Ten algorithms for the Homogeneous Set Sandwich Problem[*]

Vinícius Gusmão Pereira de Sá[1]

[1]COPPE, Universidade Federal do Rio de Janeiro

vgusmao@cos.ufrj.br

Abstract. *This thesis presents state-of-the-art results in two areas of increasing relevance over recent years: graph sandwich problems and randomized algorithms. Sandwich problems are generalizations of recognition problems where the input graph is provided with a set of optional edges. The* Homogeneous Set Sandwich Problem*, subject of this thesis, is one of the few polynomial-time sandwich problems known to date. Following the first two algorithms (not ours) published for it (one of which proved incorrect in [Sá 2003]), we introduce a series of eight faster algorithms, among deterministic and randomized ones. The text shows a comprehensive range of continuously refined techniques, culminating at the establishment of the problem's currently accepted upper bound.*

1. Introduction

A graph $G_2(V, E_2)$ is a *supergraph* of a graph $G_1(V, E_1)$ if $E_2 \supseteq E_1$. A graph $G_S(V, E_S)$ is said to be a *sandwich graph* of $G_1(V, E_1)$, $G_2(V, E_2)$ iff $E_1 \subseteq E_S \subseteq E_2$. A *sandwich problem* for property Π asks whether a given pair of graphs admits a sandwich graph that presents property Π [Golumbic et al. 1995]. Originally arisen from Computational Biology applications, they have encompassed many research areas ever since. (Figure 1 shows a typical input for a sandwich problem.)

A *homogeneous set* of a graph $G(V, E)$ is a set $H \subset V$, $2 \leq |H| < |V|$, such that for all $v \in V \setminus H$, either $(v, h) \in E$ for all $h \in H$ or $(v, h) \notin E$ for all $h \in H$. Homogeneous sets are useful, among other things, in graph decomposition procedures, specially in the perfect graphs field [Lovász 1972]. The *Homogeneous Set Sandwich Problem* (HSSP) poses the question: is there a sandwich graph $G_S(V, E_S)$ of (G_1, G_2) containing a homogeneous set? If so, such a homogeneous set is called a *sandwich homogeneous set* (SHS) of (G_1, G_2). (Figure 2 illustrates a SHS.)

Sandwich problems for most graph properties considered thus far have proved to be NP-complete [Golumbic and Wassermann 1998, Kaplan and Shamir 1999, Dantas et al. 2004]. The HSSP, which belongs to the seemingly small class of polynomial sandwich problems, has attracted attention [Cerioli et al. 1998, Tang et al. 2001, Habib et al. 2003] as a challenging problem, since its known algorithms are considerably less efficient than the existing linear-time algorithms for homogeneous sets recognition [McConnell and Spinrad 1994, Dahlhaus et al. 2001].

The first polynomial-time HSSP algorithm was published in [Cerioli et al. 1998]. A few years later, [Tang et al. 2001] tailored an interesting algorithm which would have largely diminished HSSP's upper bound. Nevertheless, that algorithm was hopelessly

[*]D.Sc. thesis submitted to COPPE/UFRJ. Advisor: Celina M. H. de Figueiredo, UFRJ. Sponsored by CAPES. The full text can be found at **http://br.geocities.com/vinguzman/tese.pdf**.

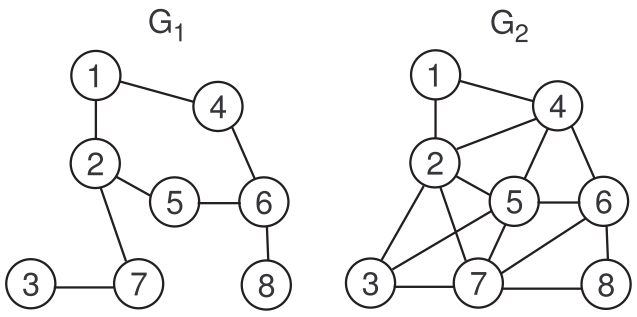

Figure 1. Typical instance for graph sandwich problems

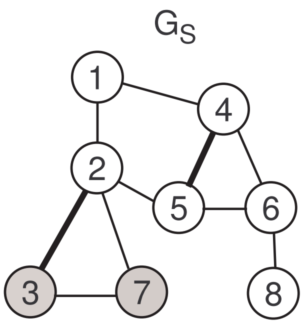

Figure 2. $\{$3,7$\}$: SHS of pair of graphs from Figure 1

incorrect [Sá 2003, Sá and Figueiredo 2005]. Consequently, the most efficient algorithm for the HSSP would have turned back to be Cerioli *et al.*'s $O(n^4)$ algorithm[1]. However, we present a sequence of six faster algorithms, including the one which sets the problem's currently accepted upper bound at $O(nm \log n)$. Furthermore, we present two efficient $O(n^3)$ randomized algorithms[2] for the HSSP (which happen to be quite didactic, as well, as experience has shown).

2. Bias Vertices

A *bias vertex* of a set $F \subset V$ is a vertex $b \in V \setminus F$ such that, for some $v_i, v_j \in F$, there hold $(b, v_i) \in E_1$ and $(b, v_j) \notin E_2$. The set $B(F)$ comprising all bias vertices of F defines the *bias set* of F. (Figure 3 illustrates this concept, the dashed edge meaning a non-edge.) It is easy to show that set $H \subset V, |H| \geq 2$, is a SHS of (G_1, G_2) iff $B(H)$ is empty.

Clearly, if b is a bias vertex of set H, then b is also a bias vertex of every set H' containing H such that $b \notin H'$. This simple fact gave rise to a procedure which we refer to as *bias envelopment*. Starting from a given initial SHS *candidate* $H_1 \subset V$, it computes $H_q = H_{q-1} \cup B(H_{q-1})$ until either (i) $B(H_q) = \varnothing$, whereupon H_q is a SHS and a *yes* answer ensues, or (ii) $H_q \cup B(H_q) = V$, when the resulting *no* answer means there is no SHS containing H_1. The bias envelopment procedure, which runs in $O(n^2)$ time, is enough to show that the HSSP is polynomial. Indeed, the idea of Cerioli *et al.*'s so-called

[1]We denote $n = |V|$, $m = \min\{|E_1|, |\overline{E_2}|\}$, $M = \max\{|E_1|, |\overline{E_2}|\}$ and $\triangle =$ maximum degree in G_2.

[2]*Randomized* algorithms employ randomly generated numbers to make choices during their computation. Recent years have witnessed a growth in the number of published randomized algorithms for a number of problems, since, as compared to deterministic algorithms, they are often simpler or faster—or both.

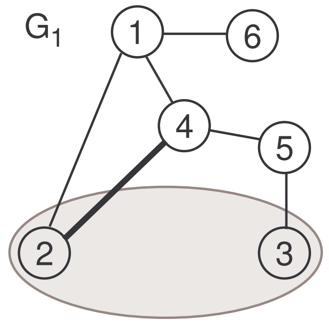

Figure 3. Vertex 4: bias vertex of $\{$2,3$\}$

Exhaustive Envelopments (EE) algorithm was that of submitting vertex pairs to the bias envelopment procedure until a SHS has been found or all $O(n^2)$ two-vertex candidates have otherwise been exhausted.

This text is a brief introduction to the ideas underlying the algorithms that followed. Obviously, it does not delve into details due to space constraints. It also skips the counterproof of Tang *et al.*'s $O(n^2\triangle)$ algorithm—referred to as the *Bias Graph Components* (BGC) algorithm—and the $O(mM)$ first algorithm of ours—namely, the *Two-Phase* (2-P) algorithm—, both in [Sá 2003] and therefore prior to the present thesis.

3. The Balanced Subsets algorithm

We introduce a variation for the bias envelopment procedure: the *incomplete* bias envelopment. Its input comprises not only a pair $G_1(V, E_1), G_2(V, E_2)$ and a candidate H_1, but also a *stop parameter* $k \leq n$. Whenever the size of the candidate has become greater than k, the envelopment stops with a *no* answer. Running in $O(nk)$ time, it states whether H_1 is contained in any SHS's *with k vertices or less*.

When the *Balanced Subsets* (BS) algorithm starts, it partitions all n input vertices into $\lceil \sqrt{n} \rceil$ disjoint subsets C_i of size $O(\sqrt{n})$ each. Then all pairs of vertices will be submitted to bias envelopment in two distinct moments: first, the candidates will be pairs consisting of vertices from the same subset C_i; only then, all remaining pairs are made candidates. The point is, if all bias envelopments in the first moment fail to find a SHS, then the input instance has no SHS's with vertices from a same subset C_i, hence the maximum size of SHS's that still have to be looked for is $\lceil \sqrt{n} \rceil$—and incomplete bias envelopments with stop parameter $k = \lceil \sqrt{n} \rceil$ can be safely employed thenceforth.

Interestingly, the two big groups of bias envelopments demand the same $O(n^3\sqrt{n})$ time[3], which is also the overall worst-case time complexity of the BS algorithm.

4. The Monte Carlo HSSP algorithm

A *yes-biased Monte Carlo* algorithm for a decision problem is one which answers *yes* with probability at least $p = 1 - \epsilon$ if the correct answer is *yes*, and *always* answers *no* when the correct answer is *no*. In other words, though it may fail to find one, all its *yes* answers come with a valid certificate.

[3]That is why this approach is generally referred to as the *balancing technique*.

Let us suppose the HSSP input has a SHS H with h vertices or more. It is not difficult to reach the following expression for the probability p_t (which stands for p as a function of t) that, among t randomly chosen vertex pairs, at least one of them belongs to a SHS of the input instance: $p_t \geq 1 - \{1 - [h(h-1)]/[n(n-1)]\}^t$.

If, instead of calculating the probability p_t, we fix p_t at some desired value $p = 1 - \epsilon$, we will be able to obtain the minimum integer value of h_t (which stands for h as a function of t) that satisfies the aforementioned inequality. It follows that h_t is such that the execution of t independent bias envelopments on randomly chosen pairs suffices to find a SHS with probability at least p, in case there is some SHS with h_t vertices or more:

$$h_t = \left\lfloor \frac{1 + \sqrt{1 + 4(n^2 - n)(1 - (1-p)^{1/t})}}{2} \right\rfloor .$$

With a yes-biased Monte Carlo algorithm in sight, we must be able to find a SHS with probability p in case *any* SHS exists, regardless of its size. As h_t decreases with the growth of t, a natural question is: how many random pairs have to be submitted to bias envelopment in order to achieve that? Of course, it is the minimum integer t' such that $h_{t'} = 2$. Determining t' comes from the equation above right away and we have

$$t' = \frac{\ln(1-p)}{\ln\left(1 - \frac{2}{n(n-1)}\right)} = \Theta(n^2).$$

Now, since $\Theta(n^2)$ bias envelopments would have to be performed and the time complexity of each one of them is $O(n^2)$, an algorithm as such would apparently run in worst-case $O(n^4)$ time—which is all the most undesirable, for there are *deterministic* algorithms more efficient than that!

However, we show that, in order to find a SHS with probability p (under the hypothesis that there exists one with h_t vertices or more), the t-th bias envelopment can be an *incomplete* one with stop parameter $k = h_{t-1}$. Well, two are the possibilities regarding the input: (i) there is a SHS with more than h_{t-1} vertices; or (ii) there is no SHS with more than h_{t-1} vertices. If (i) is true, then no more than $t-1$ bias envelopments would even be necessary (by the definition of h_t). If (ii) is the case, then bias envelopments (in particular, the t-th) do not need to look for SHS's bigger than h_{t-1} (by hypothesis). In other words, the length of the incomplete envelopments decrease with h_t.

That is the basis of our Monte Carlo (MC) HSSP algorithm, whose complexity analysis, a bit trickier than the algorithm itself, shows it runs in worst-case $O(n^3)$ time.

5. The Harmonic Series algorithm

Still another algorithm for the HSSP, named *Harmonic Series* (*HS*) after its odd complexity analysis, will run bias envelopments on the input vertex pairs employing the knowledge accrued during earlier envelopments to speed up later ones. But it will achieve that in a faster, peculiar way.

The HS algorithm proceeds as follows: the $n(n-1)/2$ vertex pairs will be submitted to bias envelopment in $\lfloor n/2 \rfloor$ turns with n pairs each. The first turn submits to bias envelopment all pairs of vertices whose indexes are one unit apart, i.e. $(\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_n, v_1\})$, the second turn submits pairs of vertices whose indexes are two units apart, i.e. $(\{v_1, v_3\}, \{v_2, v_4\}, \ldots, \{v_n, v_2\})$, and so on.

The point is, by the time the n bias envelopments of the t-th turn take place, it is already known that no SHS exists containing a pair of vertices $\{v_i, v_j\}$ such that $i - j (\mod n) < t$. This knowledge bounds the size of SHS's to be searched for during this turn at $\lfloor n/t \rfloor$. Consequently, all bias envelopments in the t-th turn can be interrupted by the time the size of the candidate has exceeded the $\lfloor n/t \rfloor$ threshold, hence incomplete bias envelopments with stop parameter $k = \lfloor n/t \rfloor$ are used.

Given the above, it is not difficult to foresee that the sum of all $O(n)$ turns' running times will give us the famous harmonic progression—which explains the $\log$ in the $O(n^3 \log n)$ bound for the overall complexity of the HS algorithm.

6. The Growing Cliques algorithm

It is clear that, in all HSSP algorithms based on the bias envelopment procedure, the tighter the known upper bound for the size of searched-for SHS's at a certain point, the earlier all remaining envelopments will be allowed to stop. In this sense, the development of an optimum envelopment-based algorithm has to cope with the question: what is the *best* order in which vertex pairs may be submitted to bias envelopment?

Two vertices x, y are said to be *enemies* (of one another) if there is no SHS for that input instance containing both x and y. The *enemies graph* of pair $G_1(V, E_1), G_2(V, E_2)$ is a graph $G_N(V, E_N)$ such that $(x, y) \in E_N$ implies x and y are enemies.

Lemma 1. *Given an enemies graph G_N of pair (G_1, G_2), the cardinality of any SHS of that pair is less than or equal to the cardinality of the maximum independent set*[4] *of G_N.*

Given Lemma 1 and the fact that unsuccessful bias envelopments reveal new enmity relationships, our former question reads: starting from an initially empty enemies graph G_N and adding edge (x, y) to it for each pair $\{x, y\}$ unsuccessfully bias-enveloped, what is the envelopment order that keeps the size of the maximum independent set of G_N as small as possible throughout all successive edge-additions?

It is not difficult to see that there is no match to a strategy in which disjoint cliques of increasing size are progressively assembled in an initially empty enemies graph, so that the size of the minimum clique cover[5] is always as small as possible.

And that is precisely what the *Growing Cliques* (*GC*) algorithm does: during the first *turn* of bias envelopments, G_N's current maximal cliques of size 1 are joint together, pairwisely, assembling $\lfloor n/2 \rfloor$ cliques of size 2. Figure 4(a) shows the enemies graph after the first GC turn. The maximum independent set of G_N currently has $O(n/2)$ vertices. The second turn of Bias Envelopments joins together cliques of size 2, pairwisely, ending up with $O(n/4)$ cliques of size 4, and so on. (Figures 4(b) and 4(c) show the enemies graph, respectively, at the end of the second and third turns of bias envelopments during the GC algorithm.) This process goes on until either a bias envelopment has answered *yes*

[4] An *independent set* of a graph G is a vertex set where no two vertices are adjacent in G.

[5] We recall that a *clique cover* is a collection of maximal cliques such that each and every vertex of the graph belongs to precisely one clique. Clearly, the number of maximal cliques in the minimum clique cover is an upper bound for the size of the maximum independent set.

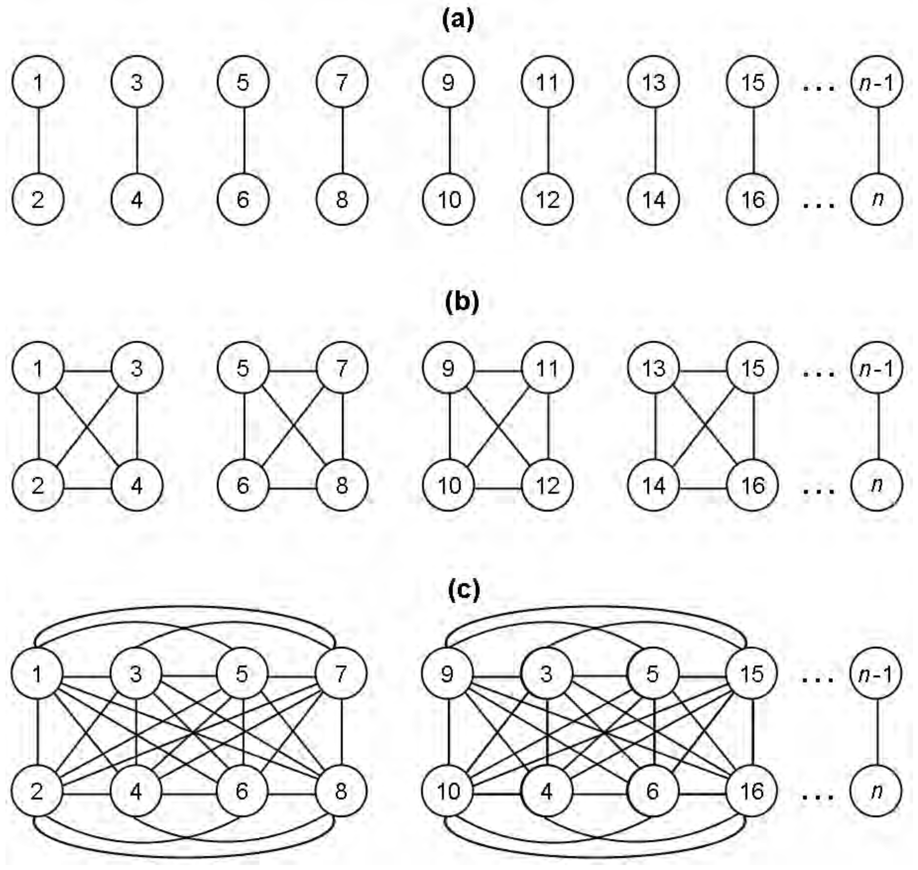

Figure 4. Enemies graph after the first three turns of the GC algorithm

or all vertices in G_N have been linked to one another in a big, single clique—yielding a *no* answer for the HSSP instance in hand.

The concept of enemies also gave rise to both a fast $O(n^3)$ randomized Las Vegas (LV) algorithm[6] and an $O\left(n^3 \log \frac{m}{n}\right)$ deterministic algorithm called *Quick Fill* (QF). Though we do not show them here, they are thoroughly depicted and analyzed in the full text [Sá 2006], where we refer the reader to.

7. The Pair Completion algorithm

The *bias graph* $G_B(V_B, E_B)$ of a pair of graphs $G_1(V, E_1), G_2(V, E_2)$ is a digraph with vertex set $V_B = \{[x, y] \mid x, y \in V, x \neq y\}$. There are two outgoing edges from vertex $[x, y]$ to vertices $[x, b]$ and $[y, b]$ in G_B iff b is a bias vertex of set $\{x, y\} \subset V$. Vertices $[x, y]$ and $[y, x]$ in G_B are the same.

We write $L(X)$ to designate the subset of vertices $v \in V$ which appear in the label of some node in subgraph $X \subseteq G_B$, referring to it as the *labeling set* of X and to its elements as X's *labeling vertices*. In other words, $L(X) = \{v \mid [v, z] \in X, \text{ for some } z\}$. A subgraph $X \subseteq G_B$ is said to be *pair-closed* iff $x, y \in L(X)$ implies $[x, y] \in X$.

[6]A *Las Vegas* randomized algorithm is such that its answer is *always* correct, but its running time is a random variable and evaluated as such in terms of its expectancy.

	Alg.	Time complexity	References
1	EE	$O(n^4)$	[Cerioli et al. 1998]
2	BGC	$O(n^2\triangle)$ [incorrect]	[Tang et al. 2001, Sá and Figueiredo 2005]
3	2-P	$O(mM)$	[Sá 2003, Sá et al. 2006a]
4	BS	$O(n^{3,5})$	[Sá et al. 2004, Sá et al. 2006a]
5	MC	$O(n^3)$ [rand.]	[Sá et al. 2003, Sá et al. 2004, Sá et al. 2006a]
6	HS	$O(n^3 \log n)$	[Sá et al. 2004, Sá et al. 2006a]
7	GC	$O(n^3 \log n)$	[Sá et al. 2006a]
8	LV	$O(n^3)$ [rand.]	[Sá et al. 2006a]
9	QF	$O(n^3 \log \frac{m}{n})$	[Sá et al. 2006a]
10	PC	$O(nm \log n)$	[Sá et al. 2006b]

Figure 5. Table of algorithms

Theorem 2. [Sá 2006, Sá et al. 2006b] *A set $H \subset V$, $|H| \geq 2$, is a SHS of (G_1, G_2) iff it is the labeling set of a pair-closed sink*[7] *in that instance's bias graph.*

The *Pair Completion* (PC) starts by generating the bias graph of the input instance and determining its end strongly connected components (ESCC)[8]. Then, for each ESCC S, it regards set $L = L(S)$ as a potential SHS. Now, for each pair $\{x, y\}$ in L, it checks whether $[x, y]$ reaches, in G_B, an ESCC *other than* S. If that is the case, then one edge is added from S to $[x, y]$—and S no longer constitutes an ESCC, whereupon the algorithm restarts the process from another ESCC or stops with a *no* answer, if the bias graph has become strongly connected itself[9]. If that is *not* the case, then no edges are added—and the algorithm just puts into L all labeling vertices of the out-neighbors of $[x, y]$ in G_B (in case they are not in L yet). If all pairs of vertices in L are investigated without the addition of any new edge, the algorithm will have found the pair-closed sink $S' = \{[u, v] \in V_B \mid u, v \in L\}$ and, by Theorem 2, will stop with a *yes* answer.

The overall time complexity of the Pair Completion algorithm is proved to be $O(nm \log n)$, showing the algorithm's sensibility to the number of edges of G_1 and the number of non-edges in G_2—which is certainly appropriate for such a problem that is invariant under taking the complements of the input graphs.

8. Overview of algorithms and publications

Figure 5 summarizes all HSSP algorithms known to date, their time-complexities and the publications where they can be found. Except for the first three algorithms, all others were developed and published during the research for the present thesis[10].

[7] A digraph's *sink* is an induced subgraph with no outgoing edges.

[8] A strongly connected component (SCC) is a maximal induced subgraph S where, for all $x, y \in S$, there is a path from x to y. An ESCC is a SCC which is a sink. It is easy to show that a sink either is an ESCC itself or it contains an ESCC.

[9] A strongly connected digraph does not contain any sinks.

[10] The Pair Completion algorithm in *Information Processing Letters* and the *Algorithmica* paper were accepted for publication during the research period and published shortly thereafter.

References

Cerioli, M. R., Everett, H., Figueiredo, C. M. H., and Klein, S. (1998). The homogeneous set sandwich problem. *Inf. Proc. Letters*, 67:31–35.

Dahlhaus, E., Gustedt, J., and McConnell, R. M. (2001). Efficient and practical algorithms for sequential modular decomposition. *Journal of Algorithms*, 41:360–387.

Dantas, S., Faria, L., and Figueiredo, C. M. H. (2004). On decision and optimization (k,l)-graph sandwich problems. *Disc. Appl. Math.*, 143:155–165.

Golumbic, M. C., Kaplan, H., and Shamir, R. (1995). Graph sandwich problems. *Journal of Algorithms*, 19:449–473.

Golumbic, M. C. and Wassermann, A. (1998). Complexity and algorithms for graph and hypergraph sandwich problems. *Graphs Combin.*, 14:223–239.

Habib, M., Lebhar, E., and Paul, C. (2003). A note on finding all homogeneous set sandwiches. *Inf. Proc. Letters*, 87:147–151.

Kaplan, H. and Shamir, R. (1999). Bounded degree interval sandwich problems. *Algorithmica*, 24:96–104.

Lovász, L. (1972). Normal hypergraphs and the perfect graph conjecture. *Disc. Math.*, 2:253–267.

McConnell, R. M. and Spinrad, J. (1994). Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proc. of the 5th ACM-SIAM Symposium on Disc. Alg.*, pages 536–545.

Sá, V. G. P. (2003). O problema-sanduíche para conjuntos homogêneos em grafos. Master's thesis, Univ. Federal do Rio de Janeiro.

Sá, V. G. P., Figueiredo, C. M. H. and Fonseca, G. D. (2003). A fast Monte Carlo algorithm for the Homogeneous Set Sandwich Problem. In *Proc. of Mathematical Programming in Rio: a Conference in Honour of Nelson Maculan*, pages 35–39.

Sá, V. G. P., Figueiredo, C. M. H., Fonseca, G. D., and Spinrad, J. (2004). Faster deterministic and randomized algorithms on the homogeneous set sandwich problem. In *3rd Workshop on Efficient and Experimental Algorithms*, volume 3059 of *Lecture Notes Comput. Sci.*, pages 243–252. Springer-Verlag.

Sá, V. G. P and Figueiredo, C. M. H. (2005). A note on the Homogeneous Set Sandwich Problem. *Inf. Proc. Letters*, 93:75–81.

Sá, V. G. P. (2006). *Dez algoritmos para o Problema-Sanduíche do Conjunto Homogêneo*. PhD thesis, Univ. Federal do Rio de Janeiro. URL: `http://br.geocities.com/vinguzman/tese.pdf`.

Sá, V. G. P., Fonseca, G. D., Figueiredo, C. M. H., and Spinrad, J. (2006a). Algorithms for the Homogeneous Set Sandwich Problem. *Algorithmica*, 46:149–180.

Sá, V. G. P, Bornstein, C. F. and Figueiredo, C. M. H. (2006b). The Pair Completion algorithm for the Homogeneous Set Sandwich Problem. *Inf. Proc. Letters*, 98:87–91.

Tang, S., Yeh, F., and Wang, Y. (2001). An efficient algorithm for solving the homogeneous set sandwich problem. *Inf. Proc. Letters*, 77:17–22.