Communications in Combinatorics and Optimization
Vol. 2 No. 2, 2017 pp.149-159
DOI: 10.22049/CCO.2017.25918.1055

CCO
Commun. Comb. Optim.

Graceful labelings of the generalized Petersen graphs

Zehui Shao[1], Fei Deng[2], Zepeng Li[3], Aleksander Vesel[4]*

[1]School of Information Science and Engineering, Chengdu University, Chengdu, 610106, China
zshao@cdu.edu.cn

[2]College of Information Science and Technology, Chengdu University of Technology, Chengdu, China
dengfei@cdut.cn

[3]Key Laboratory of High Confidence Software Technologies, Peking University, Peking, China
lizepeng@pku.edu.cn

[3]Faculty of Natural Sciences and Mathematics, University of Maribor, Maribor, Slovenia
aleksander.vesel@um.si

Received: 20 March 2017; Accepted: 30 August 2017
Published Online: 4 September 2017
Communicated by ***Alireza Ghaffari-Hadigheh***

Abstract: A graceful labeling of a graph $G = (V, E)$ with m edges is an injection $f : V(G) \to \{0, 1, \ldots, m\}$ such that the resulting edge labels obtained by $|f(u) - f(v)|$ on every edge uv are pairwise distinct. For natural numbers n and k, where $n > 2k$, a generalized Petersen graph $P(n, k)$ is the graph whose vertex set is $\{u_1, u_2, \ldots, u_n\} \cup \{v_1, v_2, \ldots, v_n\}$ and its edge set is $\{u_iu_{i+1}, u_iv_i, v_iv_{i+k} : 1 \leq i \leq n\}$, where subscript arithmetic is done modulo n. We propose a backtracking algorithm with a specific static variable ordering and dynamic value ordering to find graceful labelings for generalized Petersen graphs. Experimental results show that the presented approach strongly outperforms the standard backtracking algorithm. The proposed algorithm is able to find graceful labelings for all generalized Petersen graphs $P(n, k)$ with $n \leq 75$ within only several seconds.

Keywords: graceful labeling, generalized Petersen graph, heuristic

AMS Subject classification: 05C15, 05C85

1. Introduction

Let G be a graph with n vertices and m edges. An injection $f : V(G) \to \{0, 1, \ldots, m\}$ is called a *graceful labeling* of G if the resulting edge labels obtained by $|f(u) - f(v)|$

* *Corresponding Author*

on every edge uv are pairwise distinct. If a graph G admits a graceful labeling, then we say that G is *graceful*. The *graceful labeling problem* is to determine whether a given graph is graceful or not.
Several graph labelings have been developed since Rosa [4] introduced the graceful labeling (called the β-valuation of a graph at the time). The dynamic survey of graph labelings [2] presents an overview of extensive developments in this area.
It is shown [1] that gracefully labeled graphs can be used for modeling in a wide range of applications. The most known examples are coding theory and communication network addressing.
Although an unpublished result of Erdös [2] shows that most graphs are not graceful, it is known that many graphs that have some sort of regularity of structure are graceful. The graph which are known to be graceful are for example paths, wheels, as well as families of cycles C_{4k} and C_{4k+3} (see [2] for the complete list of graphs that are know to be graceful).
For natural numbers n and k, where $n > 2k$, a generalized Petersen graph $P(n,k)$ is the graph whose vertex set is $\{u_1, u_2, \ldots, u_n\} \cup \{v_1, v_2, \ldots, v_n\}$ and its edge set is $\{u_i u_{i+1}, u_i v_i, v_i v_{i+k} : 1 \leq i \leq n\}$, where subscript arithmetic is done modulo n using the residues $0, 1, \ldots, n-1$.
Redl [3] established that the Petersen graph $P(n,k)$ is graceful with n up to 10. In this paper, we show that the generalized Petersen graph $P(n,k)$ is graceful for all $5 \leq n \leq 75$.
The paper is organized as follows. In Section 2, a backtracking algorithm with a specific static variable ordering and dynamic value ordering is introduced. The presented algorithm is the main computer search method applied in the computation, the results of which are given in Section 3. This section also includes a comparison of the introduced algorithm with a standard backtracking algorithm and with an approach based on an integer programming formulation.

2. Algorithmic search

A *constraint satisfaction problem* (CSP) consists of a set of n variables, $x_1, \ldots, x_n$, and a set of constraints. For each variable x_i a domain D_i with r elements $d_{i_1}, d_{i_2}, \ldots, d_{i_r}$ is specified; a variable can only be assigned a value from its domain. A constraint specifies a subset of the variables and which combinations of value assignments are allowed for that subset. A *solution* to a CSP is an assignment of values to all the variables such that no constraint is violated.
It is straightforward to see that the graceful labeling problem for a graph G is an example of a constraint satisfaction problem where the vertices of a given graph correspond to the set of variables of CSP such that the domain of each variable is $\{0, 1, \ldots, |E(G)|\}$.
Most algorithms for solving CSP search systematically through the possible assignment of values to variables. A well known example of a systematic search is a backtracking algorithm, which has been used for the graceful labeling problem by the

authors of this paper.
The backtrack search process is usually represented as a *search tree*, where each node represents a choise of value of a variable, and each branch represents a candidate partial solution.

2.1. Variable ordering

A backtracking algorithm needs the order in which variables are to be considered to be determined. The ordering may be *static* where the order of the variables is specified before the search begins. On the other hand, a *dynamic* ordering requires that the choice of the next variable to be considered depends on the values of the variables that have been already determined. In order to find an efficient algorithm for the the graceful labeling problem various ordering have been considered. The best results have been achieved by using the following static ordering.
Let $|V(G)| = n$ and let $\sigma = v_1, v_2, \ldots, v_n$ be a sequence of all vertices of G. We treat σ as a function $\sigma : \{1, 2, \ldots, n\} \to V(G)$ where $\sigma(i)$ denote the i-th vertex in σ and $\sigma^{-1}(u)$ gives the position of u in σ. Let V_u denote the set of vertices of G that appear earlier in σ then u i.e. $V_u = \{v \in V(G) \mid \sigma^{-1}(v) < \sigma^{-1}(u)\}$. We say that the sequence σ is *internally connected* if for every vertex $u \in V(G)$ there exists an edge $uv \in E(G)$ such that $v \in V_u$.

2.2. Value ordering

When a current variable is selected, a backtracking algorithm has to choose a value to assign. As with variable ordering, an algorithm could decide the order in which values of the domain are to be assigned. A proper value ordering could accelerate the search provided that a solution exists and that only one solution is needed. For the graceful labeling problem, we define the following dynamic value ordering.
Let $\sigma = v_1, v_2, \ldots, v_n$ be an internally connected sequence of vertices of G and let $v \in V(G)$. Let also $N(u, \sigma)$ denote the set of neighbors of u that appear earlier in σ than u. More formally, $N(u, \sigma) = \{v \in V_u \mid uv \in E(G)\}$.
Let $f : V(G) \to \{1, \ldots, |E(G)|\}$ be a function, where for $u \in V(G)$ the value $f(u)$ is obtained by the algorithm Backtracking presented in the next subsection. Then the restriction of f to V_u is denoted by f_u.
For an internally connected sequence σ of vertices of G we define a function $w : V(G) \times V(G)^{\{1,2,\ldots,n\}} \times \{0, 1, \ldots, |E(G)|\}^{V(G)} \to \mathbb{R}$ as

$$w(u, \sigma, f) = \frac{1}{|N(u, \sigma)|} \sum_{v \in N(u,\sigma)} f(v).$$

Note that v belongs to $N(u, \sigma) \subseteq V_u$ in the definition of w. Thus, f is used in fact as the restriction of f to V_u. Since the algorithm uses the variable ordering defined by σ for the current vertex u, the restriction of f to V_u is already defined when u is selected. The value of $w(u, \sigma, f)$ is therefore well defined for every step of the algorithm.

The ordering of values for a current vertex u depends on the value of $w(u, \sigma, f)$ as follows. If $w(u, \sigma, f) \geq \frac{|E(G)|}{2}$, then the ordering $0, 1, \ldots, |E(G)|$ is used. Otherwise, the algorithm uses the ordering $|E(G)|, |E(G)| - 1, \ldots, 0$.

2.3. Algorithm

We first give the outline of the algorithm.

a) Before the search, we rearrange the order of vertices of G randomly such that the resulting sequence of vertices $\sigma = v_1, v_2, \ldots, v_n$ is internally connected.

b) The label of the vertex v_1 is set to 0 at the beginning of the computation. The algorithm then labels the vertices of the sequence $v_2, \ldots, v_n$ vertex by vertex with labels from the set $\{0, 1, \ldots, |E(G)|\}$. The order of labels used for the current vertex v_i depends on the value $w(v_i, \sigma, f)$ as described in section 2.2.

c) If the procedure runs for more than a given a period of time, the procedure is restarted until a valid graceful labeling is found (see lines 13-16 of the algorithm).

The parameters of algorithm are described as follows.

- G: a graph to be tested, the vertices of G are ordered in the sequence $\sigma = v_1, v_2, \ldots, v_n$;
- m: the number of the edges of G;
- n: the number of the vertices of G;
- i: the index of the vertex in σ, it also denotes the level of the search tree of the backtracking procedure;
- c: the label to be assigned to v_i;
- T: the time bound;
- T_0: the starting time of the algorithm.

```
**Procedure** Backtracking($G$, $m$, $n$, $i$, $c$, $T$, $T_0$);
**begin**
1:   $f(v_i) := c$;
2:   **if** $f$ restricted to $v_1, v_2, \ldots, v_i$ is not a graceful labeling
3:     **return** false;
4:   **end if**
5:   **if** $i = n$
6:     **if** $f$ is a graceful labeling
7:       report the graceful labeling;
8:       **return** true;
```

```
9:      **else**
10:       **return** false;
11:     **end if**
12:   **end if**
13:   $T_1$ := get current time;
14:   **if** $T_1 - T_0 > T$
15:     **return** false;
16:   **end if**
17:   **if** $i > 1$ **and** $w(v_i, \sigma, f) \geq m/2$
18:     **for each** $\ell$ **from** 0 **to** $m$
19:       **if** Backtracking($G$, $m$, $n$, $i+1$, $\ell$, $T$, $T_0$)
20:         **return** true;
21:       **endif**
22:     **endfor**
23:   **end if**
24:   **if** $i > 1$ **and** $w(v_i, \sigma, f) \leq m/2$
25:     **for each** $\ell$ **from** $m$ **downto** 0
26:       **if** Backtracking($G$, $m$, $n$, $i+1$, $\ell$, $T$, $T_0$)
27:         **return** true;
28:       **endif**
29:     **endfor**
30:   **end if**
31:   **return** false;
**end.**
```

We will show in the next section that the algorithm Backtracking strongly outperforms the standard backtracking algorithm with a random static variable ordering and a static value ordering. In particular, the standard backtracking algorithm has managed to compute graceful labelings of generalized Petersen graph $P(n,k)$ only for n up to 11, while Backtracking has enabled us to find graceful labelings for all graphs $P(n,k)$ with $n \leq 75$.

3. Computation

The following result presented in [5] can be used to reduce the number of graphs tested by the algorithm.

Theorem 1. *Let $n > 3$ and k_1, k_2 relatively prime to n with $k_1 k_2 \equiv 1$ mod n. Then $P(n,k_1) \cong P(n,k_2)$.*

Table 1 depicts all triples of values n, k_1, k_2, with $11 \leq n \leq 100$, which imply $P(n,k_1) \cong P(n,k_2)$.

Table 1. **Triples n, k_1, k_2 which for $11 \leq n \leq 100$ imply $P(n, k_1) \cong P(n, k_2)$**

n	k_1	k_2	n	k_1	k_2	n	k_1	k_2	n	k_1	k_2	n	k_1	k_2
11	2	5	11	3	4	13	2	6	14	3	5	15	2	7
17	2	8	17	3	6	19	2	9	19	4	5	20	3	7
21	2	10	23	2	11	23	3	8	23	4	6	25	2	12
26	3	9	27	2	13	27	4	7	29	2	14	29	3	10
29	5	6	31	2	15	31	4	8	32	3	11	33	2	16
34	5	7	35	2	17	35	3	12	35	4	9	37	2	18
38	3	13	39	2	19	39	4	10	39	5	8	41	2	20
41	3	14	41	6	7	43	2	21	43	4	11	44	3	15
44	5	9	45	2	22	47	2	23	47	3	16	47	4	12
47	6	8	49	2	24	49	5	10	50	3	17	51	2	25
51	4	13	53	2	26	53	3	18	53	6	9	54	5	11
55	2	27	55	4	14	55	7	8	56	3	19	57	2	28
59	2	29	59	3	20	59	4	15	59	5	12	59	6	10
61	2	30	62	3	21	62	7	9	63	2	31	63	4	16
64	5	13	65	2	32	65	3	22	65	6	11	67	2	33
67	4	17	68	3	23	69	2	34	69	5	14	69	7	10
71	2	35	71	3	24	71	4	18	71	6	12	71	8	9
73	2	36	74	3	25	74	5	15	75	2	37	75	4	19
76	7	11	77	2	38	77	3	26	77	6	13	79	2	39
79	4	20	79	5	16	79	8	10	80	3	27	81	2	40
83	2	41	83	3	28	83	4	21	83	6	14	83	7	12
84	5	17	85	2	42	86	3	29	87	2	43	87	4	22
87	8	11	89	2	44	89	3	30	89	5	18	89	6	15
89	9	10	90	7	13	91	2	45	91	4	23	92	3	31
93	2	46	94	5	19	95	2	47	95	3	32	95	4	24
95	6	16	95	8	12	97	2	48	97	7	14	98	3	33
98	9	11	99	2	49	99	4	25	99	5	20			

3.1. Results

We are ready now to present the main result of the paper.

Theorem 2. *The Petersen graph $P(n, k)$ is graceful for any $5 \leq n \leq 75$.*

Proof. It is proved in [3] that the Petersen graph $P(n, k)$ is graceful with n up to 10. With respect to the isomorphic pairs given by Theorem 1, graceful labelings for all graphs $P(n, k)$ with $11 \leq n \leq 75$ have been obtained by the algorithm Backtracking presented in Section 2. These labelings are presented in Table 2 for $n \leq 20$, while the others can be obtained by the authors or at the web page http://omr.fnm.um.si/wp-content/uploads/Zaposleni/claniOddelka/aleksander.vesel/labelings.pdf. The labels of the vertices of $P(n, k)$ are listed with respect to the sequence $u_1, u_2, \ldots, u_n, v_1, v_2, \ldots, v_n$ given in the definition of the generalized Petersen graph

$P(n,k)$. As an example, see the graceful labeling of $P(11,2)$ shown in Table 2 which is depicted in Figure 1. $\square$

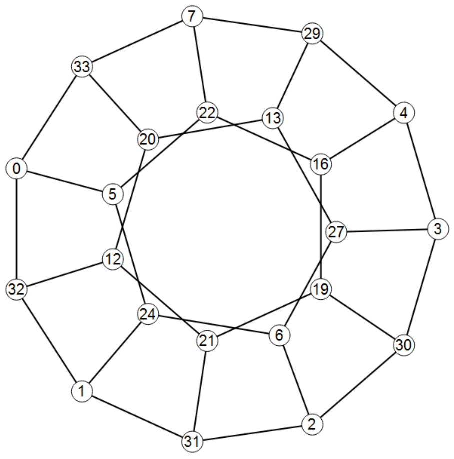

Figure 1. A graceful labeling of $P(11,2)$

Table 2: Results

graph	labeling
$P(11,2)$	3,30,2,31,1,32,0,33,7,29,4,27,19,6,21,24,12,5,20,22,13,16
$P(11,3)$	6,29,15,31,1,32,0,28,2,27,5,25,22,23,4,13,12,33,18,8,9,20
$P(12,2)$	7,34,16,4,18,9,22,2,33,1,35,0,31,8,12,14,29,17,6,32,11, 3, 10, 36
$P(12,3)$	7,33,15,5,16,10,3,34,1,35,0,29,28,8,13,9,31,18,23,4,17, 11, 36, 26
$P(12,4)$	3, 7, 33, 4, 24, 14, 2, 34, 1, 35, 0, 28, 10, 31, 6, 25, 8, 20, 5, 16, 23, 12, 36, 11
$P(12,5)$	8, 35, 0, 34, 1, 33, 2, 32, 15, 21, 16, 20, 31, 13, 36, 14, 19, 5, 23, 7, 6, 22, 3, 4
$P(13,2)$	5, 34, 4, 35, 3, 36, 2, 37, 1, 38, 0, 39, 11, 32, 24, 6, 16, 28, 33, 7, 22, 25, 15, 9, 19, 23
$P(13,3)$	36, 2, 37, 1, 38, 0, 39, 6, 35, 3, 34, 4, 8, 9, 11, 13, 27, 18, 19, 14, 28, 20, 26, 17, 25, 10
$P(13,4)$	6, 34, 23, 30, 11, 2, 37, 1, 38, 0, 33, 3, 35, 8, 9, 31, 16, 32, 5, 15, 28, 12, 39, 10, 13, 29
$P(13,5)$	32, 0, 38, 1, 37, 2, 36, 18, 21, 9, 25, 34, 3, 8, 39, 23, 29, 14, 28, 6, 10, 4, 20, 30, 7, 24
$P(14,2)$	3, 39, 2, 40, 1, 41, 0, 42, 7, 38, 4, 37, 5, 33, 32, 25, 6, 27, 28, 16, 8, 23, 31, 35, 10, 19, 15, 34
$P(14,3)$	3, 39, 2, 40, 1, 41, 0, 42, 7, 38, 4, 37, 5, 33, 32, 18, 8, 13, 16, 25, 26, 20, 30, 14, 11, 12, 15, 19
$P(14,4)$	3, 39, 2, 40, 1, 41, 0, 42, 7, 38, 4, 37, 5, 33, 32, 23, 9, 21, 6, 20, 27, 29, 31, 18, 10, 14, 19, 22
$P(14,6)$	3, 39, 2, 40, 1, 41, 0, 42, 7, 38, 4, 37, 5, 33, 32, 17, 29, 31, 26, 22, 6, 28, 8, 15, 9, 19, 13, 18
$P(15,2)$	36, 8, 20, 31, 18, 26, 6, 24, 2, 42, 1, 44, 0, 37, 4, 5, 43, 30, 16, 39, 40, 22, 17, 41, 13, 3, 14, 45, 11, 9
$P(15,3)$	36, 8, 22, 19, 38, 18, 5, 41, 3, 43, 1, 44, 0, 37, 6, 12, 35, 21, 7, 13, 28, 39, 15, 20, 4, 30, 26, 45, 14, 17

Continued on next page

Table 2 – continued from previous page

graph	labeling
$P(15,4)$	9, 42, 26, 22, 27, 15, 33, 41, 2, 43, 1, 44, 0, 38, 3, 40, 8, 6, 39, 4, 34, 20, 14, 5, 19, 31, 12, 45, 28, 10
$P(15,5)$	34, 8, 41, 4, 39, 15, 17, 42, 2, 43, 1, 44, 0, 32, 3, 19, 38, 7, 27, 28, 5, 11, 6, 20, 24, 22, 16, 45, 40, 12
$P(15,6)$	10, 31, 23, 36, 12, 38, 3, 42, 2, 43, 1, 44, 0, 37, 4, 41, 30, 32, 9, 40, 22, 7, 20, 14, 11, 15, 39, 45, 17, 33
$P(16,2)$	9, 1, 47, 0, 44, 2, 43, 8, 38, 11, 35, 14, 33, 4, 19, 45, 46, 3, 7, 48, 6, 5, 10, 39, 15, 17, 28, 42, 16, 22, 30, 13
$P(16,3)$	40, 1, 47, 0, 45, 2, 46, 19, 33, 13, 21, 34, 28, 5, 41, 3, 7, 36, 18, 48, 14, 30, 6, 4, 9, 15, 20, 16, 32, 39, 11, 29
$P(16,4)$	38, 1, 47, 0, 45, 2, 46, 4, 31, 12, 40, 9, 23, 6, 17, 5, 15, 41, 8, 48, 11, 3, 10, 7, 26, 28, 16, 39, 33, 19, 37, 13
$P(16,5)$	30, 1, 47, 0, 45, 2, 46, 4, 44, 28, 33, 15, 36, 12, 21, 20, 5, 40, 11, 48, 43, 25, 8, 39, 7, 32, 18, 27, 6, 38, 35, 26
$P(16,6)$	36, 1, 47, 0, 45, 2, 46, 4, 44, 3, 7, 32, 23, 35, 11, 26, 6, 39, 10, 48, 17, 20, 14, 5, 13, 9, 33, 18, 43, 12, 30, 31
$P(16,7)$	30, 1, 47, 0, 45, 2, 46, 4, 44, 3, 42, 9, 21, 32, 13, 14, 6, 38, 12, 48, 18, 17, 20, 27, 8, 37, 10, 5, 39, 15, 7, 22
$P(17,2)$	0, 40, 6, 44, 17, 26, 8, 20, 37, 22, 25, 4, 45, 2, 48, 1, 50, 51, 7, 12, 14, 43, 46, 27, 10, 13, 33, 38, 5, 16, 47, 11, 3, 15
$P(17,3)$	0, 39, 2, 40, 23, 12, 31, 38, 18, 27, 48, 5, 47, 3, 49, 1, 50, 51, 6, 34, 15, 24, 7, 25, 46, 21, 29, 20, 45, 13, 32, 4, 36, 19
$P(17,4)$	0, 39, 6, 44, 7, 37, 16, 29, 28, 4, 46, 3, 47, 2, 49, 1, 50, 51, 8, 13, 22, 15, 10, 41, 40, 9, 45, 12, 23, 24, 5, 17, 27, 34
$P(17,5)$	0, 41, 3, 40, 5, 39, 44, 34, 46, 7, 47, 4, 48, 2, 49, 1, 50, 51, 16, 11, 10, 9, 15, 38, 37, 17, 24, 14, 20, 6, 30, 22, 33, 29
$P(17,7)$	0, 42, 4, 45, 15, 30, 19, 41, 22, 24, 47, 3, 48, 2, 49, 1, 50, 51, 6, 37, 5, 9, 35, 12, 25, 23, 16, 8, 38, 17, 39, 21, 10, 26
$P(18,2)$	0, 42, 6, 46, 16, 43, 25, 38, 24, 44, 28, 47, 4, 48, 2, 51, 1, 53, 54, 7, 13, 12, 45, 37, 8, 14, 39, 40, 17, 19, 5, 9, 50, 18, 3, 15
$P(18,3)$	0, 41, 2, 42, 34, 17, 36, 24, 8, 7, 9, 51, 5, 50, 3, 52, 1, 53, 54, 6, 35, 16, 27, 13, 39, 18, 28, 21, 43, 23, 48, 14, 33, 4, 38, 22
$P(18,4)$	0, 42, 6, 47, 7, 18, 37, 13, 35, 22, 4, 49, 3, 50, 2, 52, 1, 53, 54, 8, 41, 17, 15, 28, 16, 40, 19, 23, 48, 12, 36, 38, 5, 14, 10, 24
$P(18,5)$	0, 39, 3, 41, 38, 14, 22, 15, 5, 49, 7, 50, 4, 51, 2, 52, 1, 53, 54, 18, 34, 9, 42, 25, 44, 29, 46, 26, 37, 16, 23, 6, 27, 19, 36, 40
$P(18,6)$	0, 11, 44, 6, 49, 10, 50, 9, 31, 25, 4, 48, 3, 51, 2, 52, 1, 53, 54, 47, 15, 26, 18, 28, 8, 12, 5, 13, 41, 20, 7, 17, 29, 45, 16, 37
$P(18,7)$	0, 44, 4, 47, 6, 34, 26, 35, 31, 30, 5, 50, 3, 51, 2, 52, 1, 53, 54, 9, 37, 28, 45, 12, 21, 41, 43, 13, 25, 8, 39, 19, 18, 14, 27, 32
$P(18,8)$	0, 44, 5, 48, 8, 15, 36, 25, 41, 22, 4, 50, 3, 51, 2, 52, 1, 53, 54, 12, 47, 18, 43, 46, 27, 49, 28, 32, 9, 14, 6, 17, 10, 24, 13, 26
$P(19,2)$	8, 49, 7, 50, 6, 51, 5, 52, 4, 53, 3, 54, 2, 55, 1, 56, 0, 57, 17, 47, 39, 9, 23, 43, 45, 10, 21, 40, 34, 11, 31, 37, 48, 12, 44, 32, 29, 18
$P(19,3)$	8, 49, 7, 50, 6, 51, 5, 52, 4, 53, 3, 54, 2, 55, 1, 56, 0, 57, 17, 47, 37, 35, 12, 33, 30, 42, 20, 29, 19, 9, 32, 38, 26, 25, 23, 16, 39, 43
$P(19,4)$	8, 49, 7, 50, 6, 51, 5, 52, 4, 53, 3, 54, 2, 55, 1, 56, 0, 57, 17, 47, 36, 24, 16, 9, 26, 29, 23, 41, 37, 31, 19, 10, 22, 13, 20, 30, 43, 39
$P(19,6)$	8, 49, 7, 50, 6, 51, 5, 52, 4, 53, 3, 54, 2, 55, 1, 56, 0, 57, 17, 47, 16, 34, 32, 14, 28, 9, 18, 10, 37, 35, 43, 39, 19, 36, 27, 13, 26, 42
$P(19,7)$	8, 49, 7, 50, 6, 51, 5, 52, 4, 53, 3, 54, 2, 55, 1, 56, 0, 57, 17, 47, 45, 9, 21, 12, 15, 31, 14, 22, 19, 28, 24, 10, 34, 36, 37, 32, 29, 48
$P(19,8)$	8, 49, 7, 50, 6, 51, 5, 52, 4, 53, 3, 54, 2, 55, 1, 56, 0, 57, 17, 47, 32, 14, 15, 26, 19, 11, 21, 9, 20, 13, 18, 28, 27, 38, 45, 34, 41, 35

Continued on next page

Table 2 – **continued from previous page**

graph	labeling
$P(20,2)$	23, 48, 19, 10, 43, 1, 59, 0, 56, 2, 55, 4, 52, 6, 58, 13, 36, 53, 31, 50, 11, 9, 39, 46, 8, 3, 18, 60, 22, 5, 15, 54, 20, 7, 28, 51, 49, 35, 25, 24
$P(20,3)$	16, 31, 22, 14, 58, 1, 59, 0, 55, 2, 54, 5, 53, 3, 49, 6, 48, 18, 44, 21, 29, 43, 28, 51, 4, 25, 23, 60, 26, 37, 9, 7, 19, 50, 24, 39, 10, 8, 12, 17
$P(20,4)$	21, 11, 42, 2, 58, 1, 59, 0, 54, 3, 56, 4, 53, 6, 52, 7, 30, 46, 14, 57, 50, 38, 8, 5, 25, 12, 20, 60, 45, 27, 18, 10, 32, 34, 17, 51, 13, 16, 22, 9
$P(20,5)$	46, 8, 37, 36, 16, 25, 57, 2, 58, 1, 59, 0, 53, 3, 55, 7, 56, 11, 51, 9, 5, 52, 10, 32, 40, 38, 6, 33, 30, 22, 24, 60, 23, 14, 47, 41, 13, 4, 29, 35
$P(20,6)$	21, 23, 38, 16, 35, 3, 57, 2, 58, 1, 59, 0, 52, 4, 55, 5, 54, 8, 53, 9, 47, 6, 20, 46, 12, 7, 33, 44, 19, 11, 32, 60, 41, 10, 26, 42, 29, 13, 25, 50
$P(20,8)$	25, 42, 44, 4, 56, 3, 57, 2, 58, 1, 59, 0, 50, 5, 54, 6, 21, 40, 13, 55, 41, 8, 7, 9, 24, 15, 22, 10, 27, 30, 20, 60, 17, 33, 16, 53, 47, 51, 23, 14
$P(20,9)$	1, 15, 55, 4, 56, 3, 57, 2, 58, 1, 59, 0, 49, 5, 36, 20, 34, 33, 30, 42, 13, 52, 10, 45, 14, 35, 40, 29, 48, 47, 24, 60, 19, 53, 16, 25, 11, 27, 22, 46

3.2. Comparison with other methods

In [3], an integer programming formulation (ILP) and a constraint programming (CSP) formulation were proposed to solve the graceful labeling problem. The approach was applied for some generalized Petersen graphs, double cones graphs as well as for product graphs of the form $K_4 \times P_n$. The results obtained by CSP are better then the ones obtained by ILP for most graphs. Therefore, in order to demonstrate the performance of our algorithm on generalized Petersen graphs, we compare our results with the results obtained by using CSP. The comparison is shown in Table 3. All results are obtained by a computer program in the C++ programming language so that the computations were performed on a personal computer.

Table 3. **Comparison of the execution time (in seconds) with the results from [2]**

graph	n	e	CSP	B	graph	n	e	CSP	B
$P(5,1)$	10	15	0.04	0.003	$P(5,2)$	10	15	< 0.01	0.003
$P(6,1)$	12	18	0.06	0.004	$P(6,2)$	12	18	0.54	0.004
$P(7,1)$	14	21	0.43	0.002	$P(7,2)$	14	21	3.69	0.003
$P(7,3)$	14	21	0.70	0.003	$P(8,1)$	16	24	3.95	0.003
$P(8,2)$	16	24	19.23	0.001	$P(8,3)$	16	24	77.04	0.005
$P(9,1)$	18	27	71.07	0.004	$P(9,2)$	18	27	636.93	0.004
$P(9,3)$	18	27	144.78	0.004	$P(9,4)$	18	27	58.28	0.003
$P(10,1)$	20	30	16.26	0.003	$P(10,2)$	20	30	7311.20	0.005
$P(10,3)$	20	30	1648.89	0.003	$P(10,4)$	20	30	1109.34	0.003

In Table 3, the CSP column shows the execution time of solving constraint programming presented in [3], while the B column gives the execution time of the algorithm Backtracking. All timing data listed is in seconds. It can be seen that our algorithm outperforms CSP for almost every instance of the generalized Petersen graphs. For graphs of order around 20, CSP needs thousands of seconds, while the running time of the algorithm Backtracking does not exceed five milliseconds. It is worth mentioning

that the running time of the program did not exceed one minute even for the largest graphs being tested.
In order to compare the algorithm Backtracking with a standard backtracking algorithm, we consider the number of nodes of search trees of both approaches. As usual, a search tree is developed until a solution (a graceful labeling) is found.
Experimental results show that the number of nodes of a search tree developed by the algorithm Backtracking is much smaller then the number of nodes of a search tree developed by a standard backtracking algorithm for all connected 3-regular graphs. In most cases it is even impossible to determine the number of nodes developed by a standard backtracking algorithm, since the corresponding computation is not concluded within a reasonable time. If a graph of interest is very small, the difference between these two numbers is substantial. For instance, if our approach is performed on $K_{3,3}$, the number of nodes of the obtained search tree is 13, while the number of nodes developed by a standard backtracking algorithm is 1768 for this graph.

4. Conclusions

In this paper, we propose a backtracking algorithm for the graceful labeling problem. The algorithm is based on a specific static variable ordering and dynamic value ordering. The presented approach was applied to find graceful labelings for generalized Petersen graphs.
We argue that the presented approach strongly outperforms the standard backtracking algorithm. In particular, we have been able to to find graceful labelings for all generalized Petersen graphs $P(n, k)$ with $n \leq 75$, while the standard algorithm has managed to compute graceful labelings of these graphs only for n up to 11.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under the grant 61309015, China Postdoctoral Science Foundation under the grant 2014M560851 and by the Ministry of Science of Slovenia under the grant 0101-P-297.

References

[1] G.S. Bloom and S.W. Golomb, *Applications of numbered undirected graphs*, Proceedings of the IEEE **65** (1977), no. 4, 562–570.

[2] J.A. Gallian, *A dynamic survey of graph labeling*, Electron. J. Combin. **16** (2009), no. 6, 1–219.

[3] T.A. Redl, *Graceful graphs and graceful labelings: two mathematical programming formulations and some other new results*, Congr. Numer. **164** (2003), 17–32.

[4] A. Rosa, *On certain valuations of the vertices of a graph*, Theory of Graphs (Internat. Symposium, Rome, 1966, pp. 349–355.

[5] A. Steimle and W. Staton, *The isomorphism classes of the generalized petersen graphs*, Discrete Math. **309** (2009), no. 1, 231–237.