Tema
Tendências em Matemática Aplicada e Computacional, **19**, N. 1 (2018), 1-13
© 2018 Sociedade Brasileira de Matemática Aplicada e Computacional
www.scielo.br/tema
doi: 10.5540/tema.2018.019.01.0001

A Novel Approach to Find Pseudo–peripheral Vertices for Snay's Heuristic

S.L. GONZAGA DE OLIVEIRA* and J.A.B. BERNARDES

Received on November 17, 2016 / Accepted on November 16, 2017

ABSTRACT. The solution of linear systems represented by $Ax = b$ is fundamental in many numerical simulations in science and engineering. Reducing the profile of A can reduce the storage requirements and time processing costs of solving such linear systems. In this work, we propose a generalized algorithm for finding pseudo–peripheral vertices for Snay's heuristic. In experiment performed on 36 instances contained in the Harwell-Boeing and SuiteSparse matrix collections, it has been found that the number of pseudo–peripheral vertices selected in Snay's heuristic may be suitable for small instances, but it is insufficient to obtain reasonable results in instances that are not small. This paper recommends to select up to 26% (0.3%) of pseudo–peripheral vertices in relation to the instance size when applied to instances smaller than 3,000 (larger than 20,000) vertices.

Keywords: Profile reduction, sparse matrix, reordering algorithms.

1 INTRODUCTION

Several real-world problems reduce into a linear system in the form $Ax = b$, where A is an $n \times n$ large-scale sparse matrix, x is the unknown n-vector solution which is sough, and b is a known n-vector. Thus, the resolution of large sparse linear systems in this form is crucial in various engineering and science applications. It is normally the part of the simulation that requires the highest processing cost. If the coefficient matrix A is dense, users employ a direct method (e.g., Gaussian Elimination, LU factorization, Cholesky factorization, etc.) to solve the linear system. On the other hand, in practice A occurs to be sparse, symmetric, and positive definite, e.g., the ones arising from the discretization of elliptic or parabolic partial differential equations [1]. Reducing the profile of a sparse symmetric matrix A can benefit the storage requirements and processing times to solve the linear system.

Let $A = [a_{ij}]$ be an $n \times n$ symmetric matrix associated with a connected undirected graph $G = (V,E)$, where V and E are sets of vertices and edges, respectively. The profile of a matrix A is defined as $profile(A) = \sum_{i=1}^{n} \left(i - \min_{1 \leq j < i} (j \mid a_{ij} \neq 0) \right)$ [13]. Equivalently, the profile of G for a vertex labeling $S = \{s(v_1), s(v_2), \cdots, s(v_{|V|})\}$ (i.e. a bijective mapping $s : V \to \{1, 2, \cdots, |V|\}$) is

*Corresponding author: Sanderson L. Gonzaga de Oliveira – E-mail: sanderson@dcc.ufla.br.
Universidade Federal de Lavras, Brasil. E-mail: jrassis@posgrad.ufla.br

defined as $profile(G) = \sum_{v \in V} \max_{\{v,u\} \in E} [|s(v) - s(u)|]$, where $s(v)$ and $s(u)$ are labels of vertices v and u, respectively. Then, vertices $v, u \in V$ with labels $s(v) = i$ and $s(u) = j$ are associated with lines i and j of A, respectively. Therefore, $a_{ii} \neq 0$, $a_{ij} \neq 0 \Leftrightarrow \{v,u\} \in E$, and $a_{ij} = 0 \Leftrightarrow \{v,u\} \notin E$. The profile minimization problem is NP-hard [17] and several heuristics for profile reduction have been proposed since the 1960s [4].

A systematic review [4] reported Snay's heuristic [19] as one of the most promising heuristics for profile reduction with low computational costs. This algorithm is an example of heuristics known as level set reorderings, where the vertices in a graph are labeled taking into account that the vertices are firstly partitioned into level sets, that is, the vertices are partitioned in relation to the distance from a given starting vertex. Thus, this starting vertex plays an important role in this context. In general, the quality of the results will be profoundly affected by the choice of the starting vertex and by the ordering of the vertices within level sets.

In Snay's heuristic, an initial step selects 10 pseudo–peripheral vertices as starting vertices of the graph labeling. The objective of this work is to identify the number of pseudo–peripheral vertices that Snay's heuristic must select to obtain the smallest profile of the instance, considering the profile that can be achieved by the heuristic, and also taking into account the computational times of the heuristic during this process. This paper is a revised and expanded version of a paper presented at the XXXVI Brazilian National Congress in Applied and Computational Mathematics (CNMAC 2016) [10].

The remainder of this manuscript is organized as follows. Section 2 outlines the new algorithm for finding pseudo–peripheral vertices for Snay's heuristic. Section 3 presents and analyzes the results. Finally, Section 4 provides the conclusions.

2 A NEW ALGORITHM FOR FINDING PSEUDO–PERIPHERAL VERTICES

Snay's heuristic [19] is composed of two main steps. The first step is a heuristic algorithm that determines a starting vertex s for the graph labeling. The second step then labels the vertices, beginning from s, and chooses the next vertex to be labeled from a set of eligible vertices by means of a modified breadth-first search (BFS) procedure. In short, given a connected graph $G = (V,E)$ and a starting vertex $v \in V$, the breadth-first search procedure traverses the graph vertices in ascending-distance order from v, where distance is a single path with fewest edges between two vertices.

In addition to vertices adjacent to vertices already labeled, Snay's heuristic considers also as candidate vertices to be labeled those vertices contained in the second level of vertices already labeled. To choose the next vertex to be labeled, its priority function considers the *current* degree of a vertex. Specifically, Snay's heuristic sorts candidate vertices in ascending-degree order, considering only adjacencies to vertices that are neither labeled nor are adjacent to vertices already labeled. Algorithm 1 [13] shows a pseudocode of Snay's heuristic.

Algorithm 1: Second step of Snay's heuristic.

Input: a connected graph $G = (V, E)$; a starting vertex $s \in V$;
Output: a graph labeling $L = \{l_1, l_2, \cdots, l_{|V|}\}$ for V (i.e. L is a bijective mapping from V to the set $\{1, 2, \cdots, |V|\}$), where $|V|$ is the cardinality of V;

1 **begin**
  `// every vertex contains a status` *inactive*
2 $l_1 \leftarrow s$;
3 $s.status \leftarrow labeled$;
4 $H \leftarrow \varnothing$;
5 $C \leftarrow \varnothing$;
6 **for** $(i \leftarrow 2; i \leq |V|; i \leftarrow i+1)$ **do**
7 $N \leftarrow Adj(G, l_{i-1}) - (\{l_1, l_2, \cdots, l_{i-1}\} \cup H)$, where $Adj(G,u) = \{w \in V : \{u,w\} \in E\}$;
8 $(\forall v \in N)$ $v.status \leftarrow hopeful$;
9 $H \leftarrow H \cup N$;
10 $C \leftarrow C \cup N$;
11 **foreach** $(u \in N)$ **do**
12 $(\forall v \in Adj(G,u) - \{l_1, l_2, \cdots, l_{i-1}\})$ $v.status \leftarrow candidate$;
13 $C \leftarrow C \cup (Adj(G,u) - \{l_1, l_2, \cdots, l_{i-1}\})$;
14 **end**
15 $min \leftarrow +\infty$;
16 $v \leftarrow \emptyset$;
17 **foreach** $(w \in C)$ **do**
18 $w_r \leftarrow |Adj(G,w) - (\{l_1, l_2, \cdots, l_{i-1}\} \cup H)|$;
19 **if** $(w \in H)$ **then** $w_r \leftarrow w_r - 1$;
20 **if** $(w_r < min)$ **then**
21 $min \leftarrow w_r$;
22 $v \leftarrow w$;
23 **end**
24 **end**
25 $l_i \leftarrow v$;
26 $v.status \leftarrow labeled$;
27 $H \leftarrow H - \{v\}$;
28 $C \leftarrow C - \{v\}$;
29 **end**
30 **return** (L);
31 **end**

The starting vertex s is established with the first label in Snay's heuristic (see line 2 in Algorithm 1). H is a set composed of adjacent vertices to the vertices already labeled (i.e. the set $\{l_1, l_2, ..., l_{i-1}\}$). N is a set composed of adjacent vertices to the last vertex labeled l_{i-1} (except vertices already labeled and vertices contained in the H set: see line 7). Every adjacent vertex to the last vertex labeled l_{i-1} is added to the H and C sets during iteration i (in lines 9 and 10, respectively). C is a set composed of candidate vertices to be labeled: vertices contained in the H set (see line 9) and adjacent vertices to vertices in H (see lines 11–14), i.e. adjacent vertices to each vertex contained in the N set are also added to the C set. Therefore, $H \subseteq C$. In addition, let's consider that each vertex is an object (i.e. in an object–oriented programming language)

that contains at least two attributes: the value r of adjacencies to vertices that neither have been labeled nor belong to the C set, and a status attribute. This status attribute indicates if the vertex was already labeled (see lines 3 and 26), if it belongs to the $C-H$ (a candidate vertex) or H (a hopeful vertex, according to Snay's terminology [19]) sets, or if it has not been considered in the algorithm yet (an inactive vertex).

The *foreach* loop in lines 17–24 selects a vertex with minimum degree (considering only vertices that are neither labeled nor are adjacent to a vertex already labeled: see line 18) to be labeled in line 25. This vertex is removed from the H and C sets in lines 27 and 28, respectively. Adjacent vertices to vertices already labeled (i.e $\forall v \in H$) have a higher priority than other candidate vertices (see line 19). Finally, this algorithm returns a reordering of graph $G=(V,E)$ in line 30. This results in concentrating the lower and upper triangular parts of A [corresponding to a graph $G=(V,E)$] towards its main diagonal. This is an efficient heuristic that reduces a profile of a matrix A to a considerable extent at low cost [4].

Snay's heuristic labels the vertices in a graph $G=(V,E)$ starting with 10 pseudo–peripheral vertices, i.e. 10 labelings are provided in this heuristic. More specifically, in the first step of Snay's heuristic [19], 10 pseudo–peripheral vertices are selected as starting vertices for each of the 10 graph labelings obtained in the second step of the heuristic. The method for determining the starting vertex proposed by Snay [19] consists of considering, starting from a random vertex $v \in V$, a set D composed of five vertices belonging to the last level of the *level structure rooted* at v. Let $G=(V,E)$ be a connected and simple graph. Given a vertex $v \in V$, the level structure rooted at v, with depth $\ell(v)$, is a partitioning $\mathscr{L}(v) = \{L_0(v), L_1(v), \ldots, L_{\ell(v)}(v)\}$, where $L_0(v) = \{v\}$, $L_i(v) = Adj(L_{i-1}(v)) - \bigcup_{j=0}^{i-1} L_j(v)$, for $i = 1, 2, 3, \ldots, \ell(v)$ and $Adj(U) = \{w \in V : (u \in U \subseteq V)\ \{u,w\} \in E\}$ returns the adjacent vertices to the argument [13].

Snay's algorithm for finding pseudo–peripheral vertices also builds the set Q comprised of five vertices belonging to $L_{\ell(u)}(u)$, where $u \in D \subseteq L_{\ell(v)}(v)$. Thus, the $D \cup Q$ set contains the 10 starting vertices of Snay's heuristic [19]. Then, Snay's heuristic [19] returns the graph labeling that provides the smaller profile among the 10 graph labelings computed (see Algorithm 1).

As described, Snay's algorithm returns 10 pseudo–peripheral vertices. This algorithm was generalized in this present work in order to receive the parameter ν so that this generalized Snay's algorithm for finding pseudo–peripheral vertices returns a set containing $2 \cdot \nu$ vertices. Algorithm 2 shows a pseudo–code of this generalized algorithm. Algorithm 2 receives a graph $G=(V,E)$ and the value $\nu \in \mathbb{N}^*$ of pseudo–peripheral vertices to be selected for both D and Q sets. Algorithm 2 returns a set of pseudo–peripheral vertices $D \cup Q$. Therefore, Snay's algorithm for finding pseudo–peripheral vertices [19] is a special case of Algorithm 2, when setting $\nu = 5$.

The condition in line 2 in Algorithm 2 ensures that $|V| \geq 2 \cdot \nu$. A random vertex is assigned to the vertex v in line 3. In line 4, $\mathscr{L}(v)$ is built. The construction of the set D (Q) is started in line 5 (21). The variable k, initialized in line 7 (22), is used to ensure that $|D| = |Q| = \nu$, since $L_{\ell(v)}(v)$ $\left[L_{\ell(u)}(u)\right]$ may have less than ν vertices. The variable i, initialized in line 8 (23), is used in the

Algorithm 2: *GeneralizedPseudoPeripheralVerticesSnay* (pseudo–peripheral vertices to be employed in Snay's heuristic).

Input: graph $G = (V, E)$; value $\nu \in \mathbb{N}^*$ of pseudo–peripheral vertices to be selected in both sets D and Q;
Output: a set of pseudo–peripheral vertices $D \cup Q$, where $|D \cup Q| = 2 \cdot \nu$;

```
1  begin
2      if (|V| < 2 · ν) then return;
3      v ← RandomVertex(V);
4      L(v) ← BFS(v);
5      D ← ∅; // |D| = ν
6      u ← ∅; // u ∈ L_ℓ(v)(v)
7      k ← 0;
8      i ← 0;
9      while (i < ν) do
10         foreach (w ∈ L_ℓ(v)−k(v)) do
11             D ← D ∪ {w};
12             i ← i + 1;
13             if (u = ∅) then
14                 u ← w;
15                 L(u) ← BFS(u);
16             end
               // go to line 17 if the condition is satisfied
17             if (i ≥ ν) then break;
18         end
19         k ← k + 1;
20     end
21     Q ← ∅; // |Q| = ν
22     k ← 0;
23     i ← 0;
24     while (i < ν) do
25         foreach (w ∈ L_ℓ(u)−k(u) − D) do
26             Q ← Q ∪ {w};
27             i ← i + 1;
               // go to line 26 if the condition is satisfied
28             if (i ≥ ν) then break;
29         end
30         k ← k + 1;
31     end
32     return D ∪ Q;
33 end
```

while loop in lines 9–20 (24–31), where ν vertices are inserted into D in line 11 (Q in line 26). $\mathscr{L}(u)$ is built in line 15.

For clarity, Algorithm 2 shows two similar parts of pseudo–codes in lines 5–20 (to build the D set) and 21–31 (to build the Q set). Finally, the algorithm returns the $D \cup Q$ set in line 32, where $|D \cup Q| = 2 \cdot \nu$.

Algorithm 3 [13] evaluates the profile of the matrix corresponding to the graph $G = (V,E)$ so that the vertices in V are reordered according to Algorithm 1 (see line 7 in Algorithm 3) for each pseudo–peripheral vertex returned in Algorithm 2 (see line 3 in Algorithm 3). Algorithm 3 receives a graph $G = (V,E)$ and a value $\nu \in \mathbb{N}^*$, where $2 \cdot \nu$ pseudo–peripheral are selected in Algorithm 2. Algorithm 3 returns a vertex labeling of the vertices contained in V.

Algorithm 3: Evaluates the profile of the matrix corresponding to the graph $G = (V,E)$ for each starting vertex selected in Algorithm 2.

```
   Input: graph G = (V,E); value ν ∈ ℕ*;
   Output: a vertex labeling S for V;
1  begin
2      if (2·ν > |V|) then ν ← ⌊|V|/2⌋;
       // Algorithm 2
3      C ← GeneralizedPseudoPeripheralVerticesSnay(G,ν);
4      small_profile ← +∞;
5      best_labeling ← ∅;
6      foreach (w ∈ C) do
7          S ← SnayHeuristic(G,w); // Algorithm 1
8          profile ← ComputeProfile(G,S);
9          if (profile < small_profile) then
10             small_profile ← profile;
11             best_labeling ← S;
12         end
13     end
14     return best_labeling;
15 end
```

3 RESULTS AND ANALYSIS

Numerical simulations were performed in order to evaluate the relation between the number of pseudo–peripheral vertices selected in the algorithm for finding pseudo–peripheral vertices (Algorithm 2) and the profile achieved by Snay's heuristic (Algorithm 1). To apply Snay's heuristics in asymmetric instances, the asymmetric matrix is added to its transpose matrix (i.e. $A + A^T$), resulting in a symmetric matrix. Subsequently, the heuristic is applied to the graph corresponding to the matrix achieved and the vertex labeling obtained is used to label the original asymmetric matrix. This is one of the simplest approach and one that presents the lowest storage and processing costs among the approaches to transform an asymmetric matrix to a symmetric one. Moreover, Snay's heuristic is applied to each connected component of a disconnected graph. Snay's heuristic was implemented in the C++ programming language. Sections 3.1 and 3.2 show simulations performed with instances contained in the Harwell-Boeing and SuiteSparse matrix collections, respectively.

3.1 Experiments with instances contained in the Harwell-Boeing sparse matrix collection

This section shows results of Snay's heuristic (in conjunction with the algorithms for finding pseudo–peripheral vertices) applied to 11 instances (ranging from 39 to 2680 vertices) contained in the Harwell-Boeing sparse matrix collection (*http://math.nist.gov/MatrixMarket/data/-Harwell-Boeing*) [7]. The workstation used in the execution of the simulations with 11 instances of the Harwell-Boeing sparse matrix collection contained an Intel® Core™i3-4005U (CPU 1.8GHz, 3MB Cache, 4GB of main memory DDR3 1333MHz, termed M1 machine; Intel; Santa Clara, CA, United States).

A set of values for the input parameter ν ($1 \leq \nu \leq \lfloor |V|/2 \rfloor$) was assigned in Algorithm 3 for each instance. Then, Algorithm 3 is applied to the corresponding instance and returns the smallest profile found. Three sequential runs were performed for each instance.

Table 1 shows the name of the instances used, their size, and original profile ($profile_0$). This table also shows the smallest ν set in Algorithm 2 to reach the smallest profile. Specifically, Table 1 shows the results of Snay's heuristic with pseudo–peripheral vertices given by two algorithms when applied to 11 instances contained in the Harwell-Boeing sparse matrix collection. In addition, this table shows the minimum value of ν used to reach the smallest profile obtained, the smallest profile obtained by Snay's heuristic with pseudo–peripheral vertices given by the generalized algorithm [Snay(ν)] and the original Snay's algorithm [19]. Additionally, Table 1 shows the execution times (in seconds), $\%_\nu = round\left(\frac{2\cdot\nu}{\text{instance size}} * 100\right)$, and $\%_r = round\left(\frac{profile}{profile_0} * 100\right)$ [a rate of profile reduction (or increase) provided by the heuristics]. The objective is to minimize $\%_r$. Profile numbers in bold face are the best results.

Table 1 shows that the Snay(ν) heuristic provided better profile results in seven of the 11 instances used. Figure 1 shows rates of profile reductions and execution costs of Snay's [19] and Snay(ν) heuristics when applied to 11 instances contained in the Harwell-Boeing sparse matrix collection [7].

3.2 Experiments with instances contained in the University of Florida sparse matrix collection

The workstation described in the previous section was also used in the execution of the simulations with 25 instances (ranging from 20,000 to 153,226 vertices) contained in the University of Florida sparse matrix collection [6]. In addition, the other workstations used in the execution of the simulations with instances of this collection contained an Intel® Core™ i7-4790K (CPU 4.00GHz, 8MB Cache, 12GB of main memory DDR3 1.6GHz, termed M2 machine) and an Intel® Core™ i7-4770 (CPU 3.40GHz, 8MB Cache, 8GB of main memory DDR3 1.333GHz, termed M3 machine; Intel; Santa Clara, CA, United States).

Table 2 shows similar columns to Table 1. Again, a set of values for the input parameter ν [$1 \leq \nu \leq \max(\nu)$; see the $\max(\nu)$ column in Table 2] was assigned in Algorithm 3 for each instance. The ν value associated with the smallest profile obtained in these runs

Table 1: Results of Snay's heuristic [19] with starting vertices given by two algorithms for finding pseudo–peripheral vertices when applied to 11 instances contained in the Harwell-Boeing sparse matrix collection [7].

Instance	Size	$pro-$ $file_0$	Snay(v)					Snay [19]			Structure
			v	$\%_v$	*profile*	$\%_r$	t(s)	*profile*	$\%_r$	t(s)	
BCSPWR01	39	292	2	10	**85**	29	0.0002	**85**	29	0.0003	symmetric
BCSPWR02	49	377	3	12	**133**	35	0.0003	**133**	35	0.0004	
PLAT362	362	45261	20	11	**14084**	31	0.0783	14283	32	0.0279	
DWT_503	503	35914	13	5	**23744**	66	0.0785	23803	66	0.0306	
DWT_592	592	28805	19	6	**19335**	67	0.1067	19360	67	0.0376	
662_BUS	662	45165	30	9	**16164**	36	0.1272	17411	39	0.0225	
NOS3	960	**39101**	48	10	53674	137	0.6324	63546	163	0.0740	
DWT_992	992	262306	78	16	**54024**	21	1.1380	62260	24	0.0741	
DWT_2680	2680	587863	118	9	**344814**	59	14.4354	372966	63	0.6247	
ORSIRR_2	886	211572	113	26	**156640**	74	2.3334	199635	94	0.1073	asymmetric
WEST0989	989	**250490**	96	19	376009	150	0.5156	389834	156	0.2165	

for an instance is stored. Then, 10 sequential runs (using Algorithm 3) were performed for each instance. These executions were performed on the M1 (*qpband*, *crystm03*, *bloweybl*, *mark3jac060sc*), M2 (*rail_20209*, *case39*, *chipcool0*, *invextr1_new*), and M3 (*t60k*, *juba40k*, *bauru5727*, *chem_master1*, *lhr71c*, *2cubes_sphere*, *s3dkt3m2*, *s4dkt3m2*, *tandem_dual*, *pkustk12*, *ASIC_100ks*, *rajat16*, *rajat17*, *rajat18*, *para-4*, *c-64b*, *boyd1*) machines.

Table 2 also provides the computational times of the algorithms for finding pseudo–peripheral vertices [in seconds; see PPV *t(s)* columns] and the renumbering step [in milliseconds; see Ren. *t(ms)* columns] separately. This table shows that the algorithm for finding pseudo-peripheral vertices shows higher computational times than the renumbering heuristic (Algorithm 1). In particular, a large number of renumbering steps (Algorithm 1) does not affect significantly the computing time to find a smaller profile. For example, when applied to the *crystm03* instance, 228 executions of the reordering algorithm (Algorithm 1) were faster (18ms) than the executions of the reordering algorithm (19ms) corresponding to the original Snay heuristic [19].

Figure 2 shows that the rates of profile reductions of the Snay(v) heuristic are similar to the rates of profile reductions of the original Snay's heuristic [19] when applied to 25 instances contained in the SuiteSparse matrix collection [6]. In particular, the Snay(v) heuristic showed much better profile results than Snay's heuristic [19] when applied to the *boyd1* instance. Small $\max(v)$ values were used in these simulations. Numerical experiments with large $\max(v)$ values (see Table 2) may improve the profile results of the Snay(v) at higher execution costs.

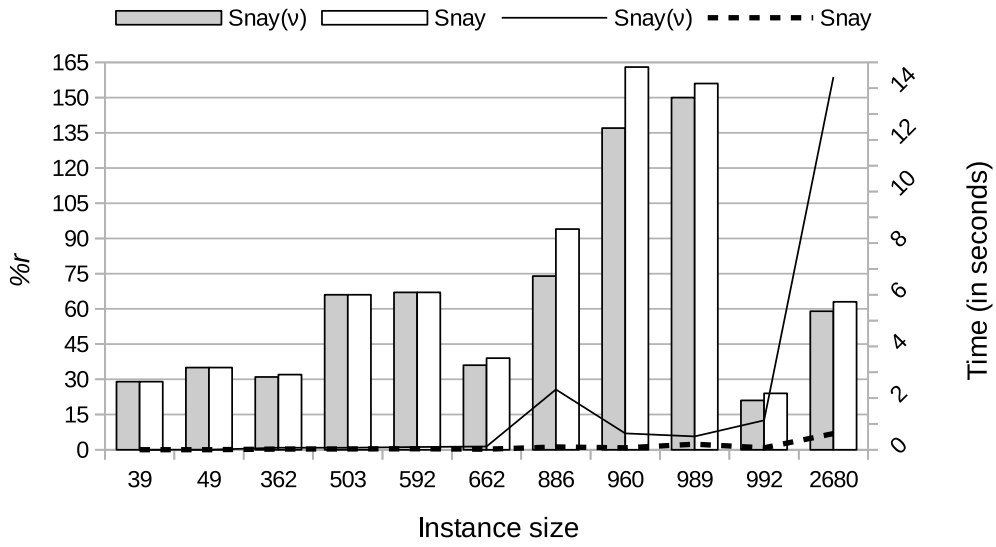

Figure 1: Rates of profile reductions and execution times obtained using Snay's [19] and Snay(ν) heuristics when applied to 11 instances contained in the Harwell-Boeing sparse matrix collection [7].

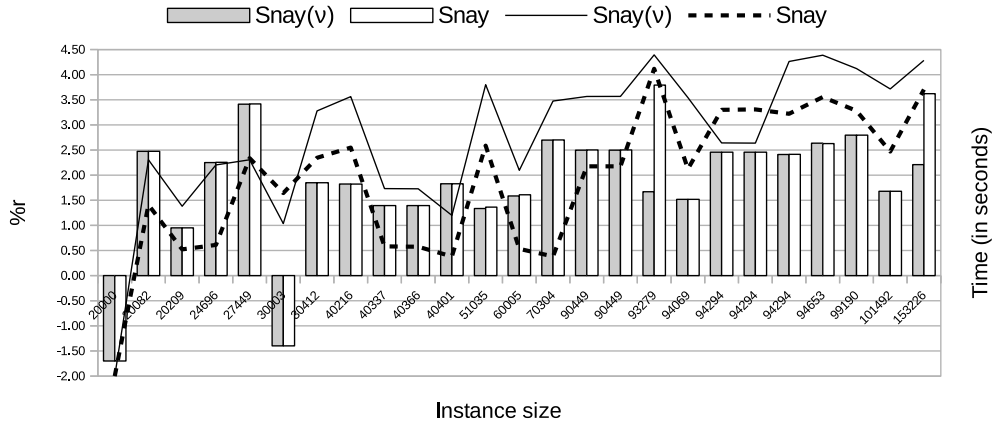

Figure 2: Rates of profile reductions and execution times, in logarithmic scale, obtained using Snay's [19] and Snay(ν) heuristics when applied to 25 instances contained in the SuiteSparse matrix collection [6].

Table 2: Results of Snay's heuristic [19] with starting vertices given by two algorithms for finding pseudo–peripheral vertices when applied to 25 instances contained in the SuiteSparse matrix collection [6].

Instance	Size	Profile	Snay(v)							Snay [19]				struc.
			v	$\%_v$	max (v)	Profile	$\%_r$	PPV *t(s)*	Ren. *t(ms)*	Profile	$\%_r$	PPV *t(s)*	Ren. *t(ms)*	
qpband	20000	75000000	1	0.01	1308	**15000**	0.02	0.01	14	**15000**	0.02	0.01	15	symmetric
rail_20209	20209	60032258	33	0.33	40	**5360990**	8.93	24.07	3	5361203	8.93	3.33	2	
crystm03	24696	**13624695**	114	0.92	124	24273933	178.16	159.78	18	24592527	180.50	4.08	19	
bloweybl	30003	200030005	1	0.01	283	**89990**	0.04	10.85	5	**89990**	0.04	44.00	6	
case39	40216	605563013	36	0.18	58	**403231130**	66.59	3664.18	14	403540017	66.64	353.80	11	
c-64b	51035	932439770	76	0.30	110	**202536321**	21.72	6320.08	15	215711609	23.13	387.43	15	
t60k	60005	18206621	194	0.65	200	**7027923**	38.60	124.88	17	7418045	40.74	3.41	11	
s3dkt3m2	90449	**54981444**	113	0.25	200	172870146	314.42	3686.18	49	174605068	317.57	150.92	40	
s4dkt3m2	90449	**54981444**	113	0.25	201	172861891	314.40	3696.13	49	174605068	317.57	149.90	41	
boyd1	93279	1508966	9	0.19	18	**706100**	46.79	24778.87	34	93785838	6215.24	13121.90	33	
tandem_dual	94069	487466755	129	0.27	211	**160793185**	32.99	3498.89	27	160906545	33.01	133.00	21	
pkustk12	94653	**158088290**	29	0.06	34	685664121	433.72	24419.48	72	695475848	439.93	3580.60	71	
ASIC_100ks	99190	**478531931**	27	0.05	48	2995700549	626.02	13238.65	42	2993583378	625.58	1911.04	26	
2cubes_sphere	101492	483241271	81	0.16	156	**230796772**	47.76	5205.60	55	231132476	47.83	295.09	52	
chipcool0	20082	**23086070**	37	0.37	40	68784302	297.95	202.22	7	68931698	298.59	25.65	7	asymmetric
mark3jac060sc	27449	**11796165**	28	0.20	28	305560702	2590.34	1294.11	14	309135334	2620.64	218.44	14	
invextr1_new	30412	607320732	50	0.33	50	**429707010**	70.75	1899.09	27	429726661	70.76	221.76	28	
juba40k	40337	90823979	54	0.27	119	**22527049**	24.80	54.04	7	22527242	24.80	3.83	6	
bauru5727	40366	90902767	54	0.27	140	**22543833**	24.80	53.54	7	22544026	24.80	3.76	6	
chem_master1	40401	16160800	30	0.15	200	**10906298**	67.49	15.82	10	10907035	67.49	2.41	8	
lhr71c	70304	**465706283**	8	0.02	63	2324138812	499.06	2987.01	80	2342068763	502.91	1685.62	112	
rajat16	94294	**1280040450**	1	0.01	47	3668856633	286.62	438.68	18	3668856633	286.62	1997.30	18	
rajat17	94294	**1280040450**	1	0.01	47	3668856633	286.62	434.67	24	3668856633	286.62	2033.01	26	
rajat18	94294	**1188606664**	45	0.10	54	3069750298	258.26	18331.96	37	3085986625	259.63	1663.86	24	
para-4	153226	**26607087**	20	0.03	30	1028210064	3864.42	19255.92	123	1115392547	4192.09	5072.59	130	

4 CONCLUSIONS

In general, Snay's heuristic with starting vertices given by the generalized algorithm for finding pseudo–peripheral vertices [Snay(v)] achieved better profile results than Snay's heuristic with starting vertices given by the original Snay's algorithm for finding pseudo–peripheral vertices [19], at a higher execution time. On the other hand, among 36 instances used, these algorithms increased the profile of 14 instances. Little (or no) gain can be obtained if the profile of the original instance is small. For example, a previous publication [14] reports that in certain cases several reordering algorithms do not reduce the computational times of the Jacobi-preconditioned conjugate gradient method.

From the results obtained with the simulations performed in this work, it is possible to realize that the selection of 10 pseudo–peripheral vertices for Snay's heuristic [19] is a reasonable choice when the heuristic is applied to very small instances. On the other hand, when the instance is not small, it would be necessary to choose a larger number of pseudo–peripheral vertices to find the smallest profile (that Snay's heuristic can achieve) of an instance. The user can set the v value depending on the class of matrices in context. Specifically, assigning the v parameter up to 26% (e.g., see the *ORSIRR_2* instance extracted from a oil reservoir simulation problem; see Table 1) of the instance size is a reasonable choice to obtain a small profile for small instances. In addition, we recommend to assign the v parameter up to 0.3% of pseudo–peripheral vertices in relation to the instance size when applied to instances larger than 20,000 vertices. This percentage is directly proportional to the execution time of the heuristic, i.e. the user can assign a lower percentage to obtain a lower execution cost when using the Snay(v) heuristic. The reason for this is because the processing times to execute several renumbering steps may not compensate the profile results.

Specifically, this algorithm provided better profile results in half of the 36 instances used. As mentioned, Figure 2 shows that in general the rates of profile reductions provided by the Snay(v) heuristic were not significantly better than the rates of profile reductions yielded by the original Snay's heuristic [19] when applied to instances contained in the SuiteSparse matrix collection [6]. This present computational experiment makes it possible to realize that searching for better profile results may not compensate the computational effort employed. A low-cost algorithm for finding pseudo-peripheral vertices is desirable, mainly if the objective is to reduce execution costs of iterative solvers for linear systems, such as the Generalized minimal residual (GMRES) [18] (e.g. see [2, 5]) and conjugate gradient methods [15, 16] (e.g. see [12]). Specifically, the use of only one pseudo-peripheral vertex (with eccentricity very close to the diameter of the graph) may be a better choice. An example is the George-Liu algorithm [9] that is commonly used in computational experiments in this field (e.g. [8, 2, 5, 11, 12]). In particular, the use of Snay's heuristic with starting vertex given by the George-Liu algorithm [9] showed promising results in preliminary simulations [3] and we plan to extend our numerical experiments.

The computational times of the renumbering step (Algorithm 1) of Snay's heuristic [19] depend mainly on the starting vertex of the vertex labeling (that is, the pseudo–peripheral vertex selected in Algorithm 2). Specifically, the computational times of Snay's heuristic [19] depend primarily

on the number of candidate vertices (to be labeled) at each iteration. We intend to present the complexity analysis of Snay's heuristic [19] in a future work.

We intend to compare the heuristics evaluated in [11, 12] with Snay's heuristic with pseudo–peripheral vertices given by the generalized algorithm to find peripheral vertices proposed in this work, with the parameter ν established after exploratory investigations in the set of instances to be used. In addition, a systematic review of algorithms for finding pseudo–peripheral vertices is a next step in this present work.

ACKNOWLEDGEMENTS

This work was undertaken with the support of the FAPEMIG - Fundação de Amparo à Pesquisa do Estado de Minas Gerais (Minas Gerais Research Support Foundation, Brazil). In addition, we would like to thank the reviewers for their valuable comments and suggestions.

RESUMO. A solução de sistemas de equações lineares, representados por $Ax = b$, é fundamental em diversas aplicações científicas e em engenharia. Ao se reduzir o *profile* da matriz A, pode-se reduzir a ocupação de espaço e o tempo de processamento da resolução de tais sistemas de equações lineares. Neste trabalho, propomos um algoritmo generalizado para encontrar vértices pseudoperiféricos para o algoritmo heurístico de Snay. Baseados em experimentos realizados em 36 instâncias contidas nas bases de matrizes Harwell-Boeing e SuiteSparse, verificou-se que o número de vértices pseudoperiféricos selecionados pelo algoritmo de Snay pode ser adequado para instâncias pequenas, mas é insuficiente para obter resultados razoáveis em instâncias que não são pequenas. Neste artigo, recomendamos a seleção de até 26% (0,3%) de vértices pseudoperiféricos em relação ao tamanho da instância, quando o algoritmo heurístico de Snay é aplicado em instâncias menores que 3.000 (maiores que 20.000) vértices.

Palavras-chave: Redução de profile, matrizes esparsas, algoritmos de reordenação de vértices.

REFERENCES

[1] M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, **182** (2002), 418–477.

[2] M. Benzi, D.B. Szyld & A. van Duin. Orderings for incomplete factorization preconditioning of nonsymmetric problems. *SIAM Journal on Scientific Computing*, **20**(5) (1999), 1652–1670.

[3] J.A.B. Bernardes. *Uma modificação na heurística de Snay para redução do custo computacional do método dos gradientes conjugados*. Master's thesis, Universidade Federal de Lavras (2016).

[4] J.A.B. Bernardes & S.L. Gonzaga de Oliveira. A systematic review of heuristics for profile reduction of symmetric matrices. *Procedia Computer Science*, **51** (2015), 221–230.

[5] J.J. Camata, A.L. Rossa, A.M.P. Valli, L. Catabriga, G.F. Carey & A.L.G.A. Coutinho. Reordering and incomplete preconditioning in serial and parallel adaptive mesh refinement and coarsening flow solutions. *International Journal for Numerical Methods in Fluids*, **69** (2012), 802–823.

[6] T.A. Davis & Y. Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software*, **38**(1) (2011), 1–25.

[7] I.S. Duff, R.G. Grimes & J.G. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, **15**(1) (1989), 1–14.

[8] A. George & J.W. Liu. *Computer solution of large sparse positive definite systems*. Prentice-Hall, Englewood Cliffs (1981).

[9] A. George & J.W.H. Liu. An implementation of a pseudoperipheral node finder. *ACM Transactions on Mathematical Software*, **5**(3) (1979), 284–295.

[10] S.L. Gonzaga de Oliveira & J.A.B. Bernardes. Um algoritmo pseudo-periférico genérico para a heurística de Snay. In XXXVI Brazilian National Congress in Applied and Computational Mathematics (CNMAC), volume 5. Gramado, SBMAC, São Carlos (2017).

[11] S.L. Gonzaga de Oliveira, J.A.B. Bernardes & G.O. Chagas. An evaluation of low-cost heuristics for matrix bandwidth and profile reductions. *Computational & Applied Mathematics*. DOI: 10.1007/s40314-016-0394-9.

[12] S.L. Gonzaga de Oliveira, J.A.B. Bernardes & G.O. Chagas. An evaluation of reordering algorithms to reduce the computational cost of the incomplete Cholesky-conjugate gradient method. *Computational & Applied Mathematics*. DOI: 10.1007/s40314-017-0490-5.

[13] S.L. Gonzaga de Oliveira & G.O. Chagas. *Introdução a heurísticas para redução de largura de banda de matrizes*. SBMAC, São Carlos (2014).

[14] S.L. Gonzaga de Oliveira, G.O. Chagas & J.A.B. Bernardes. An analysis of reordering algorithms to reduce the computational cost of the Jacobi-preconditioned CG solver using high-precision arithmetic. In Proceedings of the International Conference on Computational Science and Its Applications, ICCSA. Lecture Notes in Computer Science book series (LNCS), 10404:3-19 (2017).

[15] M.R. Hestenes & E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, **49**(36) (1952), 409–436.

[16] C. Lanczos. Solutions of systems of linear equations by minimized iterations. *Journal of Research of the National Bureau of Standards*, **49**(1) (1952), 33–53.

[17] Y.X. Lin & J.J. Yuan. Profile minimization problem for matrices and graphs. *Acta Mathematicae Applicatae Sinica*, **10**(1) (1994), 107–122.

[18] Y. Saad & M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, **7** (1986), 856–869.

[19] R.A. Snay. Reducing the profile of sparse symmetric matrices. *Bulletin Geodesique*, **50**(4) (1976), 341–352.