APPENDIX A

Table A1 presents the detail configuration of each layer in AttenCon-Risk.

Table A1. The size of each layer in AttenCon-Risk

Layer	Shape size
H_{in} : input layer	34×1 for the left branch; 16×1 for the right branch;
H_{hid} : hidden layer	64×34 for the left branch; 64×16 for the right branch;
H_{loc} : local attention layer	64×64
H_{glo} : global attention layer	64×64
H_{con} : information fusion layer	128
H_{out} : output layer	128×1

Figure A1 shows the pseudo-code for constructing the HG using the financing transaction dataset D. The inputs of the pseudo-code include the financing transaction dataset D, the suppliers set x_s, the SMEs set x_f, the geographic area set x_a, the ownership structure set x_o and the industry set x_i. The output is the heterogeneous graph $G = (V, E)$. The underlying idea of the pseudo-code is that for each financing loan x_j, we enumerate all the instances of entities and their relations described in Figure 1, and add them one-by-one into the heterogeneous graph $G = (V, E)$. Lines 2-4 are used to initialize a heterogeneous graph and 8 empty edge sets. Lines 7-9 are used to generate nodes for the heterogeneous graph, from the corresponding entity instances. Line 11 is used to add the relation instance "SME's supplier (*S*) releases one of the loans (*L*)" to the heterogeneous graph. Line 12 is used to add the relation instance "the loan (L) is released by a SME's supplier (S)" to the heterogeneous graph. In the same way, Lines 13-28 can be explicitly explained.

```
==========================================================
Input: The financing transaction dataset D, the suppliers set x_s, the SMEs set x_f,
the geographic area set x_a, the ownership structure set x_o, the industry set x_i.
Output: The heterogeneous graph G = (V, E)
==========================================================
# Initialization
Initialize a heterogeneous graph G = (V, E)
Initialize 8 empty edge sets: Set_SF, Set_FS, Set_FA, Set_AF, Set_FO,
Set_OF, Set_FI, Set_IF
For each financing loan x_j in D:
    # Generate nodes for the heterogeneous graph
    nodeL = new node ( x_{l,j} , L); nodeS = new node ( x_{s,j} , S)
    nodeF = new node ( x_{f,j} , F); nodeA = new node ( x_{a,j} , A)
    nodeO = new node ( x_{o,j} , O); nodeI = new node ( x_{i,j} , I)
    # Update edges into the HG
    edgeSL = new edge (nodeS, nodeL, 1); G.add(edgeSL) # Relation "S→L"
    edgeLS = new edge (nodeL, nodeS, 2); G.add(edgeLS) # Relation "L→S"
    edgeSF = new edge (nodeS, nodeF, 3)                # Relation "S→F"
    if edgeSF not in Set_SF: G.add(edgeSF); Set_SF.add(edgeSF)
    edgeFS = new edge (nodeF, nodeS, 4)                # Relation "F→S"
    if edgeFS not in Set_FS: G.add(edgeFS); Set_FS.add(edgeFS)
    edgeFA = new edge (nodeF, nodeA, 5)                # Relation "F→A"
    if edgeFA not in Set_FA: G.add(edgeFA); Set_FA.add(edgeFA)
    edgeAF = new edge (nodeA, nodeF, 6)                # Relation "A→F"
    if edgeAF not in Set_AF: G.add(edgeAF); Set_AF.add(edgeAF)
    edgeFO = new edge (nodeF, nodeO, 7)                # Relation "F→O"
    if edgeFO not in Set_FO: G.add(edgeFO); Set_FO.add(edgeFO)
    edgeOF = new edge (nodeO, nodeF, 8)                # Relation "O→F"
    if edgeOF not in Set_OF: G.add(edgeOF); Set_OF.add(edgeOF)
    edgeFI = new edge (nodeF, nodeI, 9)                # Relation "F→I"
    if edgeFI not in Set_FI: G.add(edgeFI); Set_FI.add(edgeFI)
    edgeIF = new edge (nodeI, nodeF, 10)               # Relation "I→F"
    if edgeIF not in Set_IF: G.add(edgeIF); Set_IF.add(edgeIF)
end for
return the heterogeneous graph G = (V, E)
==========================================================
```

Figure A1. The pseudo-code for constructing the HG in credit risk prediction

Figure A2 shows the pseudo-code of the graph traverse algorithm for producing meta paths from the constructed network schema. The inputs of the graph traverse algorithm include the start node L, the end node L and the constructed network schema. The output is a list of meta paths. In Figure B3, lines 1-9 are the "generate_meta_paths" function, which utilizes other helper functions to produce all meta paths that start from L and end with L from the provided network schema. Lines 10-15 are the "generate_paths_recursive" function, which is a recursive helper function for the "generate_meta_paths" function. This function recursively produces

```
=============================================================
Input: start node L, end node L, the network schema
Output: a list of meta paths
=============================================================
1 FUNCTION generate_meta_paths(start_node, end_node, schema){
2     Initialize an empty list: meta_path_list
3     temp_path_list = get_one_step_paths(NULL, start_node, schema)
4     for each path in temp_path_list:
5         if path.endNode == end_node: meta_path_list.add(path)
7         else: meta_path_list.add(generate_paths_recursive(path, end_node, schema))
9     return meta_path_list;}
-------------------------------------------------------------------------------------------
10 FUNCTION generate_paths_recursive(path, end_node, schema){
11     if is_path_valid(path):
12         temp_path_list = get_one_step_paths(path, path.endNode, schema)
13         for each tempPath in temp_path_list:
14             if tempPath.endNode == end_node: return tempPath
15             else: generate_paths_recursive(tempPath, end_node, schema);}
-------------------------------------------------------------------------------------------
16 FUNCTION get_one_step_paths(path, start_node, schema){
17     Initialize an empty list: temp_path_list
18     if path is NULL:
19         for each link in schema:
20             if link.startNode == start_node: temp_path_list.add(link)
21     else:
22         for each link in schema:
23             if link.startNode == path.endNode:
24                 temp_path = combine(path, link); temp_path_list.add(temp_path)
25     return temp_path_list;}
-------------------------------------------------------------------------------------------
26 FUNCTION combine_path(path, link){
27     temp_path = copy(path)
28     temp_path.Sequences.add(path.endNode); temp_path.endNode = link.endNode
29     return temp_path;}
-------------------------------------------------------------------------------------------
30 FUNCTION is_path_valid(path):
31     Initialize two empty lists: temp_nodes_list, temp_link_list
32     for each value in path.values: temp_nodes_list.add(value)
33     for i from 0 to (length(temp_nodes_list) – 1):
34         temp_link_list.add(temp_nodes_list[i] + "-->" + temp_nodes_list[i+1])
35     return length(temp_link_list) == length(set(temp_link_list));}
=============================================================
```

Figure A2. The pseudo-code of the graph traverse algorithm for producing meta paths from the constructed network schema

possible paths from the given valid path to the end node L. Lines 16-25 are the "get_one_step_paths" function, which retrieves all possible one-step paths from the provided network schema and the given start node. Lines 26-29 are the "combine_path" function, which combines the given path with link to produce a new path. Lines 30-35 are the "is_path_valid"

function, which checks whether a given path is valid. Actually, the "is_path_valid" function is used to remove meta paths that contain consecutive repetitive sub-paths. This rule can help to avoid redundancy and circular patterns in the produced meta paths, thus ensuring that the produced meta paths capture meaningful and diverse spillover effect of credit risk from the constructed HG. For instance, the meta path "*L*-*S*-*F*-*S*-*F*-*S*-*L*" contains consecutive repetitions of the sub-path "*S*-*F*", which may limit the information diversity captured by the meta path. Therefore, such meta paths should be filtered out to enhance the meaningfulness and diversity of the produced meta paths.

APPENDIX B

B1. Ablation study

As described in Section 5.2, we conduct an analysis of the effects of the local attention, the global attention and the information fusion on AttenCon-Risk's performance. We develop four variants of AttenCon-Risk, namely AttenCon-Risk-AvgLocal, AttenCon-Risk-AvgGlobal, AttenCon-Risk-LoanOnly and AttenCon-Risk. Figure B1 panel (a) presents the average recall and G-mean of the four variants of AttenCon-Risk over the five retaining levels, while Figure B1 panel (b) presents their absolute financial loss distribution over the five retaining levels. We can see from Figure B1 panel (a) that AttenCon-Risk outperforms the other three variants in both recall and G-mean, while the AttenCon-Risk-LoanOnly approach exhibits the poorest performance among the four variants. Regarding the recall, AttenCon-Risk provides an improvement of 2.53%, 5.50% and 11.42% compared to that of the AttenCon-Risk-AvgLocal approach, the AttenCon-Risk-AvgGlobal approach and the AttenCon-Risk-LoanOnly approach, respectively. Regarding the G-mean, AttenCon-Risk provides an improvement of 1.57%, 1.88% and 4.87% compared to that of the AttenCon-Risk-AvgLocal approach, the AttenCon-Risk-AvgGlobal approach and the AttenCon-Risk-LoanOnly approach, respectively.

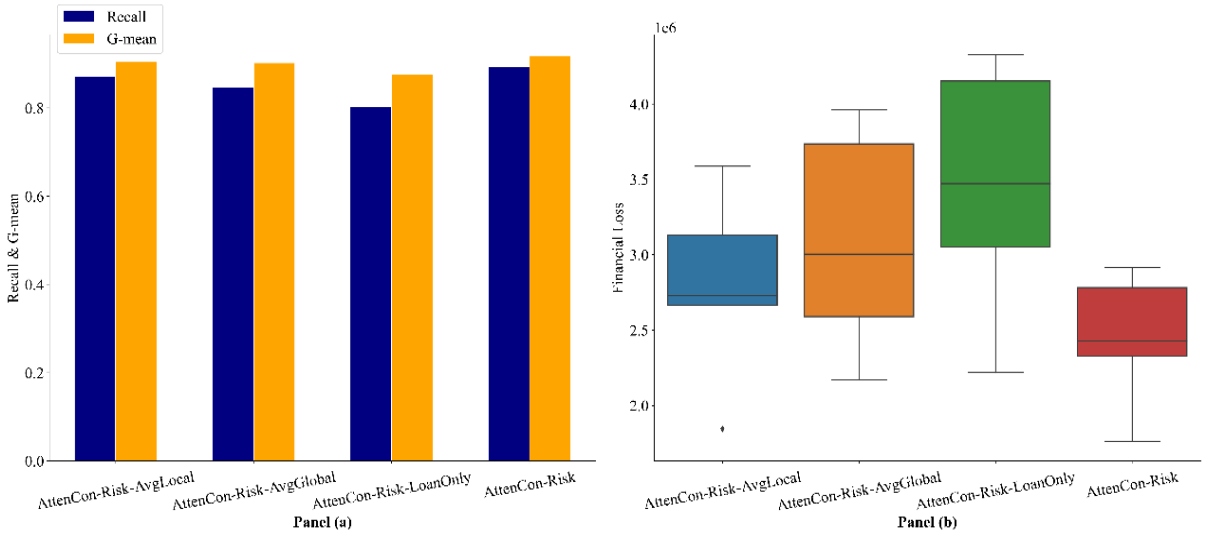

Figure B1. The performances of the four variants of AttenCon-Risk in terms of recall, g-mean and financial loss

The results in Figure B1 panel (b) show that the absolute financial loss of the four variants of AttenCon-Risk aligns with the average recall and G-mean of the four variants represented in Figure B1 panel (a). Specifically, AttenCon-Risk significantly outperforms the other three variants in terms of financial loss, achieving the lowest median financial loss and exhibiting remarkable stability in its financial loss distribution. On average, AttenCon-Risk provides a decrease in financial loss of 12.47%, 21.00% and 29.07% compared to the AttenCon-Risk-AvgLocal approach, the AttenCon-Risk-AvgGlobal approach and the AttenCon-Risk-LoanOnly approach, respectively.

In comparison to the AttenCon-Risk-AvgLocal approach and the AttenCon-Risk-AvgGlobal approach, the superior performance of AttenCon-Risk highlights the significance of both the loan level heterogeneity in spillover effects of credit risk over each meta path and the meta-path level heterogeneity in spillover effects of credit risk through multiple meta paths for improving the credit risk prediction in SCF. Nonetheless, the superior performance of the AttenCon-Risk-AvgLocal approach over the AttenCon-Risk-AvgGlobal approach reveals that the meta-path level heterogeneity in spillover effects of credit risk through multiple meta paths is more critical than the loan level heterogeneity in spillover effects of credit risk over each meta path in improving the credit prediction in SCF. Furthermore, the superior performance of AttenCon-Risk over the AttenCon-Risk-LoanOnly approach emphasizes that the demographic

data of SMEs is an essential complement for improving the credit prediction in SCF. This data provides the enterprise-specific information on SMEs that should repay financing loans.

B.2 Parameter tuning

To examine the effects of hyper parameters on AttenCon-Risk's performance, we conduct parameter tuning at a retaining level of 0.65. As discussed in Sections 4.2 and 4.3, the tradeoff rate α_b and the learning rate α are the hyper parameters that should be tuned, as the former mitigates the bias towards the non-default financing loans and the latter controls the convergence of the learning process. Following the work of Lin et al. (2017) and Lima-Junior et al. (2019), we vary the tradeoff rate from 0.65 to 0.96 and the learning rate from 0.005 to 0.05. Tables B1 and B2 present the results of tuning the tradeoff rate and learning rate, respectively. The bold number with the "*" symbol indicates the best performance. We can see from Table B1 that AttenCon-Risk's performance increases as we increase the tradeoff rate from 0.65 to 0.90. However, as we further increase the tradeoff rate to 0.96, AttenCon-Risk's performance decreases, with the maximum achieved at the tradeoff rate of 0.90. Regarding the learning rate, we can see from Table B2 that AttenCon-Risk's performance achieves its maximum at the learning rate of 0.007. Consequently, we set the tradeoff rate to 0.90 and the learning rate to 0.007 in the comparison experiments and ablation studies.

Table B1. The performance of AttenCon-Risk for tuning the tradeoff rate α_b at the retaining level of 0.65

Tradeoff rate (α_b)	**Recall**	**G-mean**	**Financial loss**
0.65	0.6842	0.8190	-8,023,603
0.70	0.7719	0.8694	-6,485,568
0.75	0.8070	0.8874	-5,052,656
0.80	0.8246	0.8927	-4,725,342
0.85	0.8421	0.8973	-4,224,713
0.90	**0.9123***	**0.9317***	**-2,916,021***
0.95	**0.9123***	0.9283	-3,051,913
0.96	0.7895	0.8714	-4,906,859

Notes: The bold number with the "*" symbol indicates the best performance.

Table B2. The performance of AttenCon-Risk for tuning the learning rate α at the retaining level of 0.65

Learning rate (α)	**Recall**	**G-mean**	**Financial loss**
0.005	0.8947	0.9255	-2,989,338
0.007	**0.9123***	**0.9317***	**-2,916,021***
0.009	0.8772	0.9141	-3,672,084
0.01	0.8947	0.9244	-3,088,088
0.03	0.8947	0.9221	-3,127,845
0.05	0.7895	0.8720	-4,661,579

Notes: The bold number with the "*" symbol indicates the best performance.