more general optimization problems which often are constrained and non-binary. For many classical combinatorial optimization

The associate editor coordinating the review of this manuscript and approving it for publication was Hongwei Du.

problems, QUBO formulations have already been proposed [8], [9]. Furthermore, the similarity between QUBO and the Ising Model makes it especially appealing for research, as this enables the problems to be solved by Ising machines such as the D-wave Quantum Annealer [10] and Fujitsu Quantum-inspired Computing Digital Annealer [11], as done by Yarkoni et al. [6] to solve the MCPSP. The general QUBO formulation is defined in (1). $Q \in \mathbb{R}^{n \times n}$ is a symmetric matrix and the goal is to minimize the energy E of the model over the binary variable vector x.

$$\min_{x \in \{0,1\}^n} E = x^T Q x \tag{1}$$

Solving non-binary and/or constrained problems involves reformulation techniques to turn the problems into QUBO. These reformulation techniques tend to increase the problem size and add additional constraint factors in the energy of the system which can obscure the main objective [8], [9]. Consequently, it becomes challenging to identify problems suitable for QUBO as well as efficient algorithms that outperform classical approaches utilizing the original problem representation.

The choice is made to solve the problems with a self-implemented QUBO solving algorithm instead of a dedicated high-end commercial QUBO solver, as this allows for the implementation and analysis of solving capabilities which are not yet available in commercial QUBO solvers. The chosen algorithm is a Scatter Search (SS) that uses Univariate Marginal Distribution Algorithm (UMDA) to combine solutions. For this SS, the performance improvement was shown in [12] compared to an existing SS that uses path relinking to combine solutions, which has been reported to be competitive on QUBO problems [13].

Since not only the binary MCPSP but also the non-binary MCMCPSP are solved within this paper, the SS is extended to be able to exploit the constrained structure of the considered problems to compensate for the additional complexity introduced by formulating non-binary problems into QUBO. The constrained structure the extended SS exploits are 1-hot constraints and the more general k-hot constraints, as these often appear in QUBO formulations of other well-known optimization problems such as permutation problems and assignment problems [8], [9]. These constraints link certain binary variables within the problem, allowing only k of them to be equal to one, hence reducing the actual feasible binary solution space. Exploitation of these constraints can therefore considerably improve the solving capabilities. It is no surprise that current high-end commercial QUBO solver have implemented or are researching such capabilities. However to the best of our knowledge, research and implementation has mostly been limited to 1-hot constraints and not the more general k-hot constraint, as is for example the case for 3rd and 4th generation of Fujitsu Quantum-inspired Computing Digital Annealer (DA) [11].

The results of the QUBO solving methods are compared to classical algorithms that use the natural representation of the problems. The first method is a self-implemented integer-based Random Parallel Multi-start Tabu Search (RPMTS), which performs multiple random-restart iterations of parallel tabu searches to fully utilize the CPU cores. This is the simplest version of a Multi-start Tabu Search (MTS), which have shown to be competitive for the Quadratic Assignment Problem (QAP) [14]. A second method is multi-color extension of the Black-first heuristic used by Yarkoni et al. to solve the MCPSP [6], which can also be used to solve the MCMCPSP.

The research of Yarkoni et al. in 2021 [6] was performed in collaboration with the Volkswagen Group and includes real industrial data which are not publicly available. To the best of our knowledge, there is currently no public dataset available that can be used to study the MCPSP and MCMCPSP. Therefore, randomly generated problem instances are used within the paper.

The main contribution within this paper is the modelling and solving of novel QUBO formulations to two generalizations of the BPSP which are more applicable to industry. Additionally, novel extensions are proposed for the self-implemented QUBO solving method to exploit k-hot constraints within the problems.

The rest of this paper is structured as follows. A review of literature is presented in Section II. The problem formulations are presented in Section III. A description of the solving methods used in this study follows in Section IV. The problem sets used and the experimental settings are presented in Section V. Results are analyzed in Section VI. Finally, conclusions and further work are discussed in Section VII.

II. PREVIOUS WORK

A. SCATTER SEARCH

Scatter Search (SS) is a metaheuristic framework which belongs to the class of Evolutionary Algorithms (EA). The framework was first proposed by Glover in 1977 [15]. Similar to other EAs such as Genetic Algorithms (GA) [16] and Differential Evolution (DE) [17], SS maintains a set of solutions which evolves over time. Within SS, this set of solutions is referred to as the *RefSet*. The SS framework contains five main procedures to reach the final solution. These steps are diversification generation, solution improvement, reference set update, subset generation, and solution combination [15]. One of the main ideas behind SS, is to guarantee a level of diversity in the solution set. This is realized by both the diversification generation and reference set update steps. During the diversification generation step, an initial set of promising solutions is generated and only a diverse subset of these solutions is selected to make the initial *RefSet*. During the iterations of the algorithm, new solutions are only included in the *RefSet* by the reference set update step if the diversity is maintained. This diversity allows to explore more valuable solutions during the solution combination step. Within SS, solutions are not only combined, but also locally improved during the solution improvement step. This step is commonly realized by Tabu Search (TS) [18]. Because of this, SS is viewed as a sibling of TS. SS has already

been applied to QUBO problems and shown to produce competitive results [13], [19]. Exploitation of k-hot constraints for binary problems when using SS has however remained limited to so-called multiple-choice problems where only k out of all the binary variables are allowed to be one [20]. Exploitation of more general k-hot constraints, containing only a subset of binary variables has not yet been studied. For an extensive review of the SS framework, its steps and possible applications, we refer the reader to [21].

B. PAINT SHOP PROBLEMS

The origins of the MCPSP and MCMCPSP can be traced back to the Paint Shop Problem for Words (PPW), which was first introduced in 2004 by Epping et al. [1]. Within the PPW, a word consisting of colored letters is considered. The goal is to find the optimal permutation of the colored letters such that the number of color switches is minimized, while also assuring that the word itself remains the same. Hence, the new permutation needs to have the same letters in the same positions but they can have different colors. Linking this formulation back to the automotive paint shop, the letters represent different car models and the word represents a fixed sequence in which they enter the paint shop. In [1], Epping et al. showed that this problem is NP-complete in both the number of car models and number of colors. Additionally, they showed that a polynomial-time dynamic programming solution exists if the number of car models and colors are bounded. However, given the expected values for these parameters in industry, this dynamic programming solution quickly becomes inapplicable in practice.

Epping et al. also introduced a restricted version of the PPW called k-regular PPW. For this problem, every letter appears k times per color. The BPSP is such a k-regular instance, where k is equal to one and only two colors are being considered. Hence, every car model in the fixed sequence appears twice and needs to be painted once in each color. Even though the BPSP seems very simple, subsequent research by Bonsma et al. in 2006 [2] showed that even this highly restricted version of the PPW is still NP-complete and even APX-hard. Efforts have been made to find heuristics for the BPSP with minimal expected color switches which show good results [4]. Recently these heuristics have been beaten by the use of a Quantum Algorithm in research performed by Streif et al. in 2021 [5].

Due to the restrictive nature of the BPSP, its applicability to industry is limited. Subsequent research of Yarkoni et al. [6] therefore, introduced two new extension to the BPSP called the Multi-Car Paint Shop Problem (MCPSP) and the Multi-Car Multi-Color Paint Shop Problem (MCMCPSP). The first extension removes the 1-regular nature of the BPSP and allows the color demand per car model to be irregular. Car models no longer appear in pairs but still only two colors are considered. The second extension additionally removes the restriction of only two colors, making the problem no longer natively binary. The research of Yarkoni et al. [6] uses an Ising machine to solve the problem and only solved the MCPSP. The research showed that the Ising machines are well suited for smaller to intermediate sized problems. However for large problem sizes, the performance quickly approached that of a simple greedy algorithm. The multi-color extension of the problem, due to its non-native binary nature, was not yet solved and left as further research, leading to the current paper.

C. K-HOT CONSTRAINED PROBLEMS

The reformulation techniques to tackle constrained and/or non-native binary problems as QUBO, result in an increased solution space and the introduction of additional constraint factors in the energy of the system [8], [9]. These additional energy terms often originate from k-hot constraints on the binary variables. K-hot constraints, as the name implies, only allow k binary variables within a set S to be equal to one, hence limiting the actual feasible solution space of the problem. In QUBO formulations, the deviations from these k-hot requirements are introduced in the energy of the system, multiplied with a penalty coefficient P. By doing this, the problem can be solved as an ''unconstrained'' problem. By properly scaling the penalty coefficient, the energy landscape of the system can be tuned to make sure that any feasible solution is preferred over an infeasible one. An example of a k-hot constraint and the resulting constraint energy factor are given by (2) and (3).

$$\sum_{i \in S} x_i = k \tag{2}$$

$$P \cdot (k - \sum_{i \in S} x_i)^2 \tag{3}$$

Given that most QUBO solving algorithms are based on single bit flips [22], [23], research to improve solvability of k-hot constrained problems has been limited to approaches which consider the QUBO solvers to be black-box entities. Efforts are made to reduce the overall solution space by finding smaller formulations [24] and using decomposition techniques [25], [26], [27], as well as trying to improve the energy landscape of the QUBO formulations by using data scaling techniques and penalty coefficient tuning [26], [28].

III. PROBLEM DEFINITION

Within this section, the problem definition and QUBO formulation for both extensions of the BPSP are provided. For a concrete definition of the BPSP itself, the reader is referred to [2]. Both The MCPSP and MCMCPSP originate from an automotive paint shop in which the following assumptions are made. The sequence of cars that enters the paint shop is assumed to be optimized for the final assembly step of the cars and is therefore fixed. The sequence of cars is made up of distinct car models and the color demand per model can vary. Within the paint shop, the cars go through two painting steps. The first painting step applies a filler paint layer, which can either be black or white. The second painting step applies a base paint layer which depends on the desired final color of the car. The set of possible base paints can be split into two groups; those that require the white filler paint and those

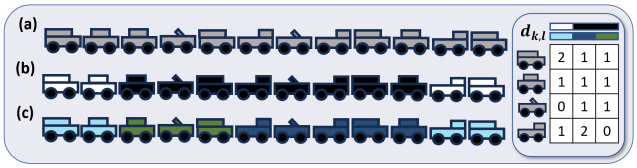

FIGURE 1. Example scenario of the paint shop with a fixed sequence of 12 cars (a) containing four distinct models and three base paint colors with the demand per model k and color l given by matrix $d_{k,l}$. A solution to the base paint application (c) and its corresponding solution to the filler paint application (b).

that require the black filler paint. For every car model and every base color, the demand over the sequence is known. The two painting steps are assumed to be realized by separate spray nozzles. In respective painting steps, before a new color can be applied, the spray nozzles need to be cleaned which can be an expensive process. The goal is therefore, to find a color assignment over the sequence which minimizes the number of color switches for both the filler paint nozzle and base paint nozzle and also meets the required color demand for the different models. When only the filler paint nozzle optimization is considered, the definition of the MCPSP is obtained. When only the base paint nozzle optimization is considered, the definition of the MCMCPSP is obtained. An example scenario can be seen in Fig. 1.

It should be noted that in the described automotive paint shop, both problems are interdependent. Optimizing the filler paint nozzle will influence the base paint nozzle and vice versa, and should thus be considered at the same time. There is however some merit in studying both problems separately, as solving the problems sequentially is also possible. First, the filler paint nozzle is optimized by using the MCPSP formulation. This will result in a solution containing segments of consecutive white or black cars. Within these segments, the base paint nozzle can then be optimized using the MCMCPSP formulation.

A. MCPSP FORMULATION

The input for the MCPSP is a fixed sequence of N cars containing M unique models, and the number of cars that need to be painted black d_k for a certain model k. Additionally, the set of sequence positions S_k of a certain model k can be obtained from the input sequence. The MCPSP can be modelled as a QUBO problem by the introduction of the binary variables x_i which are equal to one if the car in position i of the sequence is painted black. The complete QUBO formulation for the problem can be derived from the Hamiltonian given in (6). This Hamiltonian includes only quadratic terms and can thus be converted to the QUBO formulation of (1). P_d is a penalty coefficient which can be tuned. Equation (4) represents the color switch objective. Equation (5) ensures the color demand per car model k.

$$H_{CS} = \sum_{i=0}^{N-2} (x_i + x_{i+1} - 2\, x_i x_{i+1}) \tag{4}$$

$$H_D(k) = (d_k - \sum_{i \in S_k} x_i)^2 \tag{5}$$

$$H = H_{CS} + P_d \sum_{k=0}^{M-1} H_D(k) \tag{6}$$

B. MCMCPSP FORMULATIONS

Similar to the MCPSP, the MCMCPSP's main input is a fixed sequence of N cars containing M unique models, where the set of sequence positions S_k of a certain model k can be obtained from that input sequence. Compared to the MCPSP, the MCMCPSP considers a set of C unique colors which results in demand $d_{k,l}$ per model k and color l. For the MCMCPSP, we introduce the binary variables $x_{i,j}$ which are equal to one if the car in position i of the sequence is painted in color j. The complete QUBO formulations can be derived from the Hamiltonian given in (10). This Hamiltonian also includes only quadratic terms and can thus be converted to the QUBO formulation of (1) after remapping the binary matrix variables $x_{i,j}$ to binary vector variables x_i. P_d and P_{1-hot} are penalty coefficients which can be tuned. Equation (7) represents the color switch objective. Equation (8) ensures the demand per car model k and color l. Equation (9) ensures that every car can only be painted in one color.

$$H_{CS} = \frac{1}{2} \sum_{i=0}^{N-2} \sum_{j=0}^{C-1} (x_{i,j} + x_{i+1,j} - 2\, x_{i,j} x_{i+1,j}) \tag{7}$$

$$H_D(k,l) = (d_{k,l} - \sum_{i \in S_k} x_{i,l})^2 \tag{8}$$

$$H_{1-hot} = \sum_{i=0}^{N-1} (1 - \sum_{j=0}^{C-1} x_{i,j})^2 \tag{9}$$

$$H = H_{CS} + P_d \sum_{k=0}^{M-1} \sum_{l=0}^{C-1} H_D(k,l) + P_{1-hot} H_{1-hot} \tag{10}$$

IV. SOLUTION METHODS

A. PROPOSED QUBO SOLVING METHODS

We extended the SS developed in [12] to exploit QUBO formulations that contain k-hot constraints. Extension of the SS is limited to the way solutions are generated during the diversification generation step, and the way solutions are locally improved during the solution improvement step. In both steps, measurement are taken to guarantee the validity of the k-hot constraints. However, during the solution combination step, the viability of the k-hot constraints is not guaranteed by UMDA. Therefore the solution improvement methods also have capabilities to first ensure that a solution satisfies the k-hot constraints by recycling as much of the info in the solution as possible, before local improvement starts.

The developed solution generation methods have all been implemented to generate random solutions that satisfy the required constraints. Given that k-hot constrained variables require minimal two bit flips to go from one feasible solution to another, solution improvement is realized by multi-bit TS implementations. The tabu elements are chosen to be the moves. The multi-bit flip move costs are calculated

using consecutive fast single-bit flip cost calculations to prevent the heavy matrix computation in (1). Given a solution $x \in \{0, 1\}^N$, the cost of every $x' \in \{0, 1\}^N$ which differs from x in exactly one entry can be evaluated without computing $x'^T Qx'$ [29]. Assuming that x and x' differ at index i and the current cost $x^T Qx$ is known, the cost of x' can efficiently be computed as $E(x') = x^T Qx + \Delta_i$ where Δ_i is given by

$$\Delta_i = (1 - 2x_i) \cdot (q_{ii} + \sum_{j=0, j\neq i}^{N-1} 2q_{ij}x_j). \tag{11}$$

A first novel multi-bit TS implementation we propose is k-hot TS, which allows the user to define multiple non-overlapping groups of binary variables that are linked together with a k-hot constraint per group. Not all binary variables need to be in a group, nor do the grouped variables need to be consecutive. The moves that the TS considers is either flipping a non-grouped variable (single-bit flip move) or swapping the position of a one and a zero in a group (two-bit flip move). The number of moves within a group of n binary variables is thus equal to $(n - k) \cdot k$. When k is equal to $\frac{n}{2}$, this results in $(n - \frac{n}{2}) \cdot \frac{n}{2} = \frac{n^2}{4}$ moves. The possible moves within the group thus scale polynomial in the worst case and tends to be larger than the 1-bit flip neighborhood of only n moves. A noticeable exception to this, is when k is equal to one. In this case, the number of possible moves within the group is only $(n-1)\cdot 1$, which is less than the 1-bit flip neighborhood.

A second novel multi-bit TS implementation we propose is 1-hot/k-hot TS, which allows the user to again define multiple non-overlapping groups of non-consecutive binary variables. However this time the grouped variables need to be formattable as 2D binary matrices. On every matrix, either row-wise or column-wise k-hot constraints are applied, with 1-hot constraints applied on the other remaining axis. The shape of the binary matrix, as well as the k-hot axis and the k values need to be provided by the user per group. The TS move either flips a non-grouped variable (single-bit flip move) or swaps the position of two ones in different rows and columns within a grouped binary variable matrix (four-bit flip move). The combined 1-hot and k-hot constraints divide the ones within the matrix into several distinct sets, one per k-hot constraints. The total possible moves within the group is thus given by every possible unique way of swapping the ones within two of these sets. In the worst case, such a grouped matrix containing n binary variables only contains two of these sets, with the two resulting k values being both equal to $\frac{n}{4}$. This results in $\frac{n^2}{16}$ possible moves, which again scales polynomial and tends to be larger than the 1-bit flip neighborhood of only n moves.

Even though this multi-bit TS implementation seems quite niche, this problem structure often occurs in assignment problems as well as the less general form where the k-hot constraints are also 1-hot constraints, resulting in the two-way one-hot constrained structure which commonly occurs in permutation problems [8], [9].

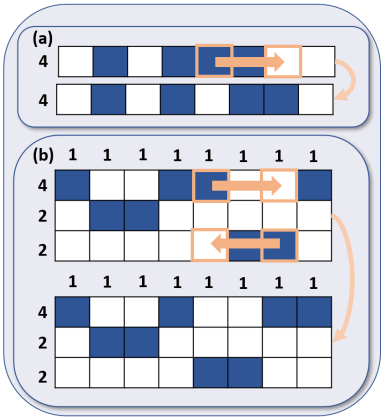

FIGURE 2. Multi-bit move example on grouped binary variables for the first novel TS implementation that exploits k-hot constraints (a) , and the second novel TS implementation that exploits 1-hot and k-hot constraints (b). Blue and white squares represent one- and zero-values respectively.

Examples of the multi-bit flip moves used within the novel TS implementations can be seen in Fig. 2. Given that both novel TS implementations exhibit the possibility of having to evaluate a large number of moves in every TS iteration, we also developed "capped" versions of these multi-bit TS implementations which confine the number of possible moves by considering only a subset of random selected moves in every TS iteration.

No-hot SS: This solving method refers to the basic SS framework developed in [12]. The TS implementation used within this SS variant hence uses only single-bit flip moves.

1-hot SS: This solving method refers to the SS framework which has been extended to exploit binary variables that are 1-hot constrained during the solution generation and the solution improvement step. The TS within this SS variant is realized by the first novel TS implementation, with the possible k values limited to one.

k-hot SS: This solving method refers to the SS framework which has been extended to exploit binary variables that are k-hot constrained during the solution generation and the solution improvement step. The TS within this SS variant is realized by the first novel TS implementation.

1-hot/k-hot SS: This solving method refers to the SS framework which has been extended to exploit binary variables that are both 1-hot constrained and k-hot constrained during the solution generation and the solution improvement step. The TS within this SS variant is realized by the second novel TS implementation.

A summary of the worst-case possible moves per TS iteration for a group of n bits for the different SS-based QUBO solving methods can be seen in Table 1.

B. NATURAL REPRESENTATION SOLVING METHODS

To support the viability of the aforementioned QUBO solving methods, a comparison with classical algorithms that use the natural representation of the problem is required.

TABLE 1. Worst-case possible moves per TS iteration for a group of n bits for the different SS-based QUBO solving methods.

SS Variant	Possible Moves	Scaling
no-hot	n	$O(n)$
1-hot	$(n-1)\cdot 1 = n-1$	$O(n)$
k-hot	$(n-\frac{n}{2})\cdot\frac{n}{2} = \frac{n^2}{4}$	$O(n^2)$
1-hot/k-hot	$\frac{n}{4}\cdot\frac{n}{4} = \frac{n^2}{16}$	$O(n^2)$

1) RPMTS

This solving method, uses multiple random-restart iteration of parallel TSs and is thus similar to the SS-based QUBO solving methods. A solution to the problems is represented by an integer vector where the integers represent the individual cars and their values represent the assigned colors. Compared to the QUBO solving methods, the cost of a solution in this integer-based method can not easily be calculated by using (1). Instead, the integer based solutions are explicitly checked for the number of color switches and whether or not they meet the required color demand. Any deviation from the required color demand is penalized by a factor P_D, similar to how this is done in the Hamiltonian's in (6) and (10). All the different iterations and parallel TSs use the same TS parameter but a different starting solution.

2) MAINTAIN COLOR

This solving method, as the name implies starts by applying a certain color over the sequence of cars and attempts to apply it until this is no longer possible. When a car in the sequence is encountered that no longer requires the current color, the color is switched. The new color is determined as the color that still needs to be applied the most over all the cars in the sequence and can be applied to the current car. If ties are encountered, the color with the lowest index is selected. All the possible colors are used once as starting color and the best encountered solution is returned. This heuristic is a natural multi-color extension of the black-first heuristic used by Yarkoni et al. when solving the MCPSP [6]. This solving method is deterministic and has limited computational requirements. It therefore serves as a lower bound on the desired performance.

V. EXPERIMENTAL PROTOCOL

A. DATASETS

Since no public dataset is available for the MCPSP and MCMCPSP, a new problem generator was created. Given the N number of cars, M unique models and C unique base colors, the problem generator creates a new MCPSP/MCMCPSP instance. The generator first splits the total number of cars N into M different non-empty model groups. The cars per model then get randomly assigned a base color, to obtain the color demand $d_{k,l}$ per model k and base color l. It is possible that certain values of $d_{k,l}$ are zero. The base colors get split into two non-empty paint groups, requiring the white and black filler paint respectively. Finally, a random car sequence is generated containing the required cars per model.

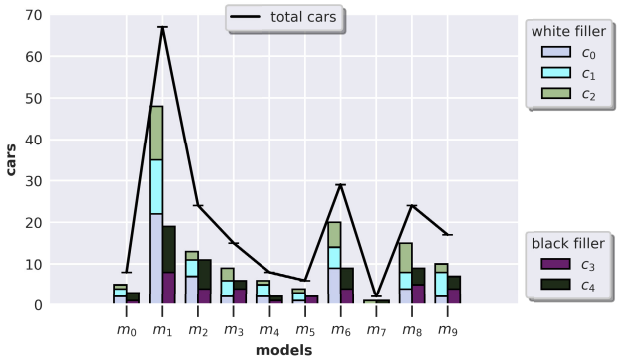

FIGURE 3. Example of problem instance containing 200 cars (N), 10 car models (M) and 5 base colors (C).

When generating the data, a check is performed to ensure a minimal amount of flexibility within the problem. It is possible that all cars of a certain model need to be painted with the same filler and/or base color and thus can not be optimized within the respective problem. With this in mind, only instances are generated for which the number of these fixed cars remains limited to 30%. The decision of 30% is in line with the data filtering performed by Yarkoni et al. on the industrial data used in [6]. An visualization of the data contained within a single problem instance can be seen in Fig. 3.

For this paper, data were generated for problems of 10 cars up to 1500 cars. The decision was made to scale the number of car models proportional to the number of cars in the problem as $M = max(N \cdot 0.05, 3)$. In terms of colors, the analysis has been limited to problems containing either 5, 10 or 15 colors. Since the MCPSP neglects the base paint color information, its complexity is independent of the number of base colors C used to generate the data. Therefore, only the generated problem instances containing 5 colors were used to analyze performance for the MCPSP. The number of cars per problem were chosen to be equal to 10, 50, 100, 200, 300, 500, 800, 1000, 1200, and 1500 for the MCPSP. For the MCMCPSP, the complexity is highly dependent on the number of base colors in the problem and can quickly become unmanageable to solve in limited time when the number of cars is too large. Therefore, the cars per problem were chosen to be equal to 10, 30, 50, 80, 100, 120, and 150 cars.

B. BENCHMARKING

Parameter tuning for the SS-based QUBO solving methods and the RPMTS has been performed in good faith to give good performance, but is not guaranteed to be optimal. The interested reader is referred to [12], for a detailed list of the parameters used within the SS. The different SS-based QUBO solving methods all use the same parameter values to be able to isolate the influence of the different solution generation and improvement methods, except for the tabu tenure. The tabu tenure of the different TS methods is scaled with the problem size by placing it equal to 10% of the number of unique moves being considered in every TS iteration. For the RPMTS, the number of parallel TS searches has been set

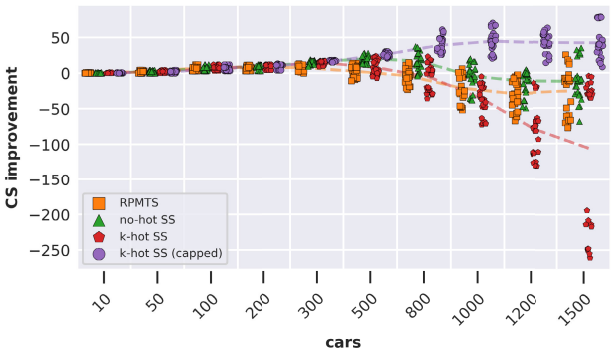

FIGURE 4. Performance of solving methods in color switch (CS) improvement over Maintain color heuristic baseline for MCPSP. Only valid solutions are shown.

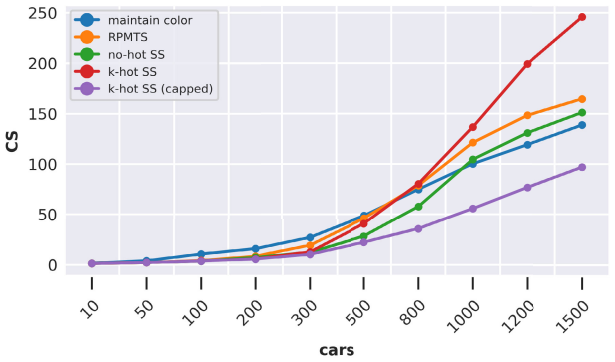

FIGURE 5. Average color switches (CS) in the solutions of different solving methods for MCPSP. Only valid solutions are included in the average.

equal to the *RefSet* size in the SS-based methods and the tabu tenure is set as 10% of the number of cars in the problem. The penalty coefficient values used in the QUBO formulations and used to penalize infeasible solutions in the RPMTS have all been set equal to the Maintain color heuristic solution value for the problem instances.

First the filler paint nozzle optimization will be discussed in Section VI-A. Similar to the procedure used by Yarkoni et al. in [6], the timing for every seeded run is scaled proportionally to the number of cars as $\frac{N}{3}$ seconds. The base paint nozzle optimization is discussed in VI-B. As the complexity increases with the number of colors C, the timing per seeded run is determined as $\frac{N \cdot C}{3}$ seconds. For every combination of number of cars N and number of colors C, five instances are solved with five seeded runs each.

All of the SS-based QUBO solving methods as well as the RPMTS have been implemented using the Rust Programming Language. the Rust-based source code was then developed as a Python module using the *Maturin* crate. Python could then be used to perform the benchmarking and analysis of the experiment results. The experiments were performed on Ubuntu 20.04.4 LTS with an Intel Core i7-9700K CPU, using Python version 3.7.12 and Rust version 1.66.1.

VI. RESULTS AND DISCUSSION

A. MCPSP RESULTS

Of the QUBO solving methods mention in Section IV-A, two can be used to solve the MCPSP. These are no-hot SS and k-hot SS. When using the latter, it is possible to exploit the k-hot constraints enforced by the color demand on the car models in (5), resulting in M distinct groups of binary variables. Results are generated when all possible moves are being considered in every TS iteration, as well as when only a random subset is being considered in the capped version. The random subset size has been chosen to be equal to the size of the 1-bit flip neighborhood. This means that the same number of moves are being considered in every TS iteration as when using no-hot SS.

Fig. 4 and Fig. 5 show that for small problems up to 200 cars, all solving methods perform equally well compared to the heuristic baseline. For the very small problems, only limited improvement is possible. When looking at the results for RPMTS and no-hot SS, which both do not exploit problem structure, it can be seen that no-hot SS is able to outperform RPMTS for all problems. These results confirm that the binary MCPSP is well suited to be solved as QUBO. For problems between 300 cars and 800 cars, a sizeable number of runs for both methods is able to improve over the Maintain color heuristic. However for problems containing a 1000 cars or more, the average performance degrades below the Maintain color heuristic. Performance can be improved when k-hot constraints within the problem are being exploited. One must however be careful when doing so, as exploiting k-hot leads to many possible moves in the TS iterations of the solution improvement step. Evaluating all of them, considerably increases the computational requirements. This explains the degraded performance of the uncapped k-hot SS. When the number of moves in every TS iteration is capped, noticeable improvements over the Maintain color heuristic are obtainable for problems up to 1500 cars. For example, for problem instances containing 1000 cars, the solutions obtained with the capped version of k-hot SS only contain 56 color switches on average compared to 100 color switches when using the heuristic.

B. MCMCPSP RESULTS

For the MCMCPSP, four QUBO solving methods can be used. These are no-hot, 1-hot, k-hot and 1-hot/k-hot SS. The 1-hot variant allows the exploitation of the 1-hot color assignment constraints per car in the sequence given by (9), resulting in N distinct groups of binary variables. The k-hot variant allows the exploitation of the k-hot color demand constraints per model and color given by (8), resulting in $M \cdot C$ distinct groups of binary variables. The 1-hot/k-hot variant allows for the exploitation of both constraint types at the same time. This results in M distinct groups of binary variables, one per car model. Every group is shaped in a matrix which contains as many rows as there are cars of that corresponding car model in the sequence, and which contains as many columns as there are colors in the problem. The k-hot constraints are applied column-wise.

Similar to the MCPSP, results are produced both when considering all moves in every TS iteration and when considering

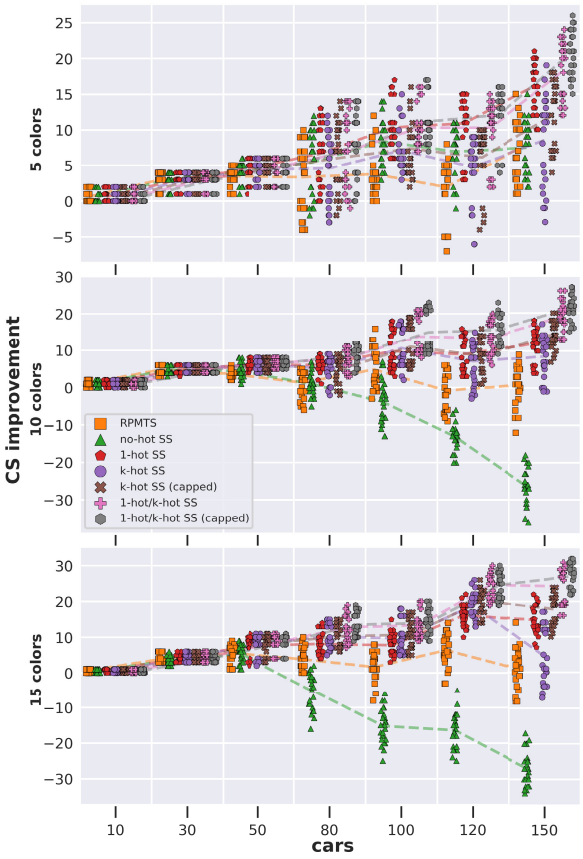

FIGURE 6. Performance of solving methods in color switch (CS) improvement over Maintain color heuristic baseline for MCMCPSP. Only valid solutions are shown.

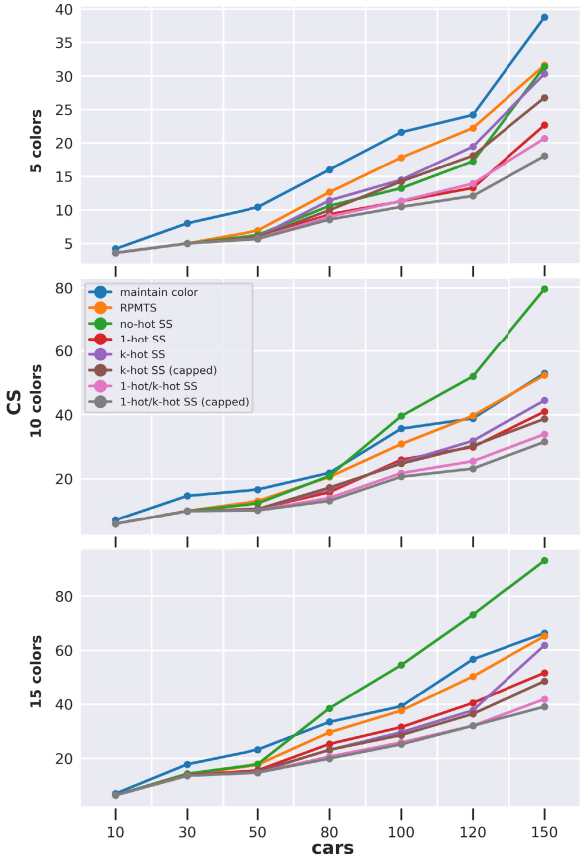

FIGURE 7. Average color switches (CS) in the solutions of different solving methods for MCMCPSP. Only valid solutions are included in the average.

only a random subset of moves equal to the size of the 1-bit flip neighborhood in the capped version. For the 1-hot SS variant, the number of moves in every TS iteration is always less than the 1-bit flip neighborhood so there is no need to cap the number of moves.

Similar to the MCPSP, Fig. 6 and Fig. 7 show that for very small problems, only limited improvement is possible over the heuristic baseline. For both 5, 10, and 15 colors, all solving methods show similar performance for problems up to 50 cars. Compared to MCPSP, no-hot SS is no longer able to consistently outperform RPMTS and for 10 and 15 colors shows considerably worse performance. This is to be expected, given the non-native binary nature of the MCMCPSP, resulting in large problem sizes when solved as QUBO. By exploiting the 1-hot constraints, which do not considerably increase the number of moves in every TS iteration, it is again possible to consistently outperform RPMTS. Further improvements can be obtained by exploiting the k-hot constraints and by exploiting both 1-hot and k-hot constraints. The additional improvements are more noticeable in more complex problems containing more colors. For problems containing 5 colors, exploiting k-hot is not able to improve over exploiting 1-hot constraints, even when capping the number of moves per TS iteration. However, capped versions do always outperform their uncapped counterparts both in exploiting k-hot constraints and in exploiting both 1-hot and k-hot constraints. By using the capped version of 1-hot/k-hot SS, solutions contained on average only 18, 32, and 39 color switches for problems containing 150 cars and 5, 10, and 15 colors respectively. Compared to this, the Maintain color heuristic results in 39, 53, and 66 color switches on average respectively.

VII. CONCLUSION AND FURTHER WORK

The research within this paper shows the viability of using QUBO solvers to solve two novel industry-inspired problems encountered within the automotive paint shop. Both a native (MCPSP) and non-native binary problem (MCMCPSP) are modelled and solved as QUBO. Methods to exploit k-hot constrained structures within the problem formulations are investigated. The results for the native binary MCPSP showed that exploitation of k-hot constraints when solving the problem as QUBO, is not required to be able to outperform the heuristic and RPMTS for problems up to 800 cars. Even better results can however be obtained when the k-hot constrained are exploited, but care should be given the number of multi-bit flip moves that are being considered during every TS iteration. The results of the non-native binary MCMCPSP showed that without exploiting k-hot constraints when solving the problem as QUBO, it is not possible to consistently outperform the heuristic and RPMTS. At a minimum,

1-hot constraints need to be exploited to bring the solving capabilities to a useful level. By exploiting k-hot constraints performance can be further improved, but again care should be given to the number of moves that are being considered in ever TS iteration to limited computational requirements.

REFERENCES

[1] T. Epping, W. Hochstättler, and P. Oertel, "Complexity results on a paint shop problem," *Discrete Appl. Math.*, vol. 136, no. 2, pp. 217–226, 2004.

[2] P. Bonsma, T. Epping, and W. Hochstättler, "Complexity results on restricted instances of a paint shop problem for words," *Discrete Appl. Math.*, vol. 154, no. 9, pp. 1335–1343, 2006.

[3] T. Epping, W. Hochstättler, and M. E. Lübbecke, "MaxFlow-MinCut duality for a paint shop problem," in *Operations Research Proceedings*, U. Leopold-Wildburger, F. Rendl, and G. Wäscher, Eds. Berlin, Germany: Springer, 2003, pp. 377–382.

[4] S. D. Andres, "Greedy versus recursive greedy: Uncorrelated heuristics for the binary paint shop problem," *Discrete Appl. Math.*, vol. 303, pp. 4–7, Nov. 2021.

[5] 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," *Phys. Rev. A, Gen. Phys.*, vol. 104, no. 1, Jul. 2021, Art. no. 012403.

[6] S. Yarkoni, A. Alekseyenko, M. Streif, D. Von Dollen, F. Neukart, and T. Bäck, "Multi-car paint shop optimization with quantum annealing," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Los Alamitos, CA, USA, Oct. 2021, pp. 35–41.

[7] G. A. Kochenberger, F. Glover, B. Alidaee, and C. Rego, "A unified modeling and solution framework for combinatorial optimization problems," *OR Spectr.*, vol. 26, no. 2, pp. 237–250, Mar. 2004.

[8] A. Lucas, "Ising formulations of many NP problems," *Frontiers Phys.*, vol. 2, p. 5, Jun. 2014.

[9] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using QUBO models," 2019, *arXiv:1811.11538*.

[10] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, and P. Bunyk, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, May 2011.

[11] H. Nakayama, J. Koyama, N. Yoneoka, and T. Miyazawa, "Third generation digital annealer technology," Fujitsu Ltd., Kawasaki, Japan, Fujitsu Ltd., Kawasaki, Japan. [Online]. Available: https://www.fujitsu.com/global/documents/about/research/techintro/3rd-g-da_en.pdf

[12] J. Pauckert, P. Debevere, M. Parizy, and M. Ayodele, "Comparing solution combination techniques in scatter search for quadratic unconstrained binary optimization," in *Proc. GECCO Companion*. New York, NY, USA: Association for Computing Machinery, 2023, pp. 2241–2249.

[13] Y. Wang, Z. Lü, F. Glover, and J.-K. Hao, "Path relinking for unconstrained binary quadratic programming," *Eur. J. Oper. Res.*, vol. 223, no. 3, pp. 595–604, Dec. 2012.

[14] T. James, C. Rego, and F. Glover, "Multistart Tabu search and diversification strategies for the quadratic assignment problem," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 39, no. 3, pp. 579–596, May 2009.

[15] F. Glover, "Heuristics for integer programming using surrogate constraints," *Decis. Sci.*, vol. 8, no. 1, pp. 156–166, Jan. 1977.

[16] X.-S. Yang, "Chapter 6—Genetic algorithms," in *Nature-Inspired Optimization Algorithms*, 2nd ed. Cambridge, MA, USA: Academic Press, 2021, pp. 91–100.

[17] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[18] F. Glover and M. Laguna, *Tabu Search*. Boston, MA, USA: Springer, 1998, pp. 2093–2229.

[19] M. Samorani, Y. Wang, Y. Wang, Z. Lv, and F. Glover, "Clustering-driven evolutionary algorithms: An application of path relinking to the quadratic unconstrained binary optimization problem," *J. Heuristics*, vol. 25, nos. 4–5, pp. 629–642, Oct. 2019.

[20] F. Gortázar, A. Duarte, M. Laguna, and R. Martí, "Black box scatter search for general classes of binary optimization problems," *Comput. Operations Res.*, vol. 37, no. 11, pp. 1977–1986, Nov. 2010.

[21] M. Kalra, S. Tyagi, V. Kumar, M. Kaur, W. K. Mashwani, H. Shah, and K. Shah, "A comprehensive review on scatter search: Techniques, applications, and challenges," *Math. Problems Eng.*, vol. 2021, pp. 1–21, May 2021.

[22] A. Verma and M. Lewis, "Penalty and partitioning techniques to improve performance of QUBO solvers," *Discrete Optim.*, vol. 44, May 2022, Art. no. 100594.

[23] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, "Physics-inspired optimization for quadratic unconstrained problems using a digital annealer," *Frontiers Phys.*, vol. 7, p. 48, Apr. 2019.

[24] S. González-Bermejo, G. Alonso-Linaje, and P. Atchade-Adelomou, "GPS: A new TSP formulation for its generalizations type QUBO," *Mathematics*, vol. 10, no. 3, p. 416, 2022.

[25] S. T. Goh, S. Gopalakrishnan, J. Bo, and H. C. Lau, "A hybrid framework using a QUBO solver for permutation-based combinatorial optimization," 2021, *arXiv:2009.12767*.

[26] S. T. Goh, J. Bo, S. Gopalakrishnan, and H. C. Lau, "Techniques to enhance a QUBO solver for permutation-based combinatorial optimization," in *Proc. Genetic Evol. Comput. Conf. Companion*. New York, NY, USA: Association for Computing Machinery, Jul. 2022, pp. 2223–2231.

[27] S. Okada, M. Ohzeki, and S. Taguchi, "Efficient partition of integer optimization problems with one-hot encoding," *Sci. Rep.*, vol. 9, no. 1, p. 13036, Sep. 2019.

[28] M. Ayodele, "Penalty weights in QUBO formulations: Permutation problems," in *Evolutionary Computation in Combinatorial Optimization*. Madrid, Spain: Springer, 2022, pp. 159–174.

[29] F. Glover and J.-K. Hao, "Efficient evaluations for solving large 0–1 unconstrained quadratic optimisation problems," *Int. J. Metaheuristics*, vol. 1, no. 1, pp. 3–10, 2010.

PIETER DEBEVERE received the B.S. degree in electrical engineering and the M.S. degree in industrial engineering and operations research from Ghent University, Ghent, Belgium, in 2020 and 2022, respectively.

In 2022, he joined the company Fujitsu Ltd., as a Research Intern, under the Vulcanus in Japan Internship Program, organized by the EU-Japan Centre for Industrial Cooperation. His research interests include combinatorial optimization, machine learning, and operations research.

MASAHIKO SUGIMURA received the B.S. degree in applied mathematics and physics and the M.S. degree in applied system science from Kyoto University, Kyoto, Japan, in 1993 and 1995, respectively.

He joined Fujitsu Ltd., Kawasaki, Japan, in 1995, where he is currently a Principal Researcher and is engaged in research areas of image recognition, machine learning, and usability engineering. From 2006 to 2010, he was a User Experience Designer and a Consultant with Fujitsu Design Ltd. Since 2010, he has been engaged in research and development on image retrieval and combinatorial optimization with Fujitsu Consulting (Canada) Inc., and Fujitsu Ltd.

MATTHIEU PARIZY received the B.Eng. and M.Eng. degrees in computer science from ESIEE Paris, France, in 2006 and 2008, respectively, and the Ph.D. degree in computer engineering from Waseda University, in 2023.

He joined the company Fujitsu Ltd., Kawasaki, Japan, in 2008, where he is currently a Research Director. His research interests include combinatorial optimization, machine learning, and VLSI design.

Dr. Parizy was a recipient of the IEEE International Conference on Consumer Electronics Best Paper Award, in 2023.