Appendices A, B, and C

for the paper "A prediction model for ranking branch-and-bound procedures for the resource-constrained project scheduling problem" published in the *European Journal of Operational Research.*

Weikang Guo[*], Mario Vanhoucke[†] and José Coelho[‡]

Abstract

This appendix contains 3 appendices A, B and C mentioned in the paper "A prediction model for ranking branch-and-bound procedures for the resource-constrained project scheduling problem" published in the *European Journal of Operational Research.*
(doi: 10.1016/j.ejor.2022.08.042)

1 Appendix A: Technical appendix

1.1 Matching procedures in the CLB framework

The branch-and-bound procedure using a composite lower bound strategy (further abbreviated as the CLB) presented in Coelho and Vanhoucke (2018) combines most of the well-performing components proposed in the academic literature. Figure 1 graphically displays the four components of the CLB, and this abbreviation will be further used to match the existing branch-and-bound procedures into the CLB framework. Among them, three search strategies are used, namely upper bound strategy (U), minimum lower bound strategy (L), or dual bound strategy (D). Three branching schemes the activity start time branching (A), parallel branching (P), or serial branching (S) are considered. Moreover, four versions of the branching orders are implemented, i.e. the best lower bound (B), minimal time window (M), random branching order (R), or the activity ID order (A). As explained earlier in the paper, various composite lower bound strategies (CLB) are introduced, i.e. CLB0 (0), CLB4 (4), CLB8 (8), or CLB12 (12).

Search strategy --	Branching scheme --	Branching order --	Composite lower bounds
(U, L or D)	(A, P or S)	(B, M, R or A)	(0, 4, 8 or 12)

Figure 1: The four components of the CLB procedure of Coelho and Vanhoucke (2018)

[*]Ghent University, Tweekerkenstraat 2, 9000 Gent (Belgium)

[†]Ghent University, Tweekerkenstraat 2, 9000 Gent (Belgium), Vlerick Business School, Reep 1, 9000 Gent (Belgium) and University College London, 1 Canada Square, London E14 5AA (United Kingdom).

[‡]Ghent University, Tweekerkenstraat 2, 9000 Gent (Belgium), INESC - Technology and Science, Porto (Portugal), and Universidade Aberta, Rua da Escola Politécnica, 147, 1269-001, Lisbon (Portugal).

Table A: Literature review on branch-and-bound

	References	Tree strategy				Search strategy			Branching scheme				Branching order				(Composite) lower bounds															
		Depth-first	Best-first	Breadth-first	Hybrid	U	L	D	A	P	S	FR2DP(2)	B	M	R	A	cp	rc(3)	cc	cs	pm0	pm1	pm2	np0	np1	np2	ip0	ip1	pr	ct	tp	LB2(4)
A	Coelho and Vanhoucke (2018)	✓				✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
	Klein and Scholl (1999)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
B	Patterson and Huber (1974)	✓				✓	✓	✓	✓				-	-	-	-	✓	✓														
	Stinson et al. (1978)		✓			✓				✓(1)c			✓				✓	✓		✓												
	Talbot and Patterson (1978)	✓				✓			✓							✓	✓															
	Christofides et al. (1987)	✓				✓				✓(1)c						✓	✓										✓					
	Bell and Park (1990)		✓			✓				✓(1)c			✓(1)d				✓															
	Demeulemeester and Herroelen (1992)	✓				✓				✓						✓	✓			✓												
	Mingozzi et al. (1998)	✓				✓				✓						✓(1)f	✓							✓								
	Demeulemeester and Herroelen (1997)	✓	✓		✓	✓				✓						✓	✓							✓								
	Brucker et al. (1998)	✓				✓						✓	✓				✓															✓
	Nazareth et al. (1999)		✓	✓		✓				✓						✓	✓															
	Dorndorf et al. (2000)	✓(1)a				✓			✓(1)b					✓(1)e			✓															
	Sprecher (2000)	✓				✓					✓					✓	✓							✓								
C	Current work					✓	✓		✓	✓	✓		✓			✓	✓	✓														
						✓	✓		✓	✓	✓		✓			✓	✓	✓	✓	✓				✓			✓					
						✓	✓		✓	✓	✓		✓			✓	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓				
						✓	✓		✓	✓	✓		✓			✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

	References	Match the procedures with our configurations		Comments
		Abbreviation(5)	# Configuration	
A	Coelho and Vanhoucke (2018)	[U,L] × [A,P,S] × [B,A] × [0,4,8,12]	48	
	Klein and Scholl (1999)	-	-	This is the list of lower bounds implemented in our composite B&B.
B	Patterson and Huber (1974)	UAA0, LAA0	2	This procedure is solved by a zero-one formulation. The "dual bound" search strategy is not used in CLB. Since no branching order is given, the activity ID order is selected.
	Stinson et al. (1978)	LPB0, LPB4*	2	The " best-first" tree strategy with upper bound strategy (U) is replaced by the "depth-first" tree strategy with minimum lower bound strategy (L). (cf. Section 1.2)
	Talbot and Patterson (1978)	UAA0	1	
	Christofides et al. (1987)	UPA0, UPA4*	2	
	Bell and Park (1990)	LPB0	1	The best-first tree strategy with upper bound strategy is replaced by the depth-first tree strategy with minimum lower bound strategy. (cf. Section 1.2)
	Demeulemeester and Herroelen (1992)	UPA0, UPA4*	2	
	Mingozzi et al. (1998)	UPA0, UPA4*	2	
	Demeulemeester and Herroelen (1997)	UPA0, UPA4*	2	The "best-first" and "breadth-first" tree strategies are not used in CLB.
	Brucker et al. (1998)	-	0	This approach cannot be matched to the CLB configurations, since FR2DP and LB2 are not in the CLB framework.
	Nazareth et al. (1999)	LPA0	1	The "breadth-first" tree strategy is not used in our study. The "best-first" tree strategy with U is replaced by the "depth-first" tree strategy with L. (cf. Section 1.2)
	Dorndorf et al. (2000)	-	0	This approach differs too much from the CLB configurations that a match could not be made.
	Sprecher (2000)	USA0, USA4*	2	
C	Current work	2 times twelve configurations with LB = CLB0 = cp	12	For calculating LB: [L,U] × [A,P,S] × [B,A] × [cp] For calculating UB: [L,U] × [A,P,S] × [B,A] × [cp]
		2 times twelve configurations with LB = CLB4	12	For calculating LB: [L,U] × [A,P,S] × [B,A] × [CLB4] For calculating UB: [L,U] × [A,P,S] × [B,A] × [CLB4]
		2 times twelve configurations with LB = CLB8	12	For calculating LB: [L,U] × [A,P,S] × [B,A] × [CLB8] For calculating UB: [L,U] × [A,P,S] × [B,A] × [CLB8]
		2 times twelve configurations with LB = CLB12	12	For calculating LB: [L,U] × [A,P,S] × [B,A] × [CLB12] For calculating UB: [L,U] × [A,P,S] × [B,A] × [CLB12]

(1) A slightly adapted version is used (details are given in column "Comments").
- (a) Depth-first: This depth-first tree strategy is implemented as a bi-directional search.
- (b) Activity start time (A): While the other activity start time schemes select nodes with the earliest possible start this, this procedure generates one child node by the earliest start time, and the second node selects an activity to be delayed.
- (c) Parallel branching (P): The parallel branching scheme used in these studies don't rely on minimal delaying alternatives.
- (d) Best lower bound (B): In case the lower bounds are the same, this procedure also uses the highest resource violating time as a tie-breaker.
- (e) Minimal time window (M): This procedure selects the nodes according to the earliest start time (and uses the minimal time window as a tie-breaker).
- (f) Activity ID order (A): This procedure select the node with the highest number of scheduled activities (instead of the lowest activity ID).

(2) FR2DP: The procedure first checks all possible activity pairs to find the pairs with flexibility relation (FR) and then branches on the selected pairs by transforming this flexibility relation either into a disjunction (D) (by introducing the resource constraint) or by placing them in parallel (P).
(3) The resource capacity lower bound rc is only used in a pre-processing phase, and not as a lower bound calculation during the search.
(4) The LB2 lower bound is proposed in Mingozzi et al. (1998) and is not used in other branch-and-bound procedures.
(5) In column "Abbreviation" , * means that not all lower bounds are used in a specific configuration.

Table A displays all the existing branch-and-bound procedures from the literature, classified into three main row blocks. The rows "A" contain the CLB procedure as well as the lower bounds study (Klein and Scholl, 1999) using as a composite lower bound strategy in the CLB framework. Combining all possible components of the CLB procedure results in 48 different configurations. Each row in "B" contains an existing branch-and-bound procedure which will be matched to one or more of the 48 CLB configurations. Finally, the rows in "C" contain more details about the

48 strategies we used in our experiments. The top part of the table shows the similarity of each branch-and-bound procedure with the four components of our study (each time a ✓ is shown, the original procedure is mapped to one or more of our components). The lower part of the table provides additional information about the choices we have made when the similarity was not crystal clear. In this table, two branching orders that are not used in our current study are also given: Minimal time window (M) and Random branching order (R). More details can be found in Coelho and Vanhoucke (2018).

Matching existing branch-and-bound procedures into the CLB framework requires some adaptations and decisions to make, which will briefly be discussed along the following lines.

- **Tree strategy:** The third column of the table displays the tree strategy of the procedure which can be either depth-first, best-first, breadth-first, or a hybrid approach. Since only the depth-first tree strategy has been implemented in the CLB procedure, some procedures cannot be matched into the CLB framework. However, for three of the four procedures that rely on a best-first tree strategy, the tree strategy has been transformed into a depth-first tree strategy to fit into the CLB framework, and this will be explained in Section 1.2.
- **Composite lower bound strategy:** The last component of Figure 1 refers to the composition of lower bounds and can be either 1, 4, 8 or 12. The rows in C show the composition of the four composite lower bound strategies and are copied from the original study of Coelho and Vanhoucke (2018). Most existing branch-and-bound procedures only use the critical-path based lower bound, which corresponds to CLB0, except for a few extended procedures. These extended procedures make use of a wider set of lower bounds and resemble the CLB4 strategy, but nevertheless do not incorporate all lower bounds of the CLB4 strategy. For this reason, we added an asterisk in the column "Abbreviation" to highlight that it resembles, but is not identical to the CLB4 approach.
- **Adaptations:** Not every branch-and-bound procedure perfectly fits into the CLB framework due to little differences between the original branch-and-bound procedure and the specific CLB implementation. Each time the superscript (1) is used in the body of the table, a footnote is displayed to show that the component is slightly different from the components used in the CLB approach.

The column "Abbreviation" finally shows the match of each branch-and-bound procedure with the CLB framework using the components of Figure 1 and, whenever necessary, some comments are added to clarify some choices we have made.

1.2 Depth-first and best-first

We explained earlier that the best-first tree strategy is not implemented in the CLB procedure, and is therefore replaced by the depth-first tree strategy for existing procedures. This section illustrates why *"best-first tree strategy* with upper bound strategy (U)" can be approximated by the *"depth-first tree strategy* with minimum lower bound strategy (L)" using an illustrative example branch-and-bound tree.

It should not be very difficult to see the similarity, and both approaches will be illustrated on an example tree of Figure 2.

The depth-first tree strategy (with minimum lower bound strategy) always expands the nodes on the deepest un-expanded level first. In case multiple nodes at the same level, the node with the lowest LB is selected first. The minimum lower bound search strategy assumes that an artificial upper bound value UB = m is known, and iteratively increases m until a feasible and hence optimal solution is found. More specifically, in a first minimum lower bound strategy run with UB $= m$, only nodes with LB that are not greater than m can be expanded. In the next run, the UB is set to $m + 1$, the procedure continues the search from this updated UB and identifies whether the node can be expanded by examining its LB. Consequently, the order of the nodes expanded in the complete search will be equal to A, B, C, I, J, O. Details of this order can be found in Table B in which it is assumed that the artificial UB $= 10$ and the search continues until UB $= 14$.

For the best-first tree strategy (with upper bound strategy), the nodes are expanded the lowest LB first, and an ordered list of nodes (OPEN) is created which consists of nodes that have been identified but not yet examined. The order of the node in the OPEN list is ranked by their LB values, i.e. the node with the lowest LB is placed the first. A second list (CLOSED) is also dynamically created along the search and consists of nodes which have been expanded. The construction of the OPEN and CLOSED lists is given in Table C. The table shows that the order of the CLOSED list is either A, B, I, J, C, O (in case node "C" is expanded) or A, B, I, J, O (in case node "O" is expanded), which both resemble the node order of the depth-first tree strategy.

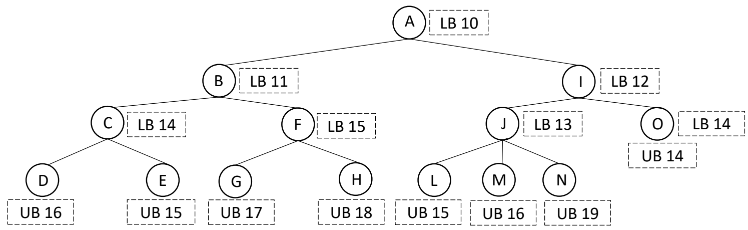

Figure 2: An example tree

Table B: The depth-first with minimum lower bound strategy (L)

The process for the depth-first with L			
Search	Artificial UB	The list of the expanded nodes	Explanation
Search 1 (Initial)	10	A	For this first search, assume that there exists a UB = LB, in this case, UB = 10, and the process terminates at node “A”. Since this tree strategy expands the deepest unexpanded node first, node “B” is newly identified, while the LB of node “B” is greater than UB, it will not be expanded further. Similarly, nodes “C”, “F” and “I” will not be expanded. After this search, no feasible solution is found, the artificial UB is increased by a one-time value (i.e. 1) and the search starts again. Note that in other problems, the time unit can be a different scale, which results in different artificial UBs.
Search 2	=10+1=11	A, B	After expanding “A”, node “B” is newly identified, and its LB (11) is examined to be not greater than UB, this node is expanded. While the LB of node “C” is examined to be greater than UB, this node will not be expanded further. Similarly, nodes “F” and “I” will not be expanded. Since no feasible solution is found, the search starts again by increasing the UB.
Search 3	=11+1=12	A, B, I	After expanding “A” and “B”, node “C” is newly identified, and its LB (14) is examined to be greater than UB, this node will not be expanded. Similarly, nodes “F” will not be expanded. While the LB (12) of node “I” is examined to be not greater than UB, this node is expanded further. The LBs of nodes “J” and “O” are examined in turn to be greater than UB, they will not be expanded. After this search, no feasible solution is found, and the search starts again.
Search 4	=12+1=13	A, B, I, J	After expanding “A”, “B”, the LB (14) of node “C” is examined to be greater than UB, this node will not be expanded. Similarly, nodes “F” will not be expanded. After expanding “I”, the LB (13) of node “J” is examined to be not greater than UB, it will be expanded. While nodes “O” will not be expanded. The search starts again.
Search 5	=13+1=14	A, B, C, I, J, O	After expanding “A” and “B”, the LB of “C” is examined to be not greater than UB, this node is expanded. Then, nodes “D” and “E” are examined and correspond to feasible solutions, and their UBs are worse than 14, which is not acceptable. Then, node “F” is examined, and its LB is greater than UB, it will not be expanded. After expanding “I”, the LB (13) node “J” is examined to be smaller than UB, it will be expanded. Then nodes “L”, “M” and “N” are examined, Similarly, their UBs are also worse than 14, which is not acceptable. After that, node “O” is examined, and its LB equals UB, this feasible solution is also an optimal solution. The search process terminates.

Table C: The best-first with upper bound strategy (U)

The process for the best-first with U				
Expansion		CLOSED	OPEN	Explanation
Initial		Empty	A	Expansion starts from the single node “A” in the OPEN.
Expansion 1		A	B, I	After expanding “A”, it is removed from “OPEN” to “CLOSED”. Then, its children nodes “B” and “I” are newly identified and added to “OPEN”. Since “B” has the smallest LB in all nodes in the “OPEN”, it is the first in the “OPEN” and will be further expanded.
Expansion 2		A, B	I, C, F	After expanding “B”, it is removed from “OPEN” to “CLOSED”. Then, its children nodes “C” and “F” are newly identified and added to “OPEN”. Since “I” has the smallest LB in all nodes in the “OPEN”, it is the first in the “OPEN” and will be further expanded.
Expansion 3		A, B, I	J, C, O, F	After expanding “I”, it is removed from “OPEN” to “CLOSED”. Then, its children nodes “J” and “O” are newly identified and added to “OPEN”. Since “J” has the smallest LB, it is first in the “OPEN” and will be further expanded.
Expansion 4		A, B, I, J	C, O, F, L, M, N	After expanding “J”, it is removed from “OPEN” to “CLOSED”. Then, its children nodes “L”, “M” and “N” are newly identified and added to “OPEN”. Since both nodes “C” and “O” have the smallest LB, either “C” or “O” will be further expanded.
If “C” is expanded	Expansion 5^{C}	A, B, I, J, C	O, F, L, E, D, M, N	After expanding “C”, it is removed from “OPEN” to “CLOSED”. Then, its children nodes “D” and “E” are newly identified and added to “OPEN”. Since the “O” has the smallest LB, it is the first in the “OPEN” and will be further expanded.
	Expansion 6	A, B, I, J, C, O	F, L, E, D, M, N	After expanding “O”, it is removed from “OPEN” to “CLOSED”, and this node is a leaf node with no children. Since the LB of “O” is equal to UB, this feasible solution is also optimal. The search process terminates.
If “O” is expanded	Expansion 5^{O}	A, B, I, J, O	C, L, M, N	After expanding “O”, it is removed from “OPEN” to “CLOSED”. This node is a leaf node with no children. Since the LB of “O” is equal to UB, this feasible solution is optimal. The search process terminates.

2 Appendix B: 48 possible combinations of all components

Table A: Correspondence between configurations and various combinations of components.

Abbreviation	Search strategy	Branching scheme	Branching order	Composite lower bound
	L/U	A/P/S	A/B	CLB0 (0)/CLB4 (4)/CLB8 (8)/CLB12 (12)
UAB0	U	A	B	0
LAB0	L	A	B	0
USB0	U	S	B	0
LSB0	L	S	B	0
UPB0	U	P	B	0
LPB0	L	P	B	0
UAB4	U	A	B	4
LAB4	L	A	B	4
USB4	U	S	B	4
LSB4	L	S	B	4
UPB4	U	P	B	4
LPB4	L	P	B	4
UAB8	U	A	B	8
LAB8	L	A	B	8
USB8	U	S	B	8
LSB8	L	S	B	8
UPB8	U	P	B	8
LPB8	L	P	B	8
UAB12	U	A	B	12
LAB12	L	A	B	12
USB12	U	S	B	12
LSB12	L	S	B	12
UPB12	U	P	B	12
LPB12	L	P	B	12
UAA0	U	A	A	0
LAA0	L	A	A	0
USA0	U	S	A	0
LSA0	L	S	A	0
UPA0	U	P	A	0
LPA0	L	P	A	0
UAA4	U	A	A	4
LAA4	L	A	A	4
USA4	U	S	A	4
LSA4	L	S	A	4
UPA4	U	P	A	4
LPA4	L	P	A	4
UAA8	U	A	A	8
LAA8	L	A	A	8
USA8	U	S	A	8
LSA8	L	S	A	8
UPA8	U	P	A	8
LPA8	L	P	A	8
UAA12	U	A	A	12
LAA12	L	A	A	12
USA12	U	S	A	12
LSA12	L	S	A	12
UPA12	U	P	A	12
LPA12	L	P	A	12

3 Appendix C: Lehmer Code

The following is an example, borrowed from Li et al. (2017) to illustrate the process of the Lehmer code. A permutation $\sigma_{x_i} = (\sigma(y_{i1}), \cdots, \sigma(y_{iQ})) \in \mathfrak{S}_Q$ may be uniquely represented via its Lehmer code, i.e. a word of the form $c_{\sigma_{x_i}} \in \mathcal{C}_Q \triangleq \{0\} \times [\![0, 1]\!] \times [\![0, 2]\!] \times \cdots \times [\![0, Q-1]\!]$, where for any configuration y_{ij} (j = 1,..., Q).

$$c_{\sigma_{x_i}}(y_{ij}) = \#\{y_{ik} : y_{ik}<y_{ij}, \sigma_{x_i}(y_{ik})>\sigma_{x_i}(y_{ij})\}$$

The coordinate $c_{\sigma_{x_i}}(y_{ij})$ is thus the number of elements y_{ik} with index smaller than y_{ij} that are ranked higher than y_{ij} in the permutation σ_{x_i}. and for any finite set C, $\#C$ denotes its cardinality. By default, $c_{\sigma_{x_i}}(y_{i1}) = 0$ and is typically omitted. Consider the following example, which shows the canonical set of items (configurations) e, a permutation σ_{x_i}, and the corresponding Lehmer code $c_{\sigma_{x_i}}$:

e (index)	$y_{i1}(1)$	y_{i2} (2)	y_{i3} (3)	y_{i4} (4)	y_{i5} (5)	y_{i6} (6)	y_{i7} (7)	y_{i8} (8)	y_{i9} (9)
σ_{x_i}	2	1	4	5	7	3	6	9	8
$c_{\sigma_{x_i}}$	0	1	0	0	0	**3**	1	0	1

In this example, the total number of elements is 9. For instance, the 6^{th} digit of the Lehmer code $c_{\sigma_{x_i}} = 3$ because in the permutation σ_{x_i}, there are 3 elements (4, 5, and 7) that appear to the left of the 6^{th} element (i.e. with a smaller index number) but are ranked higher than it. Moreover, its coordinates are decoupled, for this reason, the decoding step it trivial. More details can be found in Li et al. (2017).

References

Bell, C. E. and Park, K. (1990). Solving resource-constrained project scheduling problems by a* search. *Naval Research Logistics (NRL)*, 37(1):61–84.

Brucker, P., Knust, S., Schoo, A., and Thiele, O. (1998). A branch and bound algorithm for the resource-constrained project scheduling problem. *European journal of operational research*, 107(2):272–288.

Christofides, N., Alvarez-Valdés, R., and Tamarit, J. M. (1987). Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, 29(3):262–273.

Coelho, J. and Vanhoucke, M. (2018). An exact composite lower bound strategy for the resource-constrained project scheduling problem. *Computers & Operations Research*, 93:135–150.

Demeulemeester, E. and Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management science*, 38(12):1803–1818.

Demeulemeester, E. L. and Herroelen, W. S. (1997). New benchmark results for the resource-constrained project scheduling problem. *Management science*, 43(11):1485–1492.

Dorndorf, U., Pesch, E., and Phan-Huy, T. (2000). A branch-and-bound algorithm for the resource-constrained project scheduling problem. *Mathematical Methods of Operations Research*, 52(3):413–439.

Klein, R. and Scholl, A. (1999). Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling. *European Journal of Operational Research*, 112(2):322–346.

Li, P., Mazumdar, A., and Milenkovic, O. (2017). Efficient rank aggregation via lehmer codes. In *Artificial Intelligence and Statistics*, pages 450–459. PMLR.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., and Bianco, L. (1998). An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management science*, 44(5):714–729.

Nazareth, T., Verma, S., Bhattacharya, S., and Bagchi, A. (1999). The multiple resource constrained project scheduling problem: A breadth-first approach. *European Journal of Operational Research*, 112(2):347–366.

Patterson, J. H. and Huber, W. D. (1974). A horizon-varying, zero-one approach to project scheduling. *Management Science*, 20(6):990–998.

Sprecher, A. (2000). Scheduling resource-constrained projects competitively at modest memory requirements. *Management science*, 46(5):710–723.

Stinson, J. P., Davis, E. W., and Khumawala, B. M. (1978). Multiple resource–constrained scheduling using branch and bound. *AIIE Transactions*, 10(3):252–259.

Talbot, F. B. and Patterson, J. H. (1978). An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems. *Management Science*, 24(11):1163–1174.