TOP
https://doi.org/10.1007/s11750-021-00610-4

ORIGINAL PAPER

A simple and effective algorithm for the maximum happy vertices problem

Marco Ghirardi[1] · **Fabio Salassa**[1]

Received: 6 April 2020 / Accepted: 31 May 2021

Abstract
In a recent paper, a solution approach to the *Maximum Happy Vertices Problem* has been proposed. The approach is based on a constructive heuristic improved by a matheuristic local search phase. We propose a new procedure able to outperform the previous solution algorithm both in terms of solution quality and computational time. Our approach is based on simple ingredients implying as starting solution generator an approximation algorithm and as an improving phase a new matheuristic local search. The procedure is then extended to a multi-start configuration, able to further improve the solution quality at the cost of an acceptable increase in computational time.

Keywords Happy coloring · Matheuristics · Local search

Mathematics Subject Classification 90C27 Combinatorial Optimization · 90C11 Mixed Integer Programming · 90C59 Approximation methods and heuristics in mathematical programming

1 Introduction

Vertex coloring problems are one of the most popular and extensively studied subjects in the field of graph theory. They have received wide attention in the literature, not only for their real-world applications but also for their theoretical aspects and for the computational hardness (Malaguti and Toth 2010). Traditional vertex coloring problems consist of coloring all vertices of a graph G with different colors in such a way that any pair of adjacent vertices are labeled with different colors. Recently, interest has been also devoted to vertex coloring problems where the coloring of adjacent vertices is desired to be the same. This is the case of the problem called *Maximum Happy Vertices Problem* (MHV) considered in this paper. Given a set of

✉ Marco Ghirardi
marco.ghirardi@polito.it

[1] DIGEP, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Turin, Italy

Published online: 09 June 2021

precolored vertices, the problem asks to extend the coloring to the remaining vertices with the objective to maximize the number of nodes colored with the same color of their adjacent vertices.

The MHV problem and the concept of "happiness" related to vertices have been proposed in Zhang and Li (2015). A vertex is considered *happy* if all its neighbors are of the same color. The problem objective is the maximization of the number of happy vertices.

More formally, the MHV problem considers an undirected graph $G = (V, E)$ with n vertices and m edges (with $\Gamma(i)$ defined as the set of neighbors of vertex i), a color set $K = \{1, \ldots, k\}$, a subset of vertices $A \subseteq V$ where $|A| \geq k$ and a partial coloring $c : A \rightarrow \{1, \ldots, k\}$ such that $\forall\, i \in \{1, \ldots, k\}, \exists\, v \in A : c(v) = i$. The problem asks to extend the coloring c to the remaining non-precolored vertices to a complete graph coloring $\bar{c} : V \rightarrow \{1, \ldots, k\}$ such that the total number of happy vertices is maximized.

In a recent paper Lewis et al. (2019), the MHV problem has been addressed and a solution approach based on the *Construct, Merge, Solve & Adapt* (*CMSA*) framework of Blum et al. (2016) has been applied to deal with 380 computationally hard instances.

The problem has also been tackled from a theoretical point of view, see the proof of NP-hardness in Zhang and Li (2015), approximation algorithms in Zhang et al. (2018) and complexity results in Agrawal (2017) and Aravind et al. (2016), where polynomial algorithms for simple special cases have been proposed.

From a computational perspective, to the best of our knowledge, the work of Lewis et al. (2019) is the first attempt to propose solution procedures dealing with large-size instances. Moreover, the authors of Lewis et al. (2019) made freely available both the instance generator Lewis et al. (2018a, b) and the source code (except the part related to the mixed integer linear programming solver GUROBI) Lewis et al. (2019). The work of Lewis et al. (2019) proposes a hybrid heuristic approach, based on a constructive heuristic improved by a matheuristic local search phase.

Matheuristics are solution methods that have been successfully applied to several combinatorial optimization problems [see for instance Ball (2011), Della Croce et al. (2013)], giving rise to an impressive amount of research in recent years. Matheuristics have been applied to routing Macrina et al. (2019) Shahmanzari et al. (2020), packing Billaut et al. (2015), Martinez-Sykora et al. (2017), rostering Della Croce and Salassa (2014), Doi et al. (2018), lot sizing Ghirardi and Amerio (2019) and machine scheduling Della Croce et al. (2014), Croce et al. (2019), Fanjul-Peyro et al. (2017) just to cite a few of them. Matheuristics rely on the general idea of exploiting the strength of both metaheuristic algorithms and exact methods.

In the present work, we developed a simple but effective matheuristic algorithm, along the same line of *CMSA* in Lewis et al. (2019), to deal with the Maximum Happy Vertices Problem. The proposed matheuristic algorithm is based on an overarching neighborhood search approach with an intensification search phase realized by a MILP solver. The main advantages of our approach, with respect to the one of Lewis et al. (2019), are:

– Better performances in terms of solution quality,

– Much better performances in terms of computational times (few seconds against 1 h),
– Simple design of the solution procedure,
– Simple integration in a multi-start version able to further improve the solutions quality.

The paper is organized as follows. In Sect. 2 , the integer linear programming formulations of the problem are provided. Section 3 is devoted to the description of the proposed solution algorithms. In Sect. 4, computational results and benchmarks are presented. Sect. 5 concludes the paper with final remarks.

2 MIP models

Two mixed integer linear programming formulations are provided in Lewis et al. (2019).

In the *first model* (M1), integer variables $x_i \in \{1, \ldots, k\}$ define the color assigned to each vertex, while variables y_i are forced to be one only if vertex i is unhappy. The set A represents the set of precolored vertices while $c(i)$ is the color assigned to each vertex in A. Recall that $\Gamma(i)$ is defined as the set of neighbors of vertex i. The overall model is:

$$\max n - \sum_{i=1}^{n} y_i \tag{1}$$

$$\begin{aligned} &\text{subject to:} \\ &x_i = c(i) \qquad \forall v_i \in A \end{aligned} \tag{2}$$

$$y_i \geq \frac{|x_i - x_j|}{n} \qquad \forall i \in V, \forall j \in \Gamma(i) \tag{3}$$

$$x_i \in \{1, \ldots, k\} \qquad \forall i \in V \tag{4}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \tag{5}$$

where (1) maximizes the number of happy vertices, (2) assigns colors to all the precolored vertices, (3) sets $y_i = 1$ for unappy vertices. (4) and (5) define the optimization variables.

Note that constraints (3) are not linear, and hence they require a linearization [not explicitated in Lewis et al. (2019)], which results in their substitution with:

$$y_i \geq \frac{x_i - x_j}{n} \qquad \forall i \in V, \forall j \in \Gamma(i) \tag{6}$$

$$y_i \geq \frac{x_j - x_i}{n} \qquad \forall i \in V, \forall j \in \Gamma(i) \tag{7}$$

The *second model* (M2) uses binary variables x_{ij} where $x_{ij} = 1$ if and only if color j is assigned to vertex i. Variables y_i have the same meaning as in the first model.

$$\max n - \sum_{i=1}^{n} y_i \tag{8}$$

$$\begin{aligned} &\text{subject to:} \\ &x_{ij} = 1 \qquad \forall i \in A : c(i) = j \end{aligned} \tag{9}$$

$$\sum_{j=1}^{k} x_{ij} = 1 \qquad \forall i \in V \tag{10}$$

$$y_i \geq |x_{ij} - x_{lj}| \qquad \forall i \in V, \forall l \in \Gamma(i), \forall j \in K \tag{11}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V, \forall j \in K \tag{12}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \tag{13}$$

Here, (8) maximizes the number of happy vertices, constraints (9) specifies the precolorings, constraints (10) ensures that one color is assigned to any vertex, and constraints (11) forces $y_i = 1$ if vertex i is unhappy. (12) and (13) define the optimization variables.

As before, constraints (11) are not linear, and we propose the following linearization:

$$y_i \geq x_{ij} - x_{lj} \qquad \forall i \in V, \forall l \in \Gamma(i), \forall j \in K \tag{14}$$

$$y_i \geq x_{lj} - x_{ij} \qquad \forall i \in V, \forall l \in \Gamma(i), \forall j \in K \tag{15}$$

We also point out that y_i variables, for this second model, not necessarily need to be defined as binary. It is, in fact, sufficient to define them as $0 \leq y_i \leq 1$ given that they are constrained by (14) and (15) which enforce y_i to be 0 or 1.

Despite the fact that in Lewis et al. (2019), it is reported that model M2 is far less reliable w.r.t. solution quality, we tested it against all instances using CPLEX 12.7 as MIP solver and found out that 80 instances out of the whole dataset made of 380 instances were solved to optimality. All instances with 250, 500, 750 and 1000 nodes and $k = 10$ had been solved to optimality within the time limit of 3600 s, the same time limit used in Lewis et al. (2019). Thus, in our experiments, model M2 outperforms model M1. From now on, in our algorithms, we use model M2 for benchmarks since, overall, it gives better solutions within the same time limit with respect to the model M1.

3 A simple solution approach

We propose here a simple but effective matheuristic improvement approach. Starting from a given solution, the algorithm iteratively improves it with a scheme based on the neighborhood search approach. Each iteration explores the neighborhood by constructing a problem where the variables to be optimized refer to a subset of the variables of the original problem, while other ones are fixed to the value they have in the current solution. The detailed procedure is described in Algorithm 1. The algorithm starts with a given feasible solution $\bar{c}$ (step 1). A counter *no_improvement* of iterations passed without finding an improving solution is set to 0 (step 2). At each iteration of the main loop (cycle 3–16), a subset of candidate colors for each node is selected and an exact method is employed to build a possibly improved solution. In the current solution (steps $4--8$), candidate colors for each node i are the current color and all the colors assigned to nodes adjacent to i with a path of length L, i.e. nodes colors that can be recognized following an $L-length$ edge path. The resulting problem is then optimally solved through model (8)–(13), obtaining solution $\bar{c}'$ (step 9). If the new solution is better than the previous one, the counter of non-improving iterations is reset to 0 (step 11). Otherwise, it is increased by one (step 13). Note that solution $\bar{c}'$ cannot be worse than the current solution $\bar{c}$ because the latter is a feasible solution of the ILP model. Hence, it is always accepted as the new current solution (step 15). The improvement phase is repeated if less than S iterations have been performed without improving the current solution.

CMSA solution approach proposed in Lewis et al. (2019) is based on a similar improvement scheme as the one we propose, with a different neighborhood definition. We highlight here the main differences:

- In *CMSA*, candidate colors for each node i to be chosen for reoptimization are only the color of i plus, with a given probability, a subset of the colors of the neighbor nodes of i, while our matheuristic procedure considers, as possible candidates for each node i, all colors of the nodes that could be reached from node i with a path of a given length. Hence, the neighborhood dimension of the proposed algorithm is larger than the one of *CMSA*.
- *CMSA* uses model M1, while in our case, model M2 has been selected. This choice does not affect the algorithm results in terms of solution quality (all ILPs are solved to optimality) but influences the running time.

Algorithm 1 Matheuristic Improvement Procedure

INPUT: an initial solution $\bar{c}$ ($\bar{c}(i)$ = color assigned to vertex i)
$no\_improvement = 0$
while $no\_improvement < S$ **do**
  for $i = 1, \ldots, n$ **do**
    Define $N(i) = \emptyset$ as the set containing the candidate colors for node i.
    Add to $N(i)$ the colors $\bar{c}(l)$ of the nodes l in solution $\bar{c}$ such that $distance(i, l) \leq L$.
    Add to model M2 the constraints $x_{ij} = 0 \qquad \forall j \notin N(i)$.
  end for
  Optimally solve the resulting ILP model, obtaining solution $\bar{c}'$
  if $ObjFunction(\bar{c}') > ObjFunction(\bar{c})$ **then**
    $no\_improvement = 0$
  else
    $no\_improvement++$
  end if
  $\bar{c} = \bar{c}'$
end while
OUTPUT: solution $\bar{c}$

In Lewis et al. (2019), two constructive methods are proposed for the initial solution generation, namely $Greedy - MHV$ and $Growth - MHV$. We point out that $Greedy - MHV$ is the same procedure as the approximation algorithm $\mathcal{G}$ proposed in Zhang et al. (2018). The approximation algorithm $\mathcal{G}$ has been used in our approach as starting solution. During preliminary testings, we tried the best among algorithm $\mathcal{G}$ (a.k.a. $Greedy - MHV$) and $Growth - MHV$, but we experienced no improvements in the solutions quality.

The rationale of algorithm $\mathcal{G}$ of Zhang et al. (2018) is to label all the uncolored vertices with the same color and testing all possible k colors obtaining, in such way, k different vertices colorings. The starting solution is then chosen among all k colorings, i.e. the one exhibiting the largest number of happy vertices (Algorithm 2).

Algorithm 2 Initial solution: algorithm $\mathcal{G}$

$BestSol = \emptyset; BestVal = 0$
for $j = 1, \ldots, k$ **do**
  Build solution $\bar{c} : \bar{c}(i) = j \qquad \forall i \notin A$
  if $ObjFunction(\bar{c}) > BestVal$ **then**
    $BestVal = ObjFunction(\bar{c})$
    $BestSol = \bar{c}$
  end if
end for
OUTPUT: solution $\bar{c}$

To further test the matheuristic improvement algorithm in order to assess the quality of the proposed approach, we tested it in a multi-start setting. In this new configuration, the improvement procedure depicted in Algorithm 1 is applied not only on the solution obtained by the best coloring of algorithm $\mathcal{G}$, but on all possible k different vertices colorings. The best final result is then returned. Algorithm 3 resumes the main steps of the multi-start procedure.

Algorithm 3 Multi-start Algorithm

```
BestSol = ∅; BestVal = 0
for j = 1, ..., k do
    Build solution c̄ : c̄(i) = j        ∀i ∉ A
    Apply to c̄ the improvement procedure of algorithm 1.
    if ObjFunction(c̄) > BestVal then
        BestVal = ObjFunction(c̄)
        BestSol = c̄
    end if
end for
OUTPUT: solution BestSol
```

4 Computational experiments

We decided to test different configurations of algorithms over a dataset generated thanks to the instance generator in Lewis et al. (2018a, b). According to Lewis et al. (2019), instances were generated as random graphs using values of $p = 5/(n-1)$, where n is the total number of nodes and p the probability of two vertices being adjacent. This value of p induces an average vertex degree of 5. In all instances, 10% of the vertices were precolored. Authors of Lewis et al. (2019) state that these configurations lead to the creation of the most difficult-to-solve instances. As in Lewis et al. (2019), we considered classes of instances with a number of colors k equal to 10 or 50 on graphs having a number of nodes n of 500, 750, 1000, 2000, 3000, 4000, 5000, 7500 and 10000. Since the solver is able to solve to optimality four classes (namely all the ones with $k = 10$ and with n equal to 250, 500, 750 and 1000), these were not considered in our dataset. For each of the remaining 15 classes, 20 instances were generated. For tuning the algorithm parameters, we generated an additional smaller dataset, composed of 10 instances for each of the classes with k equal to 10 or 50 and n equal to 3000, 5000 and 10000.

The algorithms have been implemented in C++ and the source code is available upon request to the authors. All tests have been performed on an i5-8500 3 GHz CPU system with 16 GB of RAM and CPLEX 12.7 as MIP solver. CPLEX solver has been applied with no parameters tuning and in multi-threaded mode.

The following two subsections present the results of the experiments aiming to tune the algorithm parameters, and the comparison between the results of the proposed algorithms and *CMSA*, proposed in Lewis et al. (2019).

4.1 Parameter tuning

In order to tune the values of parameters L and S of Algorithm 1, a set of computational experiments has been performed.

Table 1 summarizes the results. For each class of instances (k colors and n nodes), we present the average percentage of happy nodes $H\%$ and computational

Table 1 Improvement procedure algorithm parameter tuning

L		1						2					
S		1		2		3		1		2		3	
k	n	H%	T	H%	T	H%	T	H%	T	H%	T	H%	T
10	3000	59.225	1.1	59.225	1.9	59.225	2.5	**59.244**	3.0	**59.244**	4.2	**59.244**	6.4
10	5000	59.122	2.0	59.122	3.1	59.122	4.7	**59.131**	5.4	**59.131**	9.2	**59.131**	13.1
10	10000	57.981	6.4	57.982	9.1	57.985	12.9	57.992	12.8	**57.994**	20.0	**57.994**	31.4
50	3000	56.141	5.1	56.141	8.2	56.141	11.4	56.249	7.1	**56.250**	10.9	**56.250**	13.9
50	5000	56.101	11.4	56.101	13.1	56.101	17.7	**56.132**	13.1	**56.132**	18.5	**56.132**	27.9
50	10000	54.991	26.1	54.991	35.1	54.995	39.5	54.999	29.0	55.021	39.8	**55.023**	61.4
AVG		57.260	8.7	57.260	11.8	57.262	14.8	57.291	11.7	57.295	17.1	**57.296**	25.7

The best entries for each line are highlighted in bold

time T over the 10 tuning instances, with different parameters values L and S. The best entries for each line are highlighted in bold.

Parameter L defines the neighborhood size, and has been considered equal to 1 or 2. Setting a value of 3 or more will result in the creation of ILP models with too many free variables, sometimes exceeding a time limit of 3600 s without finding the optimal solution. It is clear from the table that the best choice is $L = 2$, having better results at cost of an acceptable increase of computational time.

Paramenter S configures the algorithm stopping criterion and ranges from 1 to 3. While an improvement is clear in results obtained increasing S from 1 to 2, the results are, for most instances, the same when $S = 3$. Hence, we decided to set $S = 2$.

4.2 Algorithms results comparison

As previously pointed out, authors of Lewis et al. (2019) made freely available the source code of their algorithms except the part related to the mixed integer linear programming model and solver. Then, in order to benchmark our procedure with the reference algorithm *CMSA*, we re-implemented it, integrating their source code with a mixed integer linear programming model. In the description of *CMSA*, it is not clear how single-color labels could be efficiently excluded from the list of possible colors since the variables used are of integer type (model M1 is used) and no constraints sets (i.e. disjunctive constraints) have been explicitated to deal with values exclusion. Hence, we contacted the authors of Lewis et al. (2019) asking for details on *CMSA* implementation which is slightly different with respect to the published paper. Thanks to their help we reconstructed *CMSA* as originally implemented. Each time the LP model M1 is run, the following rules are used:

- If a node $\bar{i}$ has only one candidate color $\bar{c}$, the corresponding variable is set to that color ($x_{\bar{i}} = \bar{c}$).
- If a node $\bar{i}$ has more candidate colors, the corresponding variable is left free to get any value ($x_{\bar{i}} \in \{1, ..., k\}$).

For other details about *CMSA* refer to Lewis et al. (2019). On the other side, excluding values implying model M2 as in our matheuristic is rather simple: it is, in fact, sufficient to add constraints like $x_{\bar{i}\bar{j}} = 0$ if we want to prevent node $\bar{i}$ to be labeled with color $\bar{j}$.

We tested the following approaches:

- CPLEX: Lower Bound and Upper Bound after 3600 s calculated by CPLEX solver with model M1.
- *CMSA*: original *CMSA* using as starting solution the best among $Greedy - MHV$ and $Growth - MHV$ with a time limit of 3600 s [as in Lewis et al. (2019)]. Considering that *CMSA* is not a deterministic algorithm, we present here the best result obtained with 10 different executions.
- MH-G: matheuristic algorithm 1, configured with $L = 2$ and $S = 2$, using as starting solution the approximation Algorithm $\mathcal{G}$.

– MS: multi-start version of the procedure, depicted in algorithm 3.

Table 2 summarizes the results. The meaning of the columns of Table 2 is the following:

– Column 1: number of different colors k.
– Column 2: number of nodes n of the specific class of instances.
– Column 3: percentage of "happy" vertices w.r.t the total number of nodes of the *upper bound* provided by CPLEX after 3600 s of run.
– Column 4: percentage of "happy" vertices w.r.t the total number of nodes of the *lower bound* provided by CPLEX after 3600 s of run.
– Column 5: average values of the percentage of "happy" vertices given by the *CMSA* approach after 3600 s—best of 10 executions.
– Column 6: average values of the percentage of "happy" vertices given by the proposed $MH-G$ algorithm (bold characters if $MH-G$ is better than *CMSA*).
– Column 7: average maximum CPU time needed to compute the result of $MH-G$ algorithm, in seconds.
– Column 8: average values of the percentage of "happy" vertices given by the *MS* configuration of the proposed algorithm (bold characters if *MS* is better than $MH-G$).
– Column 9: average maximum CPU time needed to compute the result of *MS* procedure, in seconds.

Table 2 Computational results: algorithms results comparison

k	n	UB%	LB%	CMSA%	MH-G%	MH-G(s)	MS%	MS(s)
10	2000	65.764	59.540	59.883	**59.908**	2.7	59.908	27.5
10	3000	66.161	59.020	59.232	**59.252**	4.7	**59.262**	46.2
10	4000	69.237	59.011	59.570	**59.598**	6.9	59.598	65.1
10	5000	68.939	58.450	59.024	**59.040**	9.0	**59.042**	85.6
10	7500	68.524	58.493	58.987	**59.007**	14.6	**59.009**	145.0
10	10000	68.653	57.509	57.974	**57.999**	20.3	57.999	214.9
50	500	61.560	56.060	56.080	**56.140**	1.4	**56.160**	74.9
50	750	61.856	55.587	55.640	**55.707**	2.2	55.707	117.1
50	1000	64.376	57.095	57.240	**57.255**	3.0	**57.260**	159.3
50	2000	65.277	56.090	56.535	**56.568**	6.5	56.568	355.4
50	3000	65.058	55.352	55.670	**55.703**	10.7	**55.707**	581.0
50	4000	65.856	55.994	56.446	**56.480**	14.6	**56.484**	814.4
50	5000	65.684	55.622	56.062	**56.091**	19.0	56.091	1067.8
50	7500	68.127	55.387	55.809	**55.839**	30.4	55.839	1795.3
50	10000	67.878	54.492	54.860	**54.884**	43.7	54.884	2626.5

For each line, the MH-G% value is in bold if better than CMSA% and the MS% value is in bold if better than MH-G%

As can be seen, the simple proposed approach $MH - G$ outperforms *CMSA* both in terms of solution quality and CPU effort. We recall that the stopping criterion used in Lewis et al. (2019) is the time limit of 3600 s. Our approach gives better results in about two order of magnitude less CPU time. Moreover with algorithm *MS*, we gain even more solution quality, largely within the 3600 s limit. These results illustrate the effectiveness of our approach which shows up to improve with respect to the current literature.

To further assess the effectiveness of our approaches, Table 3 is reported. Even if the averages improvements of objectives function values may seem limited, the number of improvements is definitely clear. Here, Columns 1 and 2 are the same as in Table 3, while column 3 explicits the number of instances per class. Columns 4 and 5 are dedicated to enlight the number of instances improved with respect to the *CMSA* procedure of $MH - G$ and *MS* algorithms, respectively. As can be seen, apart from one case where the number of improved instances is very limited, $MH - G$ (and consequently *MS*) approaches consistently improve over *CMSA*. It is important to note that there are no instances where *CMSA* is better than any of our approaches. Globally, we could improve 180 out of 300 instances and we point out that our approach is consistently better on the larger-size instances. This again confirms the effectiveness of the proposed approach.

5 Concluding remarks

A simple procedure has been developed to deal with the *Maximum Happy Vertices Problem*. A starting solution generation obtained thanks to an approximation algorithm is improved via a large-scale neighborhood exploration made with an MILP formulation of the problem. The procedure is then extended in a multi-start

Table 3 Number of improved instances with respect to *CMSA* algorithm

k	n	# Instances	MHGvsCMSA	MSvsCMSA
10	2000	20	7	7
10	3000	20	9	9
10	4000	20	10	10
10	5000	20	13	13
10	7500	20	16	16
10	10000	20	16	16
50	500	20	6	7
50	750	20	6	6
50	1000	20	2	3
50	2000	20	11	11
50	3000	20	15	15
50	4000	20	15	15
50	5000	20	16	16
50	7500	20	19	19
50	10000	20	19	19

configuration. Both approaches have been tested over 300 instances from the literature and compared with a reference algorithm, namely *CMSA* from Lewis et al. (2019). Solution quality and very limited running times confirm the effectiveness of our approach which is based on simple elements and shows up to improve with respect to the current literature.

Funding Open access funding provided by Politecnico di Torino within the CRUI-CARE Agreement.

References

Agrawal A (2017) On the parameterized complexity of happy vertex coloring. In IWOCA

Aravind N, Kalyanasundaram S, Anjeneya SK (2016) Linear time algorithms for happy vertex coloring problems for trees. In IWOCA

Ball M (2011) Heuristics based on mathematical programming. Surv Oper Res Manag Sci 16:21–38

Billaut J, Croce F, Grosso A (2015) A single machine scheduling problem with two-dimensional vector packing constraints. Eur J Oper Res 243:75–81

Blum C, Pinacho P, López-Ibáñez P, Lozano JA (2016) Construct, merge, solve and adapt a new general algorithm for combinatorial optimization. Comput Oper Res 68:75–88

Croce F, Grosso A, Salassa F (2019) Minimizing total completion time in the two-machine no-idle no-wait flow shop problem. J Heuristics 1:1–15

Della Croce F, Salassa F (2014) A variable neighborhood search based matheuristic for nurse rostering problems. Ann Oper Res 218:185–199

Della Croce F, Grosso A, Salassa F (2013) Matheuristics: embedding milp solvers into heuristic algorithms for combinatorial optimization problems. Heuristics: theory and applications. Nova Science Publishers, New York, pp 31–52

Della Croce F, Grosso A, Salassa F (2014) A matheuristic approach for the two-machine total completion time flow shop problem. Ann Oper Res 213:67–78

Doi Tsubasa, Nishi Tatsushi, Voß Stefan (2018) Two-level decomposition-based matheuristic for airline crew rostering problems with fair working time. Eur J Oper Res 267:428–438

Ghirardi M, Amerio A (2019) Matheuristics for the lot sizing problem with back-ordering, setup carry-overs, and non-identical machines. Comput Ind Eng 127:822–831

Giusy Macrina G, Laporte F. Guerriero, Pugliese L (2019) An energy-efficient green-vehicle routing problem with mixed vehicle fleet, partial battery recharging and time windows. Eur J Oper Res 276:971–982

Lewis R, Thiruvady D, Morgan K (2018a) Algorithm source code. http://www.rhydlewis.eu/resources/happyalgs.zip. Accessed date 7 Jun 2021

Lewis R, Thiruvady D, Morgan K (2018b) Problem instance generator. http://www.rhydlewis.eu/resources/happygen.zip. Accessed date 7 Jun 2021

Lewis R, Thiruvady D, Morgan K (2019) Finding happiness: an analysis of the maximum happy vertices problem. Comput Oper Res 103:265–276

Luis Fanjul-Peyro F, Perea R. Ruiz (2017) Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. Eur J Oper Res 260:482–493

Malaguti E, Toth P (2010) A survey on vertex coloring problems. Int Trans Oper Res 17:1–34

Martinez-Sykora A, Alvarez-Valdés R, Bennell J, Ruiz R, Tamarit JM (2017) Matheuristics for the irregular bin packing problem with free rotations. Eur J Oper Res 258:440–455

Shahmanzari Masoud, Aksen D, Salhi S (2020) Formulation and a two-phase matheuristic for the roaming salesman problem: application to election logistics. Eur J Oper Res 280:656–670

Zhang P, Li A (2015) Algorithmic aspects of homophyly of networks. Theor Comput Sci 593:117–131

Zhang P, Xu Y, Li A, Lin G (2018) Improved approximation algorithms for the maximum happy vertices and edges problems. Algorithmica 80:1412–1438