A search algorithm to identify the independent feedback loop set

Abstract

System dynamics focuses on how feedback structures drive system behaviour. An established feedback loop analysis method is eigenvalue elasticity analysis (EEA), which analyzes a complete set of independent feedback loops in a given system. A widely accepted loop selection method is the shortest independent loop set (SILS) algorithm. It is utilized in EEA to compute the loop elasticity which identifies the dominant loops. However, this paper finds that in some scenarios, the SILS can only identify part of the complete independent loop set (ILS). In this case, SILS is no longer suitable for EEA, because it produces incorrect loop elasticities. An agent-based goal diffusion model is then produced to demonstrate this specific scenario. Subsequently, we specify a more robust algorithm using the depth-first search to identify the complete set of independent loops. Finally, a summary is presented and it suggests a potential area for extending EEA applications.

1 Introduction

System dynamicists explore and analyze feedback structures in order to provide explanations for system behaviour. The process of identifying the feedback loops that exert the most significant influence on the variable of interest is referred to as dominant loop analysis. Once the dominant feedback loops are identified, this information can be utilized to design more effective policies.

Dominant loop analysis can be performed using a number of related approaches. Ford (1999) proposed the behavioural method through deactivating each feedback loop, and comparing the behaviour with that in the original model. Another approach is the pathway participation metric (PPM) method, proposed by Mojtahedzadeh (1997), which identifies the most influential pathways sequentially to form a dominant feedback structure. The third method is the eigenvalue elasticity analysis (EEA), introduced by (Forrester, 1982), which is derived from linear control theory. The eigenvalues (denoted as λ) determine the behaviour of a system (Forrester, 1982; Saleh, 2002). As a result, a dimensionless ratio, elasticity, ϵ is defined to measure the percentage change in the eigenvalue in response to the percentage change in the loop gain (g_s), defined as $\epsilon = (d\lambda/\lambda)/(dg_s/g_s)$ (Forrester, 1983). The larger the magnitude of an eigenvalue elasticity associated with the gain of a certain system structure element, the more significant that loop structure element is to the corresponding eigenvalue. Among these three methods, two of which, the behavioural method and EEA, utilize loops as basic analysis block while the PPM, identifies a dominant pathway[1] until these pathways form a feedback loop. This paper focuses on the independent feedback loop set (ILS) selection method which is proposed and introduced into EEA by Kampmann (1996), and the paper is structured as follows. First, an introduction to ILS is presented. The ILS is considered as a breakthrough in EEA as it enables the direct computation of the loop elasticity and uses a reduced number of loops to explain the model behaviour. Next, the shortest independent loop set (SILS) proposed by Oliva (2004) is described. It is a simpler and more granular description of ILS. Following this, an experiment is presented, based on an individual-based goal diffusion model, that demonstrates a clear difference between the results of the SILS and those from the ILS. An algorithm is then proposed which can overcome the discrepancy between the SILS and the ILS, and this is validated through a number of experiments. Finally, our conclusion highlights potential issues associated with the determination of the maximal independent loop set.

2 The independent loop set

Despite the fact that loop elasticity is an effective way to identify dominant feedback loops, it was not widely used until Kampmann (1996) proposed a method known as the independent loop set. This is due to a previously unsolved problem: how to select feedback loops so that loop elasticities can be calculated from link elasticities (which are known). Eq. 1 formulates the relationship between these two elasticities (note: the edges and loops are in a strongly connected digraph[2]). The

[1]A number of edges which link two state variables.

[2]In a strongly connected digraph, any pair of vertices could reach each other.

manner in which the feedback loop set is selected determines the matrix C, and also determines whether loop elasticity on the right-hand side of the equation can be solved.

$$\begin{bmatrix} E_{e_1} \\ E_{e_2} \\ \cdot \\ E_{e_N} \end{bmatrix} = \mathbf{C} \begin{bmatrix} E_{c_1} \\ E_{c_2} \\ \cdot \\ E_{c_L} \end{bmatrix} \tag{1}$$

E_e and E_c denote link elasticities and loop elasticities respectively, L refers to the number of possible loops, N represents the number of edges, and $\mathbf{C}$ is an $N \times L$ binary matrix. $\mathbf{C}$ represents the directed cycle matrix (DCM) (Kampmann, 1996), each entry of which indicates if the edge (in a row) belongs to a certain loop (in a column). If the number of possible feedback loops of a given system is denoted as L, L is typically much larger than N. However, the solution to the loop elasticity is determined by the rank[3] of $\mathbf{C}$: rank($\mathbf{C}$)$\leq N < L$. In this type of scenario, there will be no unique solution to the loop elasticity, as the number of unknowns are greater than the number of equations.

The independent loop set (ILS) was proposed by Kampmann (1996) to address this problem where the ILS is "a maximum set of loops whose incidence vectors are linearly independent ... In a strongly connected digraph, the number of maximal independent loops are ***N-n+1***", where N is the number of edges, and n is the number of vertices. It is worth noting that the ILS is not unique although its size is fixed. There are multiple possible ILSs for one given system.

Because of the ILS, the number of feedback loops under consideration is lower than the maximum possible number, and is therefore reduced to a manageable number. We can see that rank($\mathbf{C}$)= $\hat{L}$ which denotes the number of independent loops. Once an ILS has been identified, a unique solution to the loop elasticity is guaranteed. This then enables the EEA method to be used.

[3]The column rank of a matrix is the maximum number of its linearly independent column vectors. The row rank of a matrix is the maximum number of its linearly independent row vectors. The fundamental theorem of linear algebra states that the column rank is always equal to the row rank.

3 Shortest independent loop set

Based on Kampmann's construction rule, Oliva (2004) proposed a new algorithm to identify the independent loop set, which is referred to as the shortest independent loop set (SILS) algorithm. The essence of SILS is based on the idea of *geodetic cycles*. A geodetic cycle is constructed by using the fewest edges connecting any two system vertices under consideration, and all geodetic cycles are a subset of all the possible feedback loops in the graph. The geodetic cycles are computed from the adjacency matrix[4]. The SILS is constrained to only consider new loops from the geodetic cycles. Furthermore, the selected candidate loop always has the fewest new edges (compared with other candidate geodetic cycles). This is consistent with Kampmann's construction rule which states to "accept a loop in the ILS if it contains at least one edge not included in the previously accepted loops". The SILS is a special implementation of ILS.

The steps to implement the SILS algorithm can be briefly described as follows:

1. Set up the geodetic cycle matrix for a given system.
2. Id the shortest feedback loop among the geodetic cycles, and transfer it to the SILS. This is the first loop introduced into the SILS.
3. Among the remaining geodetic cycles, select the shortest feedback loop that makes the smallest addition of new edges to the identified SILS. Add it to the loop set.
4. Repeat step 3 until all the geodetic cycles are visited.

The SILS algorithm has become the most widely adopted approach in feedback loop analysis, for example, the behavioural method by Phaff (2008), and the eigenvalue elasticity analysis (Güneralp, 2006; Kampmann and Oliva, 2006; Saleh et al., 2010). However, a key question to be addressed is whether the SILS algorithm can capture ILS for any system dynamics models, because it constrains the candidate feedback loops to be based on the geodetic cycles.

[4]Each row (and column) represents a vertex, and the entries are restricted to zero and one, where A_{ij}=1 IFF there is a link from vertex i to vertex j. The ones in row represent the successors while the ones in column represent the predecessors.

Our earlier discussion shows that the loop elasticities, $[E_{c1}, \cdots, E_{c\hat{L}}]^T$ in Eq. 1, can be solved provided rank($\mathbf{C}$) $= \hat{L}$. More importantly, it suggests that any smaller sized independent loop set rather than the ILS is sufficient to solve loop elasticity in Eq. 1. The explanation for the necessity of utilizing the ILS in the loop elasticity computation lies in the utilization of Mason's rule (Reinschke, 1988), and its detail can be found in Kampmann (1996).

4 Does SILS capture the complete set of independent loops?

The SILS algorithm is generally accepted as the loop selection method in eigen-based analysis, however, we have formulated a model which shows that the SILS algorithm does not capture the maximum number of independent loops. This is an individual-based system dynamics model (Feola et al., 2011; Osgood, 2009; Duggan, 2008) based on the two-stock floating goals structure (Sterman, 2000). We have adapted the original model to allow individual agents to pursuit different goals by modifying their own target goal in equation F_i. Each agent has one state variable $GoalAgent_i$, representing its current goal level. $GoalAgent_i$ is adjusted through interactions with its neighbours via $TargetGoal_i$, and the adjustment time is AT_i. A three-agent model with a fully connected network termed the goal diffusion model is shown in Figure 1, and the equations in (2) are for each individual agent.

$$\begin{aligned}
GoalAgent_i &= INTEGRAL(ChangeInGoal_i, GoalAgent_i^0) \\
ChangeInGoal_i &= (TargetGoal_i - GoalAgent_i)/AT_i \\
TargetGoal_i &= F_i(GoalAgent_1, GoalAgent_2, GoalAgent_3)
\end{aligned} \tag{2}$$

In order to clarify the feedback loop set, we present a graph representation of the model in Figure 2 (adjustment time*s* are not included as they are not part of the strongly connected digraph).

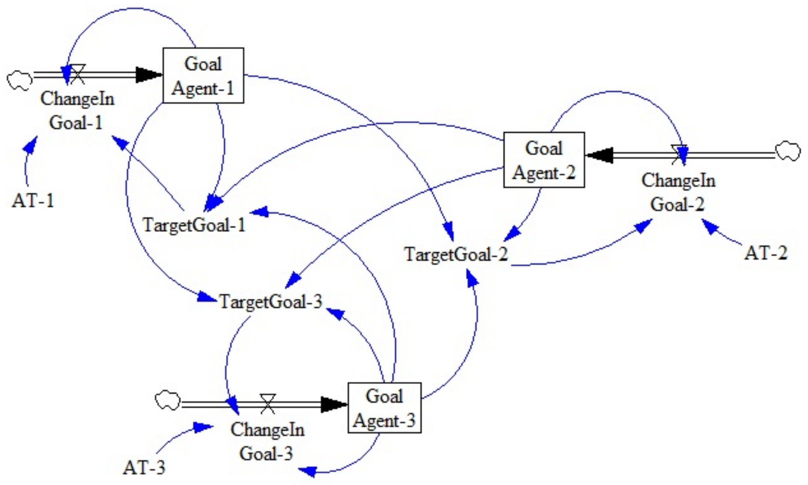

Figure 1: The stock and flow diagram of the goal diffusion model

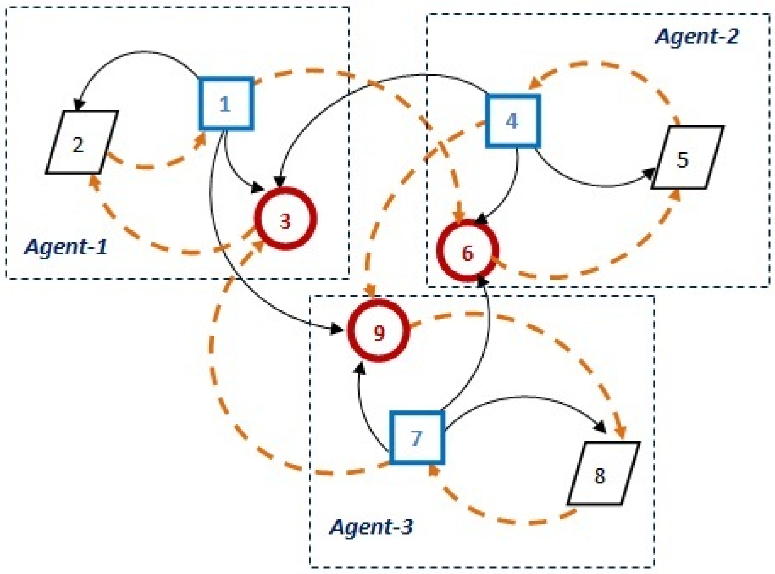

Figure 2: The simplified graph of the goal diffusion model. Squares: *GoalAgent*s, circles: *TargetGoal*s, and parallelograms: *ChangeInGoal*s.

Method		Loop name		Variables involved	Sequence
ILS	SILS	Agent1	floating goal spiral	1, 2	①
			state adjustment	1, 3, 2	②
		Agent2	floating goal spiral	4, 5	④
			state adjustment	4, 6, 5	⑤
		Agent3	floating goal spiral	7, 8	⑦
			state adjustment	7, 9, 8	⑧
		interaction of A1 & A2		1, 6, 5, 4, 3, 2	③
		interaction of A1 & A3		1, 9, 8, 7, 3, 2	⑥
		interaction of A2 & A3		4, 9, 8, 7, 6, 5	⑩
	interaction of A1, A2 & A3			1, 9, 8, 7, 6, 5, 4, 3, 2	⑨

Table 1: ILS and SILS in the goal diffusion model.

Based on Kampmann's theorem 2.1, the maximal independent loop number of the goal diffusion model is: ***18-9+1=10***. However, the SILS algorithm can only identify ***9*** loops. The independent feedback loop sets identified by these two different approaches are shown in Table 1. Particularly, the fourth column with numbers encircled shows the sequence of the independent loops that are identified by ILS. It is evident that there is one loop missing in SILS which is highlighted by the dashed line in Figure 2. The missing loop is a longest loop that spans three agents. Close scrutiny shows that the missing loop is not a geodetic cycle. In this model, any two vertices in different agents can be linked by a shortest path encompassing the corresponding two agents without involving a third agent. This shows that constraining the candidate loop pool to be the geodetic cycles is a potential drawback that will affect the number of independent loops which can be identified. This is the reason why the SILS will not achieve the full coverage of ILS in some cases.

Further investigation provides a deeper explanation to this drawback of the SILS algorithm: the loops introduced to ILS by Kampmann's construction rule are not always the geodetic cycles. Recall the first statement in point 2 of this rule (p.8, Kampmann (1996)) dealing with adding a new loop, it is expressed as "choose a shortest path outside that comes back to the current ILS". A geodetic cycle is a combination of two shortest paths. Nevertheless, this construction rule needs only the path (part of the new loop) outside the ILS to be the shortest. It does not require the path inside the ILS to be the shortest. Therefore, a non-geodetic cycle may be introduced into the ILS. It is worth noting that any two vertices in ILS are connected, but not necessarily by the shortest path. This scenario can be observed from the case example, with first 8 loops introduced in Table 1, for loop 9, the edge 7→6 is added. This new edge is part of the geodetic cycle involving vertices 7 and 6 (the geodetic cycle is 7→6→5→4→9→8→7). However, the path inside ILS from 6 to 7 is 6→5→4→3→2→1→9→8→7, and not the shortest. Consequently, a non-geodetic cycle is introduced into ILS. Finally, the last remaining edge 4→9 is introduced to form loop 10.

5 An independent loop set selection method

In this section, we will specify an ILS algorithm by making use of Kampmann's construction rule. However,there are several implementation issues that have not been specified in this rule. For example, there is no clear starting point of the ILS algorithm, and it contains an ambiguous procedure to introduce a new path returning to current ILS. In our algorithm, we select a shortest feedback loop as the first loop, which can be obtained by the geodetic loop matrix. The geodetic matrix is computed as follows (intermediate results are shown based on Figure 2. Within each matrix, i^{th} row corresponds to vertex i):

1. The adjacency matrix $\mathcal{A}$, a binary representation of the digraph, is constructed based on the model's equations.

$$\mathcal{A}=\begin{pmatrix} 0&1&1&0&0&1&0&0&1\\ 1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0\\ 0&0&1&0&1&1&0&0&1\\ 0&0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&1&0&0&1&0&1&1\\ 0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1&0 \end{pmatrix} \mathcal{D}=\begin{pmatrix} 0&1&1&3&2&1&3&2&1\\ 1&0&3&4&3&2&4&3&2\\ 2&1&0&5&4&3&5&4&3\\ 3&2&1&0&1&1&3&2&1\\ 4&3&2&1&0&2&4&3&2\\ 5&4&3&2&1&0&5&4&3\\ 3&2&1&3&2&1&0&1&1\\ 4&3&2&4&3&2&1&0&2\\ 5&4&3&5&4&3&2&1&0 \end{pmatrix} \mathcal{D}+\mathcal{D}'=\begin{pmatrix} 0&2&3&6&6&6&6&6&6\\ &0&4&6&6&6&6&6&6\\ &&0&6&6&6&6&6&6\\ &&&0&2&3&6&6&6\\ &&&&0&3&6&6&6\\ &&&&&0&6&6&6\\ &&&&&&0&2&3\\ &&&&&&&0&3\\ &&&&&&&&0 \end{pmatrix}$$

2. The distance matrix ($\mathcal{D}$)is computed where each entry shows the length of the shortest path (a sequence of non-repeating edges and vertices) between two vertices (Warfield, 1989). The distance matrix is obtained based on a well known result from graph theory which states: "an adjacency matrix with its main diagonal filled with ones – to the *ith* power, yields the matrix of a digraph with the relationship reachable with *i* steps". The above statement can be formulated as (Oliva, 2004):

$$\begin{aligned} \mathcal{B} &= \mathcal{A} + I \\ \mathcal{D} &= \mathcal{A} + \sum_{i=2}^{|\mathcal{A}|-1} i(\mathcal{B}^i - \mathcal{B}^{i-1}) \end{aligned}$$

   where $|A|$ is the row/column dimension of $\mathcal{A}$, and the power of $\mathcal{B}$ is a Boolean product.

3. The geodetic matrix is computed by adding the transpose of the lower block triangular component of the distance matrix to its upper triangular component $\mathcal{D}'$+$\mathcal{D}$. The plus operator is in ordinary matrix algebra.

If there is more than one loop with the equal shortest length, we will choose the one which is placed first in the geodetic loop matrix. The loop is divided into two paths, e.g. $u \to v$ and $v \to u$

($\mathcal{D}_{uv}$ = shortest length), thus, a loop track can be carried out by tracking the elements in individual path. Figure 1 shows the pseudo-code for the path track by taking advantage of the distance matrix.

Algorithm 1 Vector pathTrack (Matrix $\mathcal{D}$, int u, int v)

```
Vector vec, pre, suc                  {initialize the vectors}
vec ← u                               {add the first vertex of the path}
len ← $\mathcal{D}$(u,v)              {get the length of the shortest path}
for $i$ = 0 to len-1 do
  pre ← findNPre ($\mathcal{D}$, v, length-i)  {find all predecessors of v with length-i steps away}
  suc ← findNSuc ($\mathcal{D}$, u, 1)         {find all immediate successors of u}
  for each vertex j ∈ suc do
    if j ∈ pre then
      vec ← j                         {add the vertex to the vector vec}
      break
    end if
  end for
end for
return vec
```

Once the first feedback loop is selected, a number of key issues are clarified before proceeding, namely to:

- Distinguish the vertices which have been added into the ILS (or call them visited vertices) from those which have not been visited.
- Maintain two adjacency matrixes: one contains the edges in current ILS (or named as visited edges), and the other is composed of the edges outside of ILS.
- Compute the distance matrix $\mathcal{D}$ with visited edges only, and update it every time after a new loop is introduced into the ILS. This is one distinction from the SILS where the distance matrix is calculated with all the edges.

We now proceed to the core of Kampmann's construction rule: start from a visited vertex and track a *shortest* path back to the existing loop set. Our algorithm takes a slightly different strategy on how to introduce new loops to the ILS. We start from a vertex, then proceed until we identify that one vertex is visited (Figure 3). As a result, this path may not be the shortest. However, it still meets the constraint that "every new loop has at least one new edge" (Kampmann, 1996). This

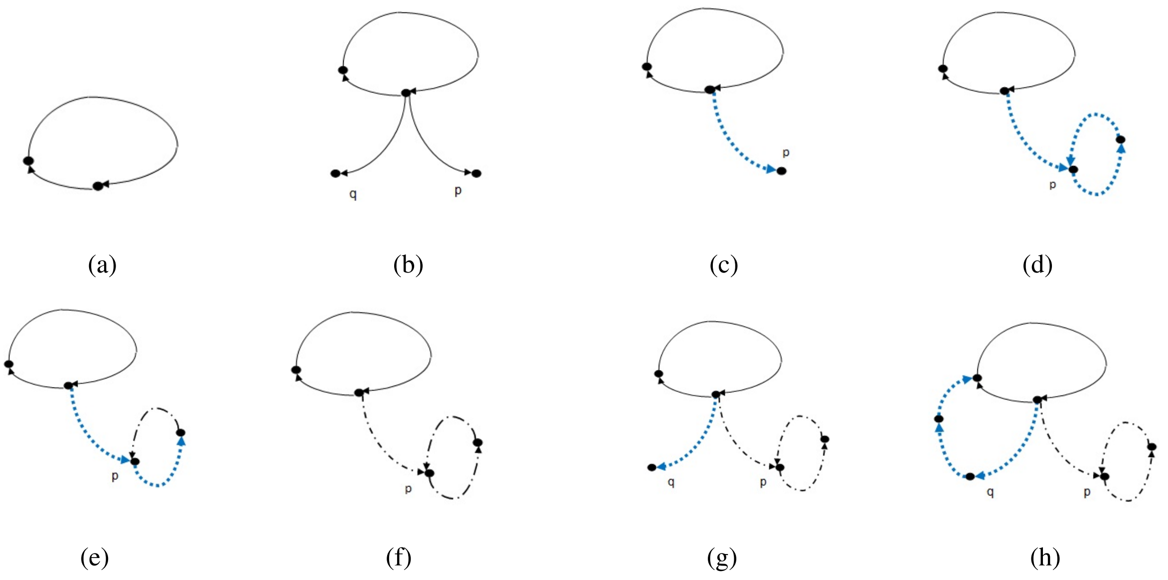

Figure 3: Illustration of the searching process for a new independent feedback loop. Dotted blue line: current valid search path; black dash line: visited but discarded edges.

Algorithm 2 Stack addLoop (int cur, Vector vertices, Matrix $\mathcal{A}$)

```
Stack S = new Stack()            {initialize a stack}
Vector path = new Vector()       {initialize a vector}
repeat
  if cur ∈ vertices then
    S.push(cur)                  {if the cur vertex is a visited vertex, push it in stack}
    return S
  else if cur != -1 then
    S.push(cur)                  {if the succeeding edge exists, push the cur vertex in stack}
    path ← cur                   {add it to the search path}
    cur ← findSuc(𝒜, cur, path)
  else
    S.pop()                      {if no succeeding edge exists, pop out the vertex}
    q ← S.peek()                 {obtain the new top vertex}
    cur ← findSuc(𝒜, cur, path)
  end if
until !S.isEmpty()
return S
```

procedure is implemented by "depth-first search" (Cormen et al., 2001) and a stack data structure is used to record all the vertices in the search path. If the vertex is not labeled as visited (i.e. not in current ILS), but it is in the current search path (marked in dashed/dotted lines in Figure 3(c)-(h)), we have to discard this edge and try another unvisited succeeding edge. In some cases, all its succeeding edges may go back to the vertices in the search path, we have to remove the top node in the stack and search for a new path from the new top node. Figure 3 demonstrates a scenario: initially we follow the edge to p to explore a way back to the ILS, however, we have to retreat three steps to the starting point and opt for another succeeding edge ending with q, which finally leads back to ILS. Algorithm 2 shows an implementation of this procedure that takes the starting vertex, visited vertices, and the adjacency matrix (with unvisited edges only) as inputs.

Once we have identified the path outside of ILS, the remaining task is to complete this feedback loop. As we know the head and tail nodes in stack hold two visited vertices, so we can make use of the code in Figure 1 to trace the path in the current ILS. Every time a new feedback loop is introduced into ILS, the adjacency matrix, the distance matrix, and the visited vertex and edge set are updated. The shortest path between two vertices thereby may change every time after one new loop is introduced. This is a significant difference with the SILS algorithm where the geodetic loop matrix is static. Moreover, this gives rise to generating more feedback loops than the geodetic loops, and therefore ensures the algorithm identifies the ILS.

Finally, a formal pseudo-code for the main function of identifying the ILS is provided in Algorithm 3. It takes four parameters as the input. $\mathcal{A}$ and $\mathcal{D}$ are the system adjacency matrix and distance matrix. Vertices is a vector containing all the vertices while Edges is a matrix with all edges in the system.

A subsequent experiment is conducted to examine the undetected loop in the goal diffusion model when increasing the agent population. Table 2 shows that the gap of the loop size between the ILS and the SILS grows when the agent population rises in the context of a fully connected network.

Algorithm 3 Main($\mathcal{A}$, $\mathcal{D}$, Vertices, Edges)

```
Vector vis_v, non_vis_v← Vertices      {Initialize vectors to store visited and non-visited vertices}
Vector vis_e, non_vis_e← Edges         {Initialize vectors to store visited and non-visited edges}
Vector g_cycles                        {Initialize vectors to geodetic cycles}
Vector ils                             {Initialize a vector to store the independent loop set}
g_cycles ← calcGeodeticM($\mathcal{D}$)            {Obtain geodetic cycles by utilizing pathTrack func}
ils←g_cycles                           {Introduce a shortest loop to ils}
vis_v←updateV(ils,Vertices)            {Update the visited vertex vector}
non_vis_v←Vertices− vis_v              {Obtain the non-visited vertex vector}
vis_e←updateE(ils,Edges)               {Update the visited edge vector}
non_vis_e←Edges− vis_e                 {Obtain the non-visited edge vector}
$\mathcal{A}$←updateAdj(non_vis_e)                {Update adjacency matrix with edges outside the ILS}
$\mathcal{D}$←updateDis(vis_e)                    {Update distance matrix with edges in the ILS}
while non_vis_e != null do
  int cur←findCur(non_vis_v, $\mathcal{A}$)       {Pick one vertex in non_vis_v and it has an edge with a vertex in vis_v}
  int v←findNPre($\mathcal{A}$,cur,1)              {Record the vertex in vis_v}
  S←addLoop(cur,vis_v,$\mathcal{A}$)               {Identify a path going back to current ILS}
  u ← S                                {Get the vertex that is in both ILS and S}
  S.add(pathTrack($\mathcal{D}$, u, v))            {Identify a path inside current ILS to form a new loop}
  ils.add(S)                           {Add the new loop}
  vis_v←updateV(S,Vertices)
  non_vis_v←Vertices− vis_v
  vis_e←updateE(S,Edges)
  non_vis_e←Edges− vis_e
  $\mathcal{A}$←updateAdj(non_vis_e)
  $\mathcal{D}$←updateDis(vis_e)
  S.clear()                            {Clear the vector S}
end while
```

Agent population	2	3	4	5	6	7	8	9	10	$i(\geq 2)$
No. of edges (N)	10	18	28	40	54	70	88	108	130	$(i+3)i$
No. of variables (n)	6	9	12	15	18	21	24	27	30	$3i$
N-n+1	5	10	17	26	37	50	65	82	101	i^2+1
No. of loops (ILS)	5	10	17	26	37	50	65	82	101	i^2+1
No. of loops (SILS)	5	9	14	20	27	35	44	54	65	$(i+3)i/2$
No. of missing loops	0	1	3	6	10	15	21	28	36	$\sum_{j=2}^{i}(j-2)$

Table 2: Agent population vs. size of independent loop set

This table shows the SILS algorithm fails to yield an ILS for such type of models. Although the specific conditions that makes the SILS fails is not clearly known yet, we think it is due to the density of the interconnections.

6 Conclusions

This paper specifies an algorithm that identifies the complete set of independent loops that outperforms the SILS for a specific individual based goal diffusion problem. The ILS algorithm conforms to Kampmann's construct rule and uses depth-first search to add a new feedback loop into the existing ILS. Another contribution of this paper is to highlight the need for caution when utilizing the SILS algorithm in EEA. The SILS algorithm is not guaranteed to identify the ILS. This is due to the fact that the SILS constrains the candidate feedback loops within only geodetic loops, which rules out other possible feedback loops. In addition, it is important to emphasize the point that only by utilizing the complete set of independent feedback loops, is the loop elasticity calculation valid, and hence leads to Eq. 1. This is the equation where the loop elasticity derived. However, although the SILS algorithm fails in few scenarios, it still can capture the ILS in most system dynamic models. Moreover, while the capability of capturing the core dynamics for the loops generated by our algorithm needs to be examined, that for the loops in SILS is proved in Oliva and Mojtahedzadeh (2004).

Future research will focus on the possible applications of EEA to network-based individual models (Mungovan et al., 2011; Rahmandad and Sterman, 2008). This includes extending the existing techniques and exploring new approaches to formally analyze feedback loop structures across disaggregated equation-based models.

References

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein (2001). *Introduction to Algorithms* (2nd ed.). MIT Press and McGraw-Hill.

Duggan, J. (2008). Equation-based policy optimization for agent-oriented system dynamics models. *System Dynamics Review 24*(1), 97–118.

Feola, G., J. A. Gallati, and C. R. Binder (2011). Exploring behavioural change through an agent-oriented system dynamics model: the use of personal protective equipment among pesticide applicators in colombia. *System Dynamics Review 28*(1), 69–93.

Ford, D. N. (1999). A behavioral approach to feedback loop dominance analysis. *System Dynamics Review 15*, 3–36.

Forrester, N. B. (1982). *A Dynamic Synthesis of Basic Macroeconomic Theory: Implications for Stabilization Policy Analysis*. Ph. D. thesis, Department of Management, College of business administration, University of Nebraska.

Forrester, N. B. (1983). Eigenvalue analysis of dominant feedback loops. Chestnut Hill, MA, USA. The 1st International Conference of the System Dynamics Society. Available at www.systemdynamics.org.

Güneralp, B. (2006). Towards coherent loop dominance analysis: progress in eigenvalue elasticity analysis. *System Dynamics Review 22*(3), 263–289.

Kampmann, C. E. (1996). Feedback loop gains and system behavior. Cambridge, MA, USA. The 14th International Conference of the System Dynamics Society. Available at www.systemdynamics.org.

Kampmann, C. E. and R. Oliva (2006). Loop eigenvalue elasticity analysis: three case studies. *System Dynamics Review 22*(2), 141–162.

Mojtahedzadeh, M. (1997). *A Path Taken: Computer Assisted Heuristics For Understanding Dynamic Systems*. Ph. D. thesis, University at Albany.

Mungovan, D., E. Howley, and J. Duggan (2011, May). The influence of random interactions and decision heuristics on norm evolution in social networks. *Journal of Computational and Mathematical Organization Theory 17*(2), 119–151.

Oliva, R. (2004). Model structure analysis through graph theory: Partition heuristics and feedback structure decomposition. *System Dynamics Review 20*(4), 313–336.

Oliva, R. and M. Mojtahedzadeh (2004). Keep it simple: Dominance assessment of short feedback loops. The 22nd International Conference of the System Dynamics Society.

Osgood, N. (2009). Lightening the performance burden of individual-based models through dimensional analysis and scale modeling. *System Dynamics Review 25*(2), –118.

Phaff, H. G. (2008). Generalized loop deactivation method. Athens, Greece. The 26th International Conference of the System Dynamics Society. Available at www.systemdynamics.org.

Rahmandad, H. and J. Sterman (2008, May). Heterogeneity and network structure in the dynamics of diffusion: Comparing agent-based and differential equation models. *Management Science 54*(5), 998–1014.

Reinschke, K. J. (1988). *Multivariable Control: A Graph-theoretical Approach*. Berlin: Springer-Verlag.

Saleh, M. (2002). *The Characterization of Model Behavior and Its Causal Foundation*. Ph. D. thesis, University of Bergen.

Saleh, M., R. Oliva, C. E.Kampmann, and P. I.Davidsen (2010). A comprehensive analytical approach for policy analysis of system dynamics models. *European Journal of Operational Research 203*, 673 – 683.

Sterman, J. (2000). *Business Dynamics: Systems Thinking and Modeling for a Complex World*. The McGraw-Hill Press.

Warfield, J. (1989). *Societal Systems: Planning, Policy and Complexity*. Intersystems Publications: Salinas, CA.