An Experimental Analysis of Heuristics for Profile Reduction

S. L. Gonzaga de Oliveira[1(✉)], C. Osthoff[2], and L. N. Henderson Guedes de Oliveira[3]

[1] Universidade Federal de Lavras, Lavras, MG, Brazil
sanderson@ufla.br
[2] Laboratório Nacional de Computação Científica (LNCC), Petrópolis, RJ, Brazil
osthoff@lncc.br
[3] Universidade do Estado do Rio de Janeiro, Nova Friburgo, RJ, Brazil
nelio@iprj.uerj.br

Abstract. This paper concentrates on low-cost heuristics for profile reduction. Low-cost methods for profile reduction are mainly heuristic in nature and based on graph-theoretic concepts. The contribution of this paper is twofold. Firstly, the paper includes a section involving a numerical examination of the current state-of-art metaheuristic and graph-theoretic methods for matrix profile reduction. With the support of extensive experiments, this paper shows that the metaheuristic-based algorithm is capable of reducing the profile of some matrices where the other algorithms do not perform well, but on average, the profile reduction obtained is similar for these algorithms, whereas the metaheuristic-based algorithm takes seven orders of magnitude more running time. These high execution times make the metaheuristic-based algorithm a noncontender for sparse matrix factorization and related problems. Secondly, this paper experimentally evaluates a hybrid algorithm based on the MPG and NSloan heuristics. This paper also evaluates the new hybrid heuristic for profile reduction when applied to matrices arising from two application areas against the most promising low-cost heuristics for solving the problem. The results obtained on a set of standard benchmark matrices show that the new hybrid heuristic method does not compare favorably with existing low-cost heuristics for profile reduction when applied to large-scale matrices.

Keywords: Profile reduction · Sparse matrices · Graph labelling · Combinatorial optimization · Graph theory · Search methods · Reordering algorithms · Renumbering · Ordering · Graph algorithm · Permutation of sparse matrices

1 Introduction

The solution of linear systems composed of large-scale sparse matrices is one of the most relevant computational kernels in scientific computing. Specifically,

S. Misra et al. (Eds.): ICCSA 2019, LNCS 11619, pp. 25–36, 2019.
https://doi.org/10.1007/978-3-030-24289-3_3

the solution of many real-world and applied problems in science and engineering (e.g., thermal and model reduction problems) reduces to solving large-scale sparse linear systems. The solution of large-scale sparse linear systems is usually a part of the numerical simulation that demands a high computational cost in execution times and memory requirements. Thus, the study to reduce the computational cost of solving linear systems composed of large-scale sparse matrices is a significant topic in numerical mathematics because of its importance in many numerical simulations.

Iterative linear system solvers that handle large-scale sparse matrices tend to suffer from poor memory performance because of the inefficient use of cache if they do not consider the order of how to process the matrix rows and columns. Simulations can generally gain substantial improvements in execution costs by accessing data with an appropriate order (e.g., [2,4,6,9,10]). Thus, the simulation should adequately label the vertices (of the sparse graph corresponding to a sparse matrix) so that data associated with adjacent vertices tend to be stored in nearby memory locations to improve cache hit rates. Therefore, spatial locality (a cache block brings in variables that the near future computation will use) should be considered an essential aspect when designing a new algorithm.

An appropriate vertex numbering is desirable to guarantee that the associated coefficient matrix will have a small profile for the low-cost solution of large and sparse linear systems, and to reduce the memory requirements of a linear system, depending on the data structure used. Thereby, the use of heuristics for profile reduction is a way of designing an application to return a sequence of graph vertices with spatial locality. The matrix profile reduction also favors direct methods for solving linear systems. Reordering rows and columns of sparse matrices also contributes to improving the arithmetic intensity of the sparse matrix-vector multiplication [21], which is the most critical aspect in the kernel of the conjugate gradient method [11,14]. As a consequence, the resulting linear system is much easier to compute than the original linear system, even when the linear system is composed of multiple right-hand side vectors [9]. Thus, heuristics for profile reduction are used to obtain low processing and small storage costs for solving large sparse linear systems.

The profile reduction is also crucial for increasing the effectiveness of data structures to represent large-scale matrices, such as applications that use the skyline data structure [7]. Another area where reordering rows and columns of sparse matrices is of fundamental importance is in serial and parallel adaptive mesh refinement. As the mesh is adapted, sparsity changes dynamically and reordering should be employed [3].

Heuristics for profile reduction belong to a family of renumbering algorithms that place nonzero coefficients of a sparse matrix close to the main diagonal. 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 $profile(A) = \sum_{i=1}^{n} [\beta_i]$ where $\beta_i = i - \min_{1 \leq j \leq i} [j \mid a_{ij} \neq 0]$ and $a_{ii} \neq 0$.

The profile minimization problem is a well known $\mathcal{NP}$-hard [15] computational search problem studied for over a half-century. The reason is that it is related to a vast range of scientific and engineering application areas. Thus, practitioners have proposed a wide variety of heuristic methods for reordering the rows and columns of a sparse matrix to reduce its profile (see [1,10] and references therein). Since this is an intense field of research, practitioners continue to devote efforts to developing heuristics for profile reductions that are capable of reducing the profile of the instances to a considerable extent (e.g., see [17]).

Metaheuristic approaches are common alternatives to graph-theoretical optimization techniques in several fields. Specifically, metaheuristic-based heuristics for profile optimization have been proposed recently (see [17] and references therein). Palubeckis [17] provides a list of applications that require the solution of the profile reduction problem where runtime is not a critical issue. Thus, these applications may apply a metaheuristic algorithm for profile optimization. In the case of a heuristic for profile reduction applied as a preprocessing step while solving linear systems, however, there has been a limited exploration of such techniques mainly because the heuristics for this problem must reach low computational costs. In general, computational costs (time and space) of metaheuristic-based heuristics for profile optimization are impractical when the objective is to accelerate a linear system solver [1,8–10].

This paper focuses on the group of applications based on sparse matrix factorization and related problems so that the experiments conducted here evaluate low-cost (time and space) heuristics for profile reductions against the state-of-the-art metaheuristic-based algorithm for this problem [17]. Apart from substantially reducing the profile, a heuristic must also achieve low computational costs, i.e., it can neither be slow nor show large memory requirements. To provide more specific details, an adequate vertex labeling of a graph corresponding to a matrix contained in a linear system may reduce the computational times of a (possibly preconditioned) linear system solver, such as the conjugate gradient method [6] (as previously mentioned, by improving cache hit rates [2,4]). Additionally, the profile reductions obtained by a reordering algorithm are not directly proportional to the computational time reduction of solving linear systems. Moreover, the objective is to minimize the total computing time of the simulation including the preprocessing time required by the reordering heuristic, at least when only a single linear system is to be solved [10]. Therefore, a vertex labeling algorithm must perform at low-cost [9].

This paper evaluates a modified heuristic against the Sloan's [19], MPG [16], NSloan [13], Sloan-MGPS [18], and Hu-Scott [12] heuristics. The new heuristic for profile reduction is a hybrid algorithm of the MPG [16] and NSloan [13] heuristics. This paper also compares the profile results obtained by these low-cost heuristics with the state-of-the-art metaheuristic-based algorithm for profile reduction [17].

Section 2 describes the heuristics evaluated in this computational experiment and states a modified heuristic for profile reduction. Section 3 describes how this paper conducted the tests in this computational experiment. Section 4 shows the experimental results. Finally, Sect. 5 addresses the conclusions.

2 Related Work

Sloan [19] proposed one of the most important heuristics in this field. His heuristic is still one of the most widely used reordering algorithm for reducing the profile size of sparse matrices (e.g., [1,9,10,13,16,18]). The reason is that it is inexpensive (in terms of execution times and storage costs) and generates quality solutions. Sloan's heuristic [19] employs two weights in its priority scheme in order to label the vertices of the instance: w_1, associated with the distance $d(v, e)$ from the vertex v to a pseudo-peripheral (target end) vertex e that belongs to the last level of the level structure rooted at a starting vertex s, and w_2, associated with the degree of each vertex. The priority function $p(v) = w_1 \cdot d(v, e) - w_2 \cdot (deg(v) + 1)$ employed in Sloan's heuristic [19] presents different scales for both criteria. The value of $deg(v) + 1$ ranges from 1 to $m + 1$ (where $m = \max_{v \in V}[deg(v)]$ is the maximum degree found in the graph $G = (V, E)$), and $d(v, e)$ ranges from 0 (when $v = e$) to the eccentricity $\ell(e)$ (of the target end vertex e).

The MPG [16], NSloan [13], and Sloan-MGPS [18] heuristics are based on Sloan's heuristic [19]. Specifically, the Sloan-MGPS heuristic [18] is essentially the Sloan's heuristic [19] with the starting and target end vertices given by an algorithm named modified GPS [18] instead of using the original Sloan's algorithm for finding these two pseudo-peripheral vertices.

The MPG heuristic [16] employs two max-priority queues: t contains vertices that are candidate vertices to be labeled, and q contains vertices belonging to t and also vertices that can be inserted to t. Similarly to Sloan's heuristic [19] (and its variations), the current degree of a vertex is the number of adjacencies to vertices that neither have been labeled nor belong to q. A main loop performs three steps. First, a vertex v is inserted into q in order to maximize a specific priority function. Second, the current degree $cdeg(v)$ of each vertex $v \in t$ is observed: the algorithm labels a vertex v if $cdeg(v) = 0$, and the algorithm removes from t a vertex v (i.e., $t \leftarrow t - \{v\}$) if $cdeg(v) > 1$. Third, if t is empty, the algorithm inserts into t each vertex $u \in q$ with priority $p_u \geq p_{max}(q) - 1$ where $p_{max}(q)$ returns the maximum priority among the vertices in q. The priority function in the MPG heuristic is $p(v) = d(v, e) - 2 \cdot cdeg(v)$.

Kumfert and Pothen [13] normalized the two criteria used in Sloan's algorithm with the objective of proposing the Normalized Sloan (NSloan) heuristic [13]. These authors used the priority function $p(v) = w_1 \cdot d(v, e) - w_2 \cdot \lfloor d(s, e)/m \rfloor \cdot (deg(v) + 1)$.

Regarding Sloan's [19], NSloan [13], and Sloan-MGPS [18] heuristics, this paper established the two weights as described in the original papers. When the authors suggested more than one pair of values in the original papers, exploratory investigations were performed to determine the pair of values that obtains the best profile results [10]. Thus, the two weights are assigned as $w_1 = 1$ and $w_2 = 2$ for Sloan's and Sloan-MGPS [18] heuristics, and as $w_1 = 2$ and $w_2 = 1$ for the NSloan heuristic [13].

In addition to the four low-cost heuristics for profile reductions selected from reviews of the literature [9,10], this paper evaluates a modified MPG heuristic

in this paper. The new heuristic for profile reduction is essentially a hybrid of the MPG [16] and NSloan [13] heuristics. Specifically, the new heuristic is based on the MPG heuristic [16] in conjunction with the normalized scheme proposed by Kumfert and Pothen [13]. This heuristic uses the priority function $p(v) = d(v,e) - 2 \cdot d(s,e) / \max_{v \in V}[deg(v)] \cdot (cdeg(v))$. This paper refer to this new heuristic as the NMPG heuristic.

The Hu-Scott heuristic [12] is a multilevel algorithm that uses a maximal independent vertex set for coarsening the adjacency graph of the matrix and Sloan-MGPS heuristic [18] on the coarsest graph. This paper also analyzes the results yielded by the state-of-the-art metaheuristic algorithm for profile reduction [17] against five low-cost heuristics. The MSA-VNS heuristic is a hybrid algorithm based on the multi-start simulated annealing (MSA) algorithm along with the Variable Neighborhood Search (VNS) metaheuristic [17].

3 Description of the Tests

This paper divides the experiments into two main parts. The first part of the experiments compares the results of five low-cost heuristics for profile reductions with the results obtained by the MSA-VNS heuristic [17]. Palubeckis [17] compared the results of his heuristic with the results of the previous state-of-the-art metaheuristic algorithms for profile reduction when applied to two groups of matrices: 38 (ranging from 24 to 292 vertices) and 39 matrices (ranging from 307 to 2,680 vertices). The SuiteSparse matrix collection [5] contains these matrices. Since the matrices of the first group are too small for today's standards, this paper uses the 39 matrices of the second group to compare the six heuristics evaluated in this computational experiment.

Palubeckis [17] implemented his heuristic in the C++ programming language, and performed his experiments on a workstation containing an Intel® Core™ 2 Duo CPU running at 3.0 GHz. To provide a reasonable comparison of the running times, we performed the executions of the five low-cost heuristics for profile reductions evaluated here on a workstation containing an Intel Core™ 2 Duo CPU running at 2.3 GHz (Intel; Santa Clara, CA, United States). Although the profile reduction heuristics evaluated here are deterministic algorithms, 10 serial runs for each matrix were carried out to obtain average results to mitigate possible interferences in the execution costs.

This appraisal also employs the C++ programming language to implement five low-cost heuristics for profile reduction (Sloan's, MPG, NSloan, Sloan-MGPS, and NMPG heuristics). The implementations of these five heuristics for profile reductions appraised here employ binary heaps to code the priority queues (although the original Sloan's algorithm [19] used a linked list to code it). A previous publication [10] shows the testing and calibration performed to compare the implementations with the ones used by the original proposers of the four low-cost heuristics (Sloan's [19], MPG [16], NSloan [13], and Sloan-MGPS [18]) to ensure that the codes employed here were comparable to the formerly proposed algorithms.

A second part of the experiments uses 15 real symmetric matrices contained in the SuiteSparse matrix collection [5]. We also used the Hu-Scott heuristic [12], namely the MC73 routine, contained in the HSL [20], in this experiment. We employed the Fortran programming language to use this routine. The workstations used in the execution of the simulations with these 15 matrices contained an Intel® Core™ i7-4770 (CPU 3.40 GHz, 8 MB Cache, 8 GB of main memory DDR3 1.333 GHz) (Intel; Santa Clara, CA, United States). We performed three sequential runs for each large-scale matrix. The profile reduction depends on the choice of the initial ordering, and this paper considers the original ordering given in the instance contained in the SuiteSparse matrix collection [5].

4 Results and Analysis

The first part of the experiments (in Sect. 4.1) compares the profile results of five low-cost heuristics (Sloan's, MPG, NSloan, Sloan-MGPS, and NMPG) with the results of the state-of-the-art metaheuristic algorithm for profile reduction (i.e., the MSA-VNS heuristic [17]) when applied to instances ranging from 307 to 2,680 vertices. These experiments show that the MSA-VNS heuristic yields better profile results than the five other heuristics evaluated do, at much higher execution costs. Thus, the second part of the experiments (in Sect. 4.2) shows the results of six low-cost heuristics for profile reductions (i.e., including the Hu-Scott heuristic) when applied to 15 instances ranging from 19,994 to 1,228,045 vertices [up to 47,851,783 edges (or nonzero coefficients)].

4.1 Comparison of the Results Obtained Using State-of-the-Art Metaheuristic Algorithm Against Five Low-Cost Heuristics

Tables 1 and 2 show the characteristics of the instance (name, size, original profile ($profile_0$)) and the average values of profile obtained when using six heuristics applied to reduce the profile of 39 small matrices. Figure 1 shows that the MSA-VNS heuristic [17] obtains better profile reductions than the five other heuristics evaluated do in this computational experiment. The figure shows that the MSA-VNS heuristic achieves much higher profile rate reduction than the five other heuristics included in this appraisal when applied to some instances (e.g., can445, bcsstk20, can634, dwt869). Additionally, the MSA-VNS heuristic [17] reduced the profile of the gr_30_30 and instances nos3, whereas, in general, the five other low-cost heuristics increased the profiles of these two instances. On the other hand, Fig. 1 shows that in general the profile rate reduction obtained by the six heuristics evaluated are similar.

Figure 2 (in line charts for clarity) shows that the executions costs of the MSA-VNS heuristic is much higher than the five other heuristics for profile reduction evaluated here. The experiments conducted here reveal that the MSA-VNS heuristic [17] achieved its results with a higher processing cost of at least seven magnitudes in relation to the five other heuristics evaluated. For example, Sloan's [19], NSloan [13], Sloan-MGPS [18], MPG [16], and NMPG heuristics computed

Table 1. Results of six heuristics applied to reduce the profile of 26 instances contained in the SuiteSparse matrix collection [5]. Palubeckis [17] obtained the results of the MSA-VNS heuristic (times in seconds) in simulations performed on an Intel® Core™ 2 Duo running at 3.0 GHz [17]. The five other heuristics (times in milliseconds) were executed on a machine containing an Intel® Core™ 2 Duo running at 2.3 GHz. (Continued on Table 2.)

Matrix	Result	$profile_0$ /size	MSA-VNS (s)	SLOAN-MGPS (ms)	MPG (ms)	SLOAN (ms)	NSLOAN (ms)	NMPG (ms)
dwt307	profile	7825	6172	6842	6883	6813	6842	6883
	time (s)	307	266.2	0.017	0.042	0.004	0.011	0.044
dwt310	profile	2696	2630	2661	2657	2658	2641	2666
	time (s)	310	42.5	0.009	0.037	0.004	0.006	0.039
dwt346	profile	8708	5788	6214	6189	6191	6214	6189
	time (s)	346	272.5	0.019	0.056	0.006	0.012	0.059
dwt361	profile	5084	4631	4706	4755	4699	4706	4758
	time (s)	361	181.3	0.012	0.038	0.004	0.008	0.040
plat362	profile	45261	8206	8517	8517	8511	8517	8517
	time (s)	362	276.8	0.019	0.071	0.006	0.012	0.076
lshp406	profile	13224	5955	6282	6203.2	6260	6194	6191
	time (s)	406	114.7	0.017	0.055	0.005	0.011	0.058
dwt419	profile	39726	6073	6794	6714	6836	6794	6714
	time (s)	408	268.3	0.044	0.096	0.008	0.028	0.102
bcsstk06	profile	14691	12829	13771	13870	13691	13771	13870
	time (s)	419	270.6	0.019	0.068	0.006	0.012	0.071
bcspwr05	profile	36248	2608	3849	4414	4095	3862	4803
	time (s)	420	272.4	0.024	0.090	0.007	0.015	0.098
can445	profile	22321	14199	18035	16511	17295	18035	16511
	time (s)	443	276.5	0.016	0.052	0.006	0.011	0.051
nos5	profile	27286	19896	20447	20494	20549	20447	20494
	time (s)	445	275.7	0.050	0.083	0.009	0.031	0.087
bcsstk20	profile	4309	2602	3242	3124	3248	3177	3123
	time (s)	485	291.1	0.013	0.066	0.007	0.008	0.067
dwt492	profile	33790	2805	2866	2958	2856	2861	2938
	time (s)	492	269.1	0.014	0.067	0.007	0.009	0.067
494bus	profile	40975	2592	4327	4738	4486	3882	4667
	time (s)	494	287.5	0.017	0.046	0.005	0.010	0.045
dwt503	profile	35914	11428	14259	14164	13608	14259	14164
	time (s)	503	753.7	0.035	0.081	0.008	0.022	0.087
dwt512	profile	6018	3820	4150	4167	4322	4150	4169
	time (s)	512	696	0.022	0.085	0.011	0.013	0.088
lshp577	profile	22816	10035	10625	10499	10607	10488	10469
	time (s)	577	457.8	0.028	0.078	0.007	0.018	0.082
dwt592	profile	28805	8816	9697	9580	9871	9177	9437
	time (s)	592	576.7	0.027	0.074	0.008	0.017	0.078
dwt607	profile	30008	12431	13856	14470	13825	13856	14470
	time (s)	607	739.9	0.042	0.114	0.012	0.026	0.119
can634	profile	68586	25170	38033	33896	34609	38033	33896
	time (s)	634	767.1	0.099	0.122	0.014	0.062	0.129
662bus	profile	45165	5994	9180	10023	9701	8587.1	10339
	time (s)	662	762.4	0.032	0.075	0.006	0.021	0.071
nos6	profile	16229	9095	9095.000	9095.000	9095.000	9095.000	9095
	time (s)	675	600.2	0.024	0.060	0.006	0.015	0.063
685bus	profile	28621	5993	8547.700	9674.000	8560.000	8912.000	8716
	time (s)	685	769.9	0.028	0.078	0.008	0.020	0.076
can715	profile	72423	20280	30423.000	29810.000	30695.000	30423.000	29810
	time (s)	715	1697.1	0.078	0.115	0.011	0.049	0.122
nos7	profile	53144	34110	34698.000	35202.000	34473.000	34690.000	35321
	time (s)	729	1380.7	0.084	0.127	0.011	0.052	0.132
dwt758	profile	23113	6364	6967.000	6534.000	6983.000	6490.000	6535
	time (s)	758	1528.1	0.023	0.090	0.010	0.015	0.095

Table 2. Results of six heuristics applied to reduce the profile of 13 instances contained in the SuiteSparse matrix collection. Palubeckis [17] obtained the results of the MSA-VNS heuristic (times in seconds) in simulations performed on an Intel® Core™ 2 Duo running at 3.0 GHz [17]. The five other heuristics (times in milliseconds) were executed on a machine containing an Intel® Core™ 2 Duo running at 2.3 GHz. (Continued from Table 1.)

Matrix	Result	$profile_0$ /size	MSA-VNS (s)	SLOAN-MGPS (ms)	MPG (ms)	SLOAN (ms)	NSLOAN (ms)	NMPG (ms)
lshp778	profile	36284	15652	16630.000	16424.000	16592.000	16399.000	16373
	time (s)	778	1364.3	0.046	0.118	0.009	0.027	0.123
bcsstk19	profile	74051	7240	8613.000	8450.000	9250.000	20837.000	22146
	time (s)	817	1651.8	0.024	0.126	0.011	0.042	0.122
dwt869	profile	19528	11806	14672.000	14493.000	14918.000	14916.000	15521
	time (s)	869	1609	0.040	0.100	0.011	0.027	0.106
dwt878	profile	26055	16903	18967.000	18227.000	18783.000	19522.000	18237
	time (s)	878	1606.3	0.047	0.095	0.011	0.032	0.100
gr_30_30	profile	26970	23836	28764.000	26970.000	28664.000	26538.000	26970
	time (s)	900	1616.5	0.070	0.098	0.012	0.041	0.104
dwt918	profile	108355	15007	16332.000	15768.000	16852.000	17754.000	19199
	time (s)	918	1734.3	0.046	0.144	0.015	0.032	0.147
nos2	profile	3180	1907	1907.000	1910.000	1907.000	1907.000	1910
	time (s)	957	1783.7	0.017	0.095	0.011	0.011	0.099
nos3	profile	39101	35916	39673.000	40195.000	39457.000	39673.000	40195
	time (s)	960	1739.4	0.067	0.216	0.021	0.042	0.231
dwt992	profile	262306	31620	33013.200	33376.000	32960.000	33012.000	33376
	time (s)	992	1678.4	0.053	0.193	0.017	0.034	0.208
dwt1005	profile	121070	29631	33980.000	33413.000	33737.000	33980.000	33413
	time (s)	1005	1694.4	0.094	0.162	0.016	0.058	0.171
dwt1007	profile	25786	18880	22852.000	21321.900	22882.000	20184.000	21424
	time (s)	1007	1613.5	0.060	0.114	0.015	0.034	0.121
dwt1242	profile	110188	31756	35611.000	36102.000	36377.000	33937.000	39951
	time (s)	1242	1727.9	0.090	0.167	0.016	0.056	0.166
dwt2680	profile	587863	82575	87621.000	88130.000	87242.000	90123.000	91656
	time (s)	2680	3221.4	0.230	0.389	0.041	0.164	0.406

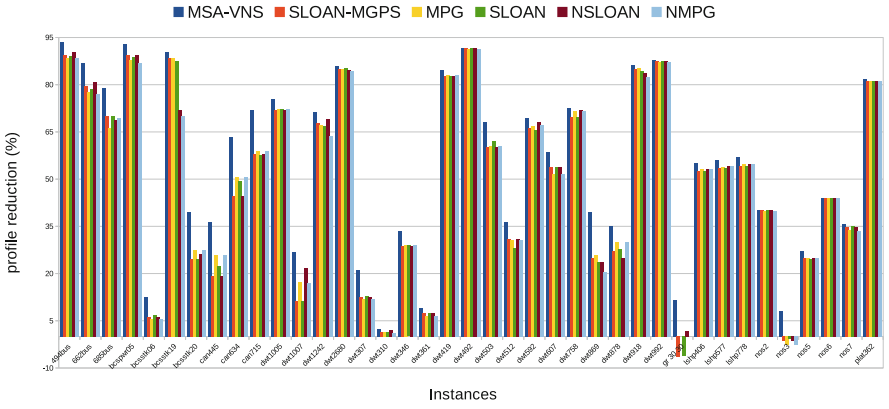

Fig. 1. Results of six heuristics applied to reduce the profile of 39 instances contained in the SuiteSparse matrix collection [5].

the instance dwt2680 (i.e., the largest instance contained in this dataset) in 0.04, 0.16, 0.23, 0.39, and 0.41 ms (in simulations performed on an Intel® Core™ 2 Duo running at 2.3 GHz), respectively, whereas the MSA-VNS heuristic computed this instance in more than 3221 s (in simulations performed on an Intel® Core™ 2 Duo running at 3.0 GHz). As an example, Sloan's heuristic [19] computes an instance composed of 1,228,045 vertices in two seconds.

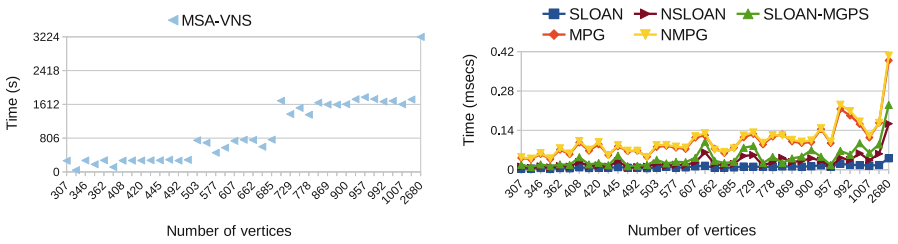

Fig. 2. Execution times of six heuristics applied to reduce the profile of 39 instances contained in the SuiteSparse matrix collection. Palubeckis [17] obtained the results of the MSA-VNS heuristic (times in seconds (s)) in simulations performed on an Intel® Core™ 2 Duo running at 3.0 GHz [17]. The simulations with the five other heuristics (times in milliseconds (msecs)) were performed on an Intel® Core™ 2 Duo running at 2.3 GHz.

On 34 of the 39 matrices contained in the dataset used here, the MSA-VNS heuristic [17] delivers smaller profiles than any other previously known profile reduction algorithm does. While this result is impressive, the MSA-VNS heuristic is very slow, taking approximately $2n^2$ milliseconds for a matrix of order n. It is not practical for larger instances.

The quality and time of the MSA-VNS algorithm may depend on the setting of its parameters. However, we considered the results provided by its original publication [17], which are expected to have the best parameters. Since low-cost heuristics for profile reductions [10] yield in general reasonable profile results at much lower costs than the state-of-the-art metaheuristic algorithm for profile reduction [17], Sect. 4.2 concentrates on evaluating six low-cost heuristics for profile reductions when applied to larger matrices (up to 1,228,045 vertices).

4.2 Results of Six Low-Cost Heuristics for Profile Reductions

This section describes the results of six low-cost heuristics for profile reductions applied to a dataset comprised of 15 symmetric matrices contained in the SuiteSparse matrix collection. Table 3 shows the matrix's name and size, the value of the initial profile of the instance (profile_0), and the average values of profile obtained when using each heuristic. The same table also highlights the best profile results.

Table 3 shows that the Hu-Scott heuristic yielded the highest number of best profile results when applied to symmetric instances originating from thermal (in

Table 3. Results of six heuristics applied to reduce the profile of 15 symmetric matrices contained in the SuiteSparse matrix collection [5] originating from thermal (four matrices) and model reduction (11 matrices) problems.

matrix	size	$profile_0$	Hu-Scott	NMPG	MPG	NSloan	Sloan-MGPS	Sloan
thermal2	1,228,045	53,657,335,080	587,662,601	586,063,633	584,643,264	587,246,981	609,179,419	593,981,740
thermo mech_dM	204,316	10,671,293,780	28,549,159	30,796,587	30,825,327	30,939,618	32,276,780	31,984,980
thermo mech_TC	102,158	2,667,823,445	13,477,995	15,346,513	15,412,663	15,446,340	16,109,532	16,021,130
thermo mech_TK	102,158	2,667,823,445	13,372,590	15,398,294	15,504,258	15,493,278	16,109,532	15,906,569
Nr. best results		0	3	0	1	0	0	0
bone010	986,703	8,846,266,758	187,539,359	1,471,105,979	2,231,282,192	1,282,743,704	1,687,120,778	1,571,237,233
boneS10	914,898	6,345,023,025	2,147,483,647	3,742,417,527	3,742,417,527	3,823,790,163	3,952,972,833	3,890,992,689
boneS01	127,224	331,330,356	245,645,928	320,120,227	302,253,322	330,301,515	320,695,596	308,162,745
filter3D	106,437	260,719,523	68,760,290	86,376,100	87,199,353	95,192,166	98,931,960	97,975,438
rail_79841	79,841	551,148,483	11,968,229	8,910,916	12,851,179	9,830,923	14,398,928	13,732,868
t3dh	79,171	250,243,367	154,218,807	160,795,113	158,470,211	160,944,302	164,226,809	160,693,484
gas_sensor	66,917	69,391,231	63,046,780	65,044,792	72,856,073	68,858,703	69,736,475	71,821,911
t3dl	20,360	14,726,265	14,392,289	14,146,144	14,081,961	44,263,732	14,754,577	14,654,390
rail_20209	20,209	60,032,258	1,295,632	1,172,744	1,744,663	1,214,743	1,562,130	1,632,981
LF10000	19,998	49,990	69,988	49,990	49,990	49,990	49,990	49,990
LFAT5000	19,994	84,958	54,978	34,984	34,984	34,984	34,984	34,984
Nr. best results		1	6	4	3	2	2	2

three instances) and model reduction (in six instances) problems. On the other hand, the same table shows that the MPG heuristic delivered the best profile result when applied to the highest matrix (*thermal2*) used in this study.

Figure 3 (in line charts for clarity) shows the execution times of the six low-cost heuristics for profile reductions evaluated. The Sloan and NSloan heuristics obtained lower execution times than the four other heuristics evaluated. On average, the Hu-Scott, NMPG, and MPG heuristics obtained similar execution times.

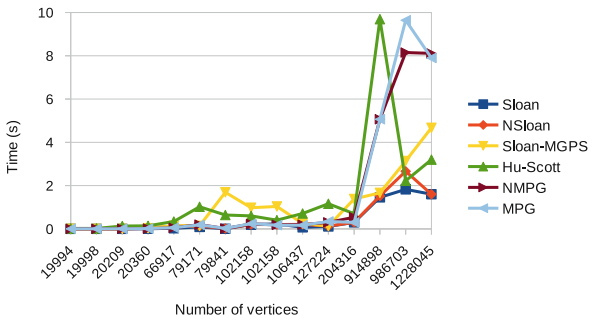

Fig. 3. Execution times (in seconds) of six heuristics for profile reductions in simulations using 15 symmetric instances contained in the SuiteSparse matrix collection.

5 Conclusions

This computational experiment evaluated five existing low-cost heuristics for profile reduction (variants of Sloan's algorithm [19]) along with a new hybrid heuristic based on the MPG [16] and NSloan [13] heuristics, termed NMPG

heuristic. This computational experiment also compared the results provided by a metaheuristic algorithm based on simulated annealing and variable neighborhood search metaheuristics [17] with low-cost heuristics for profile reduction. As expected, the simulations conducted in this paper show that the state-of-the-art metaheuristic algorithm for profile reduction [17] reaches better profile results than low-cost heuristics do at much higher execution costs. Currently, no metaheuristic-based heuristic for profile reduction exists in the literature that can successfully reduce the profile of large-scale matrices at a reasonable amount of time. In this field, scientific and engineering applications apply low-cost heuristics for profile reductions when the computational time is a critical subject. Consequently, in the case of instances of rather large dimensions, a practical option is to use low-cost heuristics for obtaining satisfactory-quality solutions for problem instances defined by such matrices. The experiments conducted here were carried out on several matrices arising from different application domains. The results show that the metaheuristic algorithm provides better profile results, but takes a lot more time than graph-theoretic heuristics for profile reduction.

This paper also applied six low-cost heuristics to 15 matrices arising from thermal and model reduction problems. Numerical experiments show that the NMPG heuristic provides, in general, further worthwhile gains when compared with classical heuristics in this field (Sloan's [19], MPG [16], NSloan [13], Sloan-MGPS [18]). However, the NMPG heuristic for profile reduction did not perform better than Hu-Scott heuristic did when applied to matrices arising from the application domains used here.

The next step in this work is to evaluate low-cost heuristics for profile reductions implemented using parallel libraries (e.g., OpenMP, Galois, and Message Passing Interface systems) and in GPU-accelerated computing. Similarly, regarding massively parallel computing, an evaluation of these heuristics implemented within the Intel® Math Kernel Library running on Intel® Xeon® Many Integrated Core Architecture and Scalable processors is another future step of this investigation.

References

1. Bernardes, J.A.B., Gonzaga de Oliveira, S.L.: A systematic review of heuristics for profile reduction of symmetric matrices. Procedia Comput. Sci. **51**, 221–230 (2015). https://doi.org/10.1016/j.procs.2015.05.231
2. Burgess, D.A., Giles, M.: Renumbering unstructured grids to improve the performance of codes on hierarchial memory machines. Adv. Eng. Softw. **28**(3), 189–201 (1997)
3. Camata, J.J., Rossa, A.L., Valli, A.M.P., Catabriga, L., Carey, G.F., Coutinho, A.L.G.A.: Reordering and incomplete preconditioning in serial and parallel adaptive mesh refinement and coarsening flow solutions. Int. J. Numer. Meth. Fluids **69**(4), 802–823 (2012)
4. Das, R., Mavriplis, D.J., Saltz, J.H., Gupta, S.K., Ponnusamy, R.: The design and implementation of a parallel unstructured Euler solver using software primitives. AIAA J. **32**(3), 489–496 (1994)

5. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Trans. Math. Softw. **38**(1), 1–25 (2011)
6. Duff, I.S., Meurant, G.A.: The effect of ordering on preconditioned conjugate gradients. BIT Numer. Math. **29**(4), 635–657 (1989)
7. Felippa, C.A.: Solution of linear equations with skyline-stored symmetric matrix. Comput. Struct. **5**(1), 13–29 (1975)
8. Gonzaga de Oliveira, S.L., Abreu, A.A.A.M., Robaina, D.T., Kischnhevsky, M.: An evaluation of four reordering algorithms to reduce the computational cost of the Jacobi-preconditioned conjugate gradient method using high-precision arithmetic. Int. J. Bus. Intell. Data Min. **12**(2), 190–209 (2017)
9. Gonzaga de Oliveira, S.L., Bernardes, J.A.B., Chagas, G.O.: An evaluation of reordering algorithms to reduce the computational cost of the incomplete Cholesky-conjugate gradient method. Comput. Appl. Math. **37**(3), 2965–3004 (2018)
10. Gonzaga de Oliveira, S.L., Bernardes, J.A.B., Chagas, G.O.: An evaluation of low-cost heuristics for matrix bandwidth and profile reductions. Computat. Appl. Math. **37**(1), 641–674 (2018). (First Online: 5 July 2016)
11. Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stan. **49**(36), 409–436 (1952)
12. Hu, Y., Scott, J.A.: A multilevel algorithm for wavefront reduction. SIAM J. Sci. Comput. **23**(4), 1352–1375 (2001)
13. Kumfert, G., Pothen, A.: Two improved algorithms for envelope and wavefront reduction. BIT Numer. Math. **37**(3), 559–590 (1997)
14. Lanczos, C.: Solutions of systems of linear equations by minimized iterations. J. Res. Natl. Bur. Stan. **49**(1), 33–53 (1952)
15. Lin, Y.X., Yuan, J.J.: Profile minimization problem for matrices and graphs. Acta Mathematicae Applicatae Sinica **10**(1), 107–122 (1994)
16. Medeiros, S.R.P., Pimenta, P.M., Goldenberg, P.: Algorithm for profile and wavefront reduction of sparse matrices with a symmetric structure. Eng. Comput. **10**(3), 257–266 (1993)
17. Palubeckis, G.: A variable neighborhood search and simulated annealing hybrid for the profile minimization problem. Comput. Oper. Res. **87**, 83–97 (2017)
18. Reid, J.K., Scott, J.A.: Ordering symmetric sparse matrices for small profile and wavefront. Int. J. Numer. Meth. Eng. **45**(12), 1737–1755 (1999)
19. Sloan, S.W.: A Fortran program for profile and wavefront reduction. Int. J. Numer. Meth. Eng. **28**(11), 2651–2679 (1989)
20. STFC. The Science and Technology Facilities Council. HSL. A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk. Accessed Dec 2015
21. Williams, S., Waterman, A., Patterson, D.: Roofline: an insightful visual performance model for multicore architectures. Commun. ACM **52**(4), 65–76 (2009)